Το περιβάλλον προγραμματισμού Pencil Code και η γλώσσα CoffeeScript

Δρ. Μαργαρίτα Καραλιοπούλου
Δρ. Ευάγγελος Κανίδης

Διδακτικές σημειώσεις για το περιβάλλον Pencil Code

Γ΄ Γυμνασίου

ΕΛΛΗΝΟΓΕΡΜΑΝΙΚΗ ΑΓΩΓΗ

Το περιβάλλον προγραμματισμού Pencil Code και η γλώσσα CoffeeScript

Διδακτικές σημειώσεις
για το περιβάλλον Pencil Code

για τη Γ' Γυμνασίου

Δρ. Μαργαρίτα Καραλιοπούλου, Δρ. Ευάγγελος Κανίδης

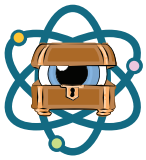
ΣΥΓΓΡΑΦΕΙΣ:

Μαργαρίτα Καραλιοπούλου
PhD, MSc, Εκπαιδευτικός Πληροφορικής

Ευάγγελος Κανίδης
PhD, Σχολικός Σύμβουλος Β Αθήνας και Αν. Αττικής

ΕΠΙΜΕΛΕΙΑ ΕΚΔΟΣΗΣ:

Γεώργιος Παπαδόπουλος
Εκπαιδευτικός Πληροφορικής Ελληνογερμανικής Αγωγής



Ark of Inquiry

Η δημιουργία και αναπαραγωγή του βιβλίου «Το περιβάλλον προγραμματισμού Pencil Code και η γλώσσα CoffeeScript» συγχρηματοδοτήθηκε από την Ευρωπαϊκή Επιτροπή στο πλαίσιο του έργου Ark of Inquiry.
Η πρωτοβουλία Ark of Inquiry, που ξεκίνησε τον Μάρτιο του 2014 και θα ολοκληρωθεί τον Φεβρουάριο του 2018, προσφέρει μια σειρά επιμορφώσεων, σεμιναρίων, εργαστηρίων, βιωματικών δράσεων σε εκπαιδευτικούς όλων των βαθμίδων, με στόχο την εισαγωγή της Υπεύθυνης Έρευνας και Καινοτομίας (ΥΕΚ) στη σχολική τάξη. Συγκεκριμένα, το έργο επιχειρεί να συνδυάσει τη διερευνητική μάθηση με ζητήματα που αφορούν στην ισότητα των φύλων, στην ανοιχτή πρόσβαση, στην ηθική και στη συμμετοχή των πολιτών στη διακυβέρνηση. Η δράση διαθέτει τη διαδικτυακή πύλη Ark of Inquiry (http://www.arkofinquiry.eu) με πλήθος εκπαιδευτικών σεναρίων και δραστηριοτήτων που μπορούν να χρησιμοποιηθούν στην τάξη από μαθητές ηλικίας 7-18 ετών σε διάφορους τομείς των θετικών επιστημών, καθώς και εργαλεία για την αξιολόγηση και την επιβράβευση των μαθητών.

Ευχαριστούμε θερμά τον Δρ. Σοφοκλή Σωτηρίου, Διευθυντή του τμήματος Έρευνας και Ανάπτυξης της Ελληνογερμανικής Αγωγής για την επιστημονική αρωγή στο έργο μας καθώς και το τμήμα του για την τεχνική και διοικητική στήριξη που μας παρείχε κατά τη διάρκεια της συγγραφής του παρόντος βιβλίου.

ISBN: 978-960-473-888-5

Περιεχόμενα

Πρόλογος 5

Μέρος Πρώτο: Το Περιβάλλον του Pencil Code 7
- Τι είναι το Pencil Code 7
- Περιγραφή περιβάλλοντος 8
  - Λογαριασμοί χρηστών 10
  - Αποθήκευση προγράμματος 10
  - Διαμοιρασμός 11
  - Σβήσιμο αρχείου 12
  - Στιγμιότυπο 12
  - Ερώτηση 12
  - Online οδηγός 12
  - Ορατές μόνο συγκεκριμένες περιοχές 12
- Τα 2 περιβάλλοντα: Οι Περιοχές τους 12
  - Περιοχή αποτελέσματος 13
  - Περιοχή δοκιμών 13
  - Περιοχή προγράμματος στο περιβάλλον κώδικα 13
  - Περιοχή προγράμματος στο περιβάλλον με πλακίδια 15
  - Η περιοχή πλακιδίων 16
- Βασικές εντολές της Coffeescript 19
  - Εντολές εξόδου και αριθμητικές πράξεις 19
  - Χρήση της εντολής `write` 20
  - Οι μεταβλητές κι η χρήση τους 22
  - Βασικές εντολές σχεδίασης 23
  - Δομή Επανάληψης: Η εντολή `for` 26
  - Υποπρογράμματα–συναρτήσεις 27
  - Τοπικές μεταβλητές και γενικές μεταβλητές 30
  - Είσοδος δεδομένων 31
  - Δυσκολίες στη χρήση συναρτήσεων 33
  - Εισαγωγή στις λίστες 34
  - Παραδείγματα επαναλήψεων με τιμές μεταβλητής από λίστα 36
  - Δομή επιλογής 37
  - Λογικοί σύνδεσμοι 39
  - Δομή επανάληψης `forever` 40

Μέρος Δεύτερο: Σχέδια μαθημάτων - Φύλλα εργασίας 43
- 1η Διδακτική ώρα: Γνωριμία με το περιβάλλον 43
- 2η Διδακτική ώρα: Εντολές εξόδου 43
  - Φύλλο εργασίας – Εντολές εξόδου: Κείμενο, αριθμοί, ήχος 45
- 3η Διδακτική ώρα: Εντολές εισόδου δεδομένων 46
  - Φύλλο εργασίας – Είσοδος και έξοδος δεδομένων 47
- 4η-5η Διδακτική ώρα: Εντολές σχεδίασης 48
  - Φύλλο εργασίας – Εντολές σχεδίασης 51

6η Διδακτική ώρα: Δομή Επανάληψης: Η εντολή `for` 52
Φύλλο εργασίας – Δομή επανάληψης- η εντολή `for` 53
7η Διδακτική ώρα: Η εντολή for – Κανονικά πολύγωνα 55
Φύλλο εργασίας – Δομή επανάληψης- Κανονικά Πολύγωνα 56
8η-10η Διδακτική ώρα: Υποπρογράμματα-συναρτήσεις 57
Φύλλο εργασίας – Συναρτήσεις 59
Φύλλο εργασίας – Συναρτήσεις-σχήματα εκ περιστροφής 61
Φύλλο εργασίας – Υπερδιαδικασίες- Σπίτι 63
11η -12η Διδακτική ώρα: Συναρτήσεις με μεταβλητές 64
Φύλλο εργασίας (1) – Συναρτήσεις με Μεταβλητές 66
Φύλλο εργασίας (2) – Συναρτήσεις με Μεταβλητές 68
13η-14η Διδακτική ώρα: Η εντολή for με μεταβλητές από λίστα 70
Φύλλο εργασίας (1) – Μεταβλητές και η εντολή `for` 71
Φύλλο εργασίας (2) – Μεταβλητές και η εντολή `for` 72
15η Διδακτική ώρα: Δημιουργώ μεταβλητές 74
Φύλλο εργασίας – Δημιουργώ μεταβλητές 75
16η-17η Διδακτική ώρα: Δομή επιλογής `if` 77
Φύλλο εργασίας (1) – Η εντολή `if` 79
Φύλλο εργασίας (2) – Η εντολή `if` 80
18η – 20η Διδακτική ώρα: Επαναληπτική Δραστηριότητα 81
Φύλλο εργασίας – Λαβύρινθος 82

Πρόλογος

Το παρόν βιβλίο είναι το αποτέλεσμα μιας προσπάθειας για την αξιοποίηση του προγραμματιστικού περιβάλλοντος Pencil Code στην εκπαίδευση. Συγκεκριμένα στα Γυμνάσια, στο πλαίσιο του μαθήματος της Πληροφορικής, προσφέρονται εισαγωγικά μαθήματα προγραμματισμού και στις τρεις τάξεις του Γυμνασίου. Ιδιαίτερα ο προγραμματισμός καλύπτει ένα μεγάλο μέρος της διδασκαλίας στη Γ' Γυμνασίου.

Το διδακτικό βιβλίο της Πληροφορικής στο Γυμνάσιο, το οποίο γράφτηκε το 2006 και χρησιμοποιείται ακόμα, προτείνει για τη Γ' Γυμνασίου τη Logo. Για το Λύκειο προτείνεται στην Α΄ Λυκείου ένα περιβάλλον με πλακίδια, ενώ από τη Β' Λυκείου εισάγεται η αλγοριθμική με χρήση ψευδοκώδικα και στη Γ' Λυκείου μια τεχνητή γλώσσα προγραμματισμού. Στην Ελλάδα, ειδικότερα τα τελευταία χρόνια, χρησιμοποιείται ευρέως το Scratch στην εκπαιδευτική διαδικασία και στο δημοτικό αλλά και στο Γυμνάσιο.

Έχει διαπιστωθεί ερευνητικά ότι υπάρχουν προβλήματα εννοιολογικής μετάβασης των μαθητών από το ένα περιβάλλον, με χρήση πλακιδίων, σε ένα περιβάλλον με χρήση εντολών. Οι μαθητές αντιμετωπίζουν τη δημιουργία προγραμμάτων με πλακίδια ως "παιχνίδι" ή γενικότερα μια δραστηριότητα που δεν έχει σχέση με τον πραγματικό προγραμματισμό.

Η χρήση του περιβάλλοντος Pencil Code επιτρέπει στους μαθητές να επιλέγουν το περιβάλλον προγραμματισμού που επιθυμούν να εργαστούν. Υπάρχει παράλληλη χρήση ενός περιβάλλοντος με πλακίδια και ενός περιβάλλοντος με συγγραφή εντολών. Μόλις ο μαθητής χρησιμοποιήσει μια εντολή σε ένα από τα δύο περιβάλλοντα, αυτή "μεταφράζεται" αυτόματα και εμφανίζεται και στο άλλο περιβάλλον.

Το παρόν βιβλίο περιέχει στο Α' μέρος μια εισαγωγή στο περιβάλλον του Pencil Code, όπου περιγράφονται αναλυτικά όλες οι εντολές και οι δυνατότητες του περιβάλλοντος. Στο Β' μέρος περιέχει μια σειρά από εκπαιδευτικές δραστηριότητες με συγκεκριμένα φύλλα εργασίας για κάθε μια από αυτές. Οι δραστηριότητες αυτές έχουν υλοποιηθεί από τους εκπαιδευτικούς που συμμετείχαν στην προσπάθεια αυτή και έχουν βελτιωθεί από τις παρατηρήσεις τους.

Στο Α' μέρος, όπου περιγράφεται το περιβάλλον, πολλά στοιχεία έχουν ληφθεί από το επίσημο περιβάλλον του Pencil Code το www.pencilcode.org και τις πηγές που παρέχει.

Η προσπάθεια αξιοποίησης του Pencil Code υποστηρίχτηκε από την Ελληνογερμανική Αγωγή και συνδέθηκε με άλλες διεθνείς δραστηριότητες του σχολείου στο πλαίσιο του Ark of Inquiry, ενώ οι δραστηριότητες αναρτήθηκαν στην πλατφόρμα Inspiring Science Education. Η Ελληνογερμανική Αγωγή χρηματοδότησε και την έκδοση αυτού του βιβλίου.

Οι εκπαιδευτικοί που συμμετείχαν στην προσπάθεια είναι:

Γαρδίκη Ιωάννα	1ο Γυμνάσιο Βούλας
Γροντάς Παναγιώτης	Καλλιτεχνικό Γέρακα
Καραλιοπούλου Μαργαρίτα	2ο Γυμνάσιο Παιανίας
Κουτρουμπάς Παύλος	Γυμνάσιο Αγίου Στεφάνου
Μαραβέλια Σοφία	Γυμνάσιο Παπάγου
Μπουτσιάνη Κωνσταντίνα	1ο Γυμνάσιο Βριλησσίων
Παναγιωτοπούλου Κωνσταντίνα	3ο Γυμνάσιο Γέρακα
Παντελοπούλου Σταυρούλα	2ο Γυμνάσιο Αρτέμιδος
Παπαδοπούλου Ελένη	1ο Γυμνάσιο Καλυβίων
Παπαδοπούλου Μαριάνθη	1ο Γυμνάσιο Παιανίας

Παπαμιχαλοπούλου Ρούλα	5ο Γυμνάσιο Χαλανδρίου
Περτσινίδου Κυριακή	1ο Γυμνάσιο Γέρακα
Σκιαδά Ευφροσύνη	2ο Γυμνάσιο Παλλήνης
Τόγια Αντωνία	3ο Γυμνάσιο Γέρακα
Χατζηϊωάννου Άσπα	1ο Γυμνάσιο Νέου Ψυχικού

Επικεφαλής της ομάδας και υπεύθυνοι για την παραγωγή και την επιμέλεια του υλικού είναι: Ευάγγελος Κανίδης, Σχολικός Σύμβουλος Πληροφορικής Β' Αθήνας και Ανατολικής Αττικής και Μαργαρίτα Καραλιοπούλου, Εκπαιδευτικός Πληροφορικής, 2ο Γυμνάσιο Παιανίας.

Μέρος Πρώτο: Το Περιβάλλον του Pencil Code

Τι είναι το Pencil Code

Το Pencil Code είναι ένα προγραμματιστικό περιβάλλον ανοιχτού κώδικα, το οποίο επιτρέπει την παράλληλη συγγραφή και επεξεργασία κώδικα σε δύο διαφορετικές μορφές. Συνδυάζει την αναπαραστατική δύναμη ενός γραφικού περιβάλλοντος με πλακίδια με την πυκνότητα και την απλότητα μιας απλής γλώσσας προγραμματισμού. Διαθέτει αρκετές επεκτάσεις που της επιτρέπουν να συνδυάζει την κλασική χελώνα μιας γλώσσας τύπου Logo με εξελιγμένες δυνατότητες αναπαράστασης γραφικών.

Το Pencil Code είναι ένα περιβάλλον προγραμματισμού κατάλληλο για εκπαιδευτικούς σκοπούς κυρίως για αρχάριους προγραμματιστές. Στην πραγματικότητα είναι δύο προγραμματιστικά περιβάλλοντα. Ο χρήστης έχει τη δυνατότητα να επιλέγει ανάμεσα στα δύο αυτά διαφορετικά περιβάλλοντα. Μπορεί να δημιουργήσει το πρόγραμμά του είτε συναρμολογώντας πλακίδια κώδικα είτε συντάσσοντας εντολές. Επιπλέον έχει τη δυνατότητα να μεταβαίνει από το ένα περιβάλλον στο άλλο κατά τη δημιουργία ενός προγράμματος.

Το περιβάλλον με τα πλακίδια επιτρέπει στον χρήστη να επικεντρωθεί στην υλοποίηση του προγράμματος χωρίς να τον απασχολεί η ακριβής σύνταξη των εντολών της γλώσσας. Αντίστοιχα το περιβάλλον με κώδικα προσφέρει μια εικόνα «επαγγελματικού» προγραμματισμού. Ο μαθητής, καθώς μπορεί να μεταφέρεται από το ένα περιβάλλον στο άλλο, κατανοεί ότι προγραμματίζει με μια πραγματική γλώσσα προγραμματισμού και αυτό που υλοποιεί με πλακίδια δεν είναι ένα παιχνίδι αλλά ένα πρόγραμμα. Με τον τρόπο αυτό ο μαθητής αποκτά εμπιστοσύνη στις προγραμματιστικές του ικανότητες και είναι ικανός να χειριστεί μια πραγματική γλώσσα προγραμματισμού σε μεγαλύτερες ηλικίες.

Στο Pencil Code στο περιβάλλον προγραμματισμού με κώδικα υπάρχει δυνατότητα επιλογής ανάμεσα σε διαφορετικές γλώσσες προγραμματισμού (Coffeescript, Javascript, CSS, html).

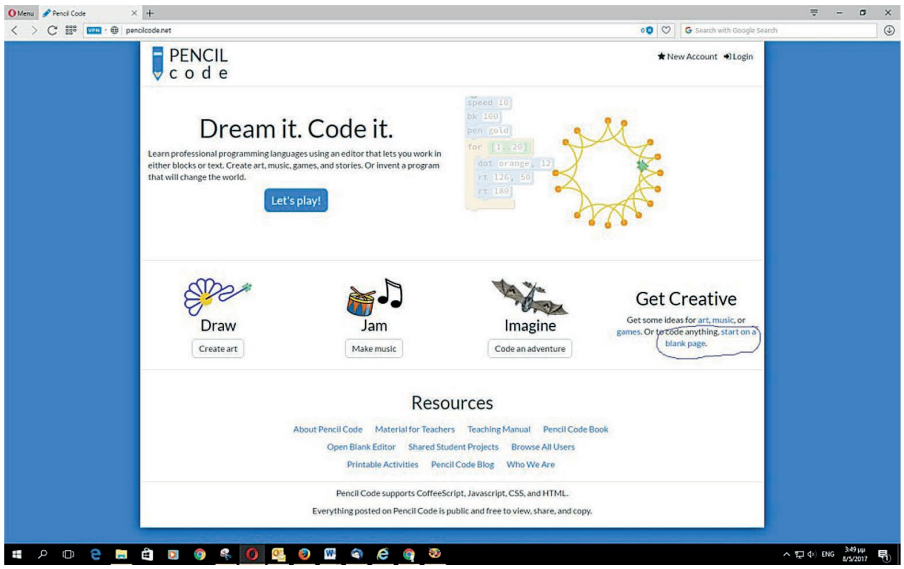
Η γλώσσα προγραμματισμού με την οποία θα ασχοληθούμε στο παρόν εγχειρίδιο και προτείνεται για διδασκαλία στο Γυμνάσιο είναι η CoffeeScript[1]. Η CoffeeScript είναι επαγγελματική γλώσσα αλλά θεωρείται αρκετά απλή για αρχάριους προγραμματιστές. Είναι μια γλώσσα που μεταγλωττίζεται σε Javascript. Το συντακτικό της είναι επηρεασμένο από την Python και τη Ruby και έχει στοιχεία και από τις δύο γλώσσες. Μπορεί να χρησιμοποιηθεί και ανεξάρτητα από το Pencil Code αλλά το βιβλίο αυτό αναφέρεται στην υλοποίησή της μέσα στο Pencil Code.

Επίσης για τον προγραμματισμό της χελώνας πέρα από απλές εντολές σε πλακίδια ή σε εντολές της CoffeeScript, θα χρησιμοποιηθούν και βιβλιοθήκες της CoffeeScript.

1. Γράφτηκε από τον Jeremy Ashkenas το 2009

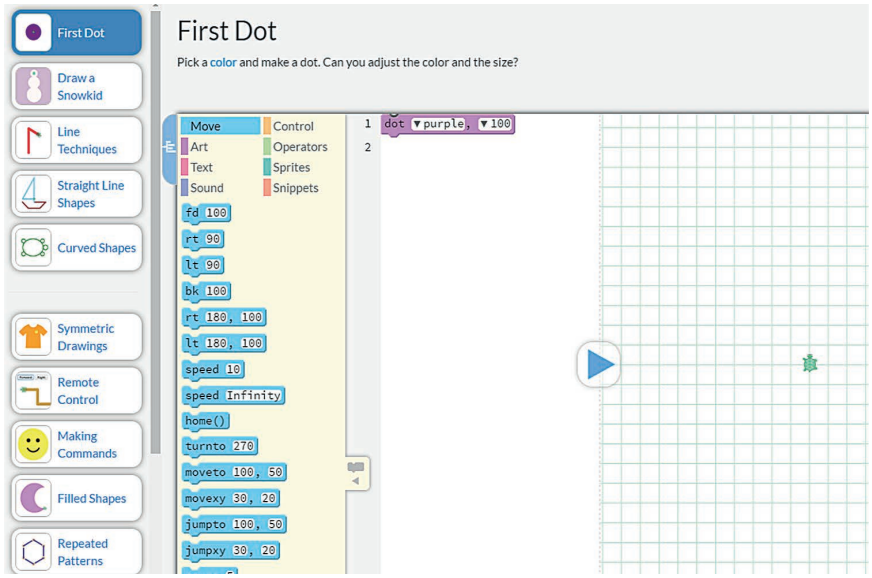
Περιγραφή περιβάλλοντος

Το περιβάλλον προγραμματισμού Pencil Code είναι διαθέσιμο online στη διεύθυνση http://pencilcode.net. Είναι διαδικτυακό περιβάλλον και μπορεί να χρησιμοποιηθεί μόνο μέσα από κάποιο φυλλομετρητή. Το γεγονός αυτό έχει πλεονεκτήματα και μειονεκτήματα. Βασικό μειονέκτημα είναι ότι χωρίς πρόσβαση στο Internet δε μπορούμε να το χρησιμοποιήσουμε. Τα πλεονεκτήματα όμως είναι περισσότερα: Δεν υπάρχουν προβλήματα ασυμβατότητας υλικού και λειτουργικού συστήματος. Είναι προσβάσιμο από τον διδάσκοντα και τους μαθητές από οποιοδήποτε μέρος (και από το σπίτι). Οι εργασίες κάθε μαθητή αποθηκεύονται στο περιβάλλον και είναι προσβάσιμες από τον μαθητή ανεξάρτητα από θέση υπολογιστή. Τελικά έχει διαπιστωθεί ότι είναι ένα «ελαφρύ» περιβάλλον που μπορεί να λειτουργήσει σε σχολικά εργαστήρια.



Εικόνα 1: Η αρχική εικόνα του περιβάλλοντος

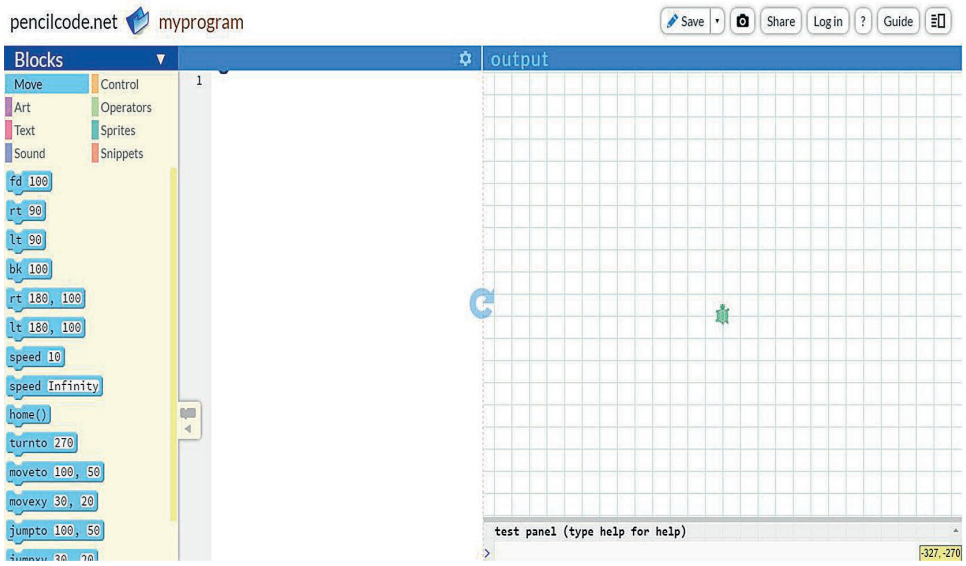
Στην αρχική εικόνα της σελίδας ξεχωρίζουν στο κέντρο οι περιοχές Draw, Jam, Imagine και Get Creative (Εικόνα 1). Οι τρεις πρώτες περιέχουν παραδείγματα έτοιμα προς εκτέλεση. Για παράδειγμα η Εικόνα 2 εμφανίζει τα περιεχόμενα της Draw.



Εικόνα 2: Η περιοχή Draw

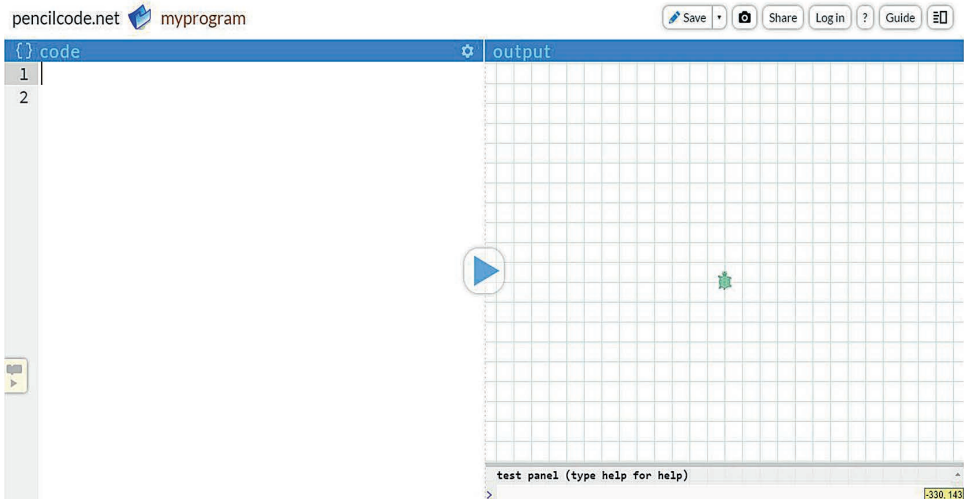
Στην τέταρτη περιοχή (Get Creative) εκτός από έτοιμα παραδείγματα μπορείτε να ξεκινήσετε από μια "λευκή" σελίδα (Start on a blank page) και να δημιουργήσετε το δικό σας πρόγραμμα.

Αν ξεκινήσετε με αυτή την επιλογή, εμφανίζεται η Εικόνα 3, που είναι το περιβάλλον προγραμματισμού με πλακίδια. Για να μεταβείτε στο περιβάλλον προγραμματισμού με κώδικα (Εικόνα 4) θα πρέπει να πατήσετε πάνω στο κουμπί που φαίνεται στη δεξιά εικόνα και βρίσκεται ενδιάμεσα των δύο περιοχών (κάτω αριστερά).



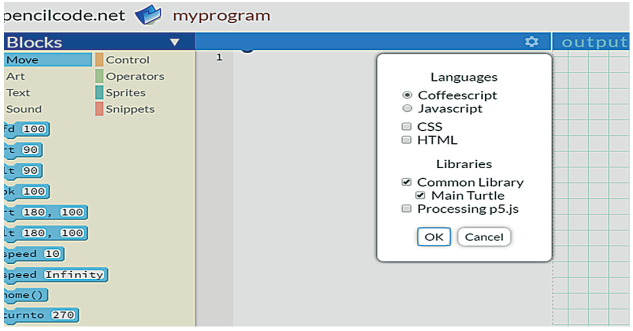
Εικόνα 3: Περιβάλλον με πλακίδια

Αν είμαστε στο περιβάλλον με κώδικα, μπορούμε να γυρίσουμε στο περιβάλλον με πλακίδια (εκτός από το κουμπί αλλαγής) πατώντας με το ποντίκι μας πάνω στη λέξη code που βρίσκεται πάνω αριστερά.



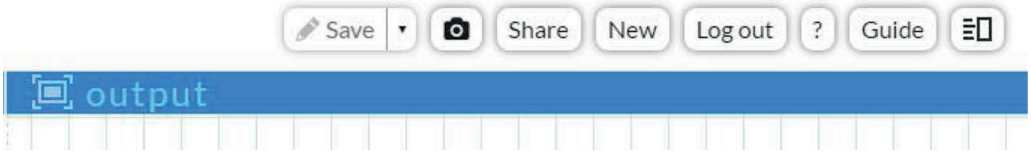
Εικόνα 4: Περιβάλλον με κώδικα

Πατώντας πάνω στο γρανάζι επιλέγουμε τη γλώσσα και τις βιβλιοθήκες (Εικόνα 5).



Εικόνα 5: Οι διαθέσιμες γλώσσες

Και στα δυο περιβάλλοντα, στο πάνω μέρος υπάρχει ένα μενού, οι λειτουργίες του οποίου θα αναλυθούν στη συνέχεια.

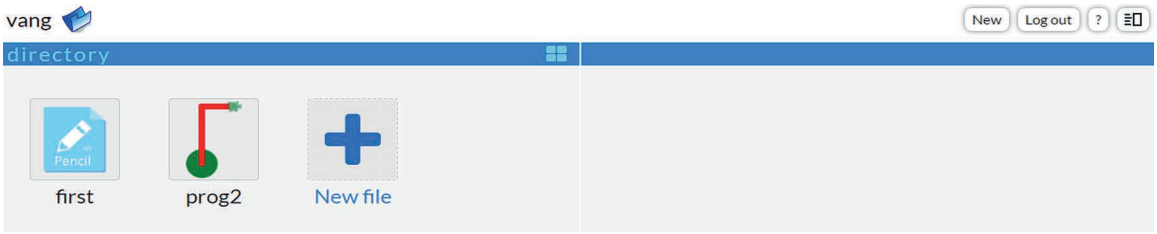


Λογαριασμοί χρηστών

Όπως περιγράψαμε στην προηγούμενη παράγραφο, στην πρώτη σελίδα του προγράμματος, επιλέγοντας το Start on a blank page μπορείτε να ξεκινήσετε τη δημιουργία ενός δικού σας προγράμματος. Αυτό το πρόγραμμα έχει την ονομασία myprogram. Το πρόγραμμα αυτό μπορείτε να το εκτελέσετε αλλά δεν έχετε τη δυνατότητα να το αποθηκεύσετε στο περιβάλλον του Pencil Code, αν πρώτα δεν δημιουργήσετε ένα λογαριασμό στο περιβάλλον. Το πιο πιθανό είναι να επιθυμείτε την αποθήκευση των προγραμμάτων που δημιουργούν.

Στο πάνω δεξιά άκρο της αρχικής οθόνης παρατηρήστε ότι υπάρχουν οι επιλογές που εμφανίζονται στη δεξιά εικόνα. Από τις επιλογές αυτές μπορείτε να δημιουργήσετε ένα νέο λογαριασμό. Τότε το Pencil Code θα δημιουργήσει ένα δικό σας χώρο όπου μπορούν να αποθηκεύονται όλα τα προγράμματα που θα δημιουργήσετε.

Μπορείτε να δημιουργήσετε άφοβα ένα νέο λογαριασμό τόσο για εσάς όσο και για τους μαθητές σας. Το Pencil Code για τη δημιουργία λογαριασμού δεν ζητά προσωπικά στοιχεία, ούτε λογαριασμό ηλεκτρονικού ταχυδρομείου (mail). Αρκεί ένα όνομα και ένα συνθηματικό. Φυσικά θα πρέπει να ζητήσετε από τους μαθητές να σημειώσουν τα στοιχεία αυτά. Την επόμενη φορά που θα χρησιμοποιήσετε το περιβάλλον του Pencil Code, θα πατήσετε το Login, θα δώσετε το όνομα και το συνθηματικό σας και θα μεταφερθείτε στον προσωπικό σας φάκελο που βρίσκεται στο περιβάλλον. Η παρακάτω εικόνα δείχνει έναν τέτοιο χώρο (φάκελο) ενός χρήστη με όνομα vang.

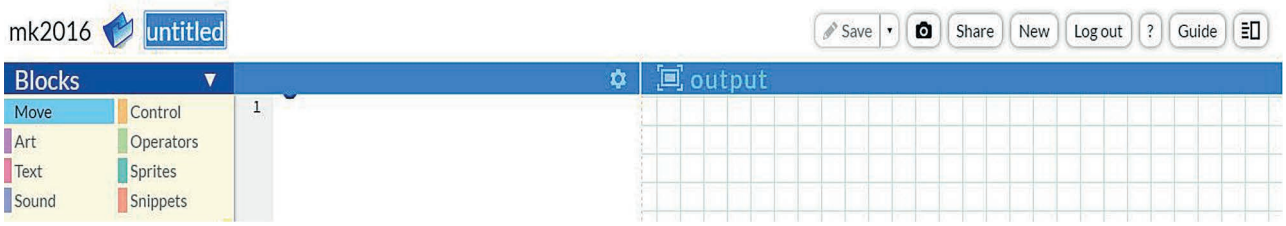


Εικόνα 6: Ο φάκελος ενός χρήστη

Παρατηρούμε ότι στο φάκελο υπάρχουν δύο έτοιμα προγράμματα. Αν πατήσουμε πάνω στο New file θα μπορέσουμε να γράψουμε ένα νέο πρόγραμμα. Μπορούμε να επιλέξουμε είτε το περιβάλλον με πλακίδια είτε το περιβάλλον με κώδικα. Το πρόγραμμά μας μπορεί να αποθηκευτεί σε ένα αρχείο.

Αποθήκευση προγράμματος

Για να αποθηκευτεί ένα πρόγραμμα, αν έχουμε ήδη συνδεθεί στο περιβάλλον, κάνουμε τα παρακάτω βήματα:



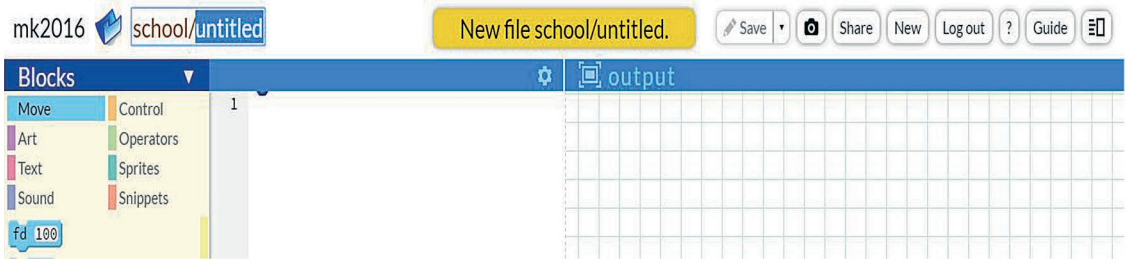
1. Δίνουμε όνομα στο αρχείο μας αντικαθιστώντας το "untitled"

2. Πατάμε save στο μενού

Το αρχείο μας αποθηκεύεται ως μια σελίδα. Για παράδειγμα έστω ότι ο χρήστης mk2016 έχει δημιουργήσει ένα πρόγραμμα και το αποθηκεύει με το όνομα "flower". Τότε δημιουργείται μια ιστοσελίδα με διεύθυνση https://mk2016.pencilcode.net/edit/flower

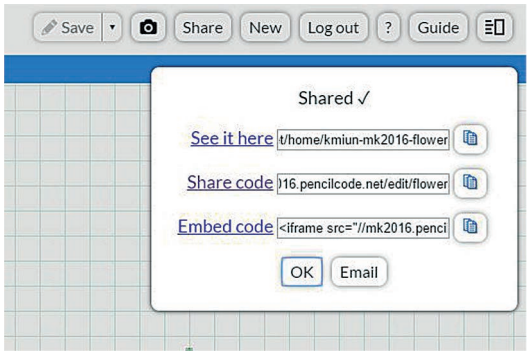
Υπάρχει η δυνατότητα να δημιουργούμε φακέλους για την καλύτερη οργάνωση των αποθηκευμένων προγραμμάτων μας. Για παράδειγμα, έστω ότι έχουμε δημιουργήσει ένα πρόγραμμα και θέλουμε να το αποθηκεύσουμε σε ένα φάκελο με το όνομα "school" ως αρχείο με το όνομα "diorofo". Για να γίνει αυτό, αρκεί να γράψουμε το όνομα του αρχείου ως εξής: "school/diorofo" και αποθηκεύεται ως μια ιστοσελίδα με διεύθυνση https://mk2016.pencilcode.net/edit/school/diorofo

Πατώντας New έχουμε τη δυνατότητα να δημιουργήσουμε νέο αρχείο στον ίδιο φάκελο.



Διαμοιρασμός

Έχουμε τη δυνατότητα να διαμοιράσουμε το πρόγραμμα μας επιλέγοντας Share στο μενού.



Η επιλογή See it here μας επιτρέπει να διαμοιραστούμε το αποτέλεσμα (output) της εκτέλεσης του προγράμματος. Η επιλογή Share code μας επιτρέπει να διαμοιραστούμε τον κώδικα. Η επιλογή Embed code μας επιτρέπει να ενσωματώσουμε τον κώδικα σε μια ιστοσελίδα ώστε να είναι ορατό το αποτέλεσμα.

Σβήσιμο αρχείου

Για να σβήσουμε ένα αρχείο, αρκεί να διαγράψουμε όλο το περιεχόμενο του, δηλαδή να σβήσουμε όλο το πρόγραμμα και να πατήσουμε save στο μενού.

Στιγμιότυπο

Επιλέγοντας την εικόνα στο μενού, μπορούμε να πάρουμε στιγμιότυπο του αποτελέσματός μας. Το στιγμιότυπο αυτό με δεξί κλικ του ποντικιού μας, μπορούμε να το αποθηκεύσουμε/ αντιγράψουμε. Στο στιγμιότυπο αυτό δεν φαίνονται τα τετραγωνάκια του πλέγματος στο πίσω μέρος.

Ερώτηση

Επιλέγοντας την εικόνα στο μενού, εμφανίζεται

Αν πατήσουμε πάνω στο Ask a question, μας στέλνει σε googlegroup για το Pencil Code https://groups.google.com/forum/#!forum/pencilcode, ενώ με το Change password μπορούμε να αλλάξουμε συνθηματικό.

Online οδηγός

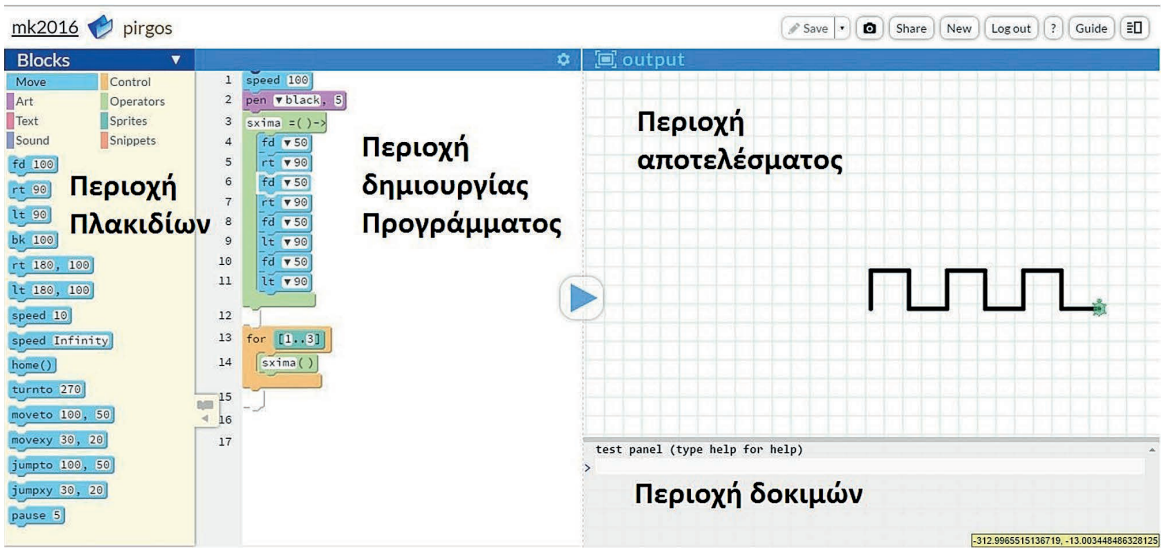
Επιλέγοντας το Guide στο μενού, ανοίγει ένας Online οδηγός https://guide.pencilcode.net/home/

Ορατές μόνο συγκεκριμένες περιοχές

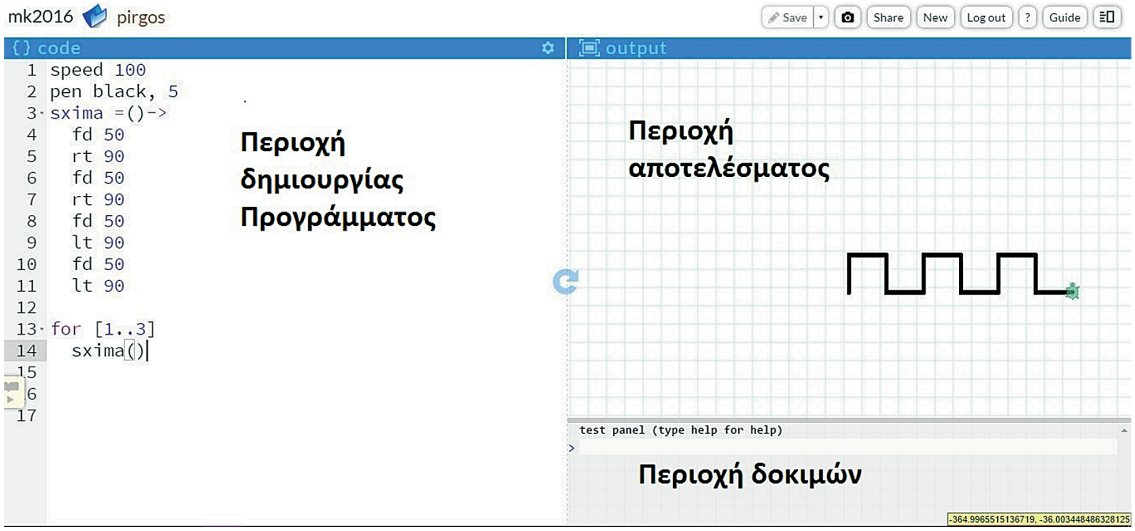
Αν επιλέξουμε κρύβουμε/ εμφανίζουμε το αποτέλεσμα του προγράμματα μας. Αν πατήσουμε πάνω στο output, ανοίγει μια νέα σελίδα (νέο παράθυρο) όπου είναι ορατό μόνο το αποτέλεσμα.

Τα 2 περιβάλλοντα: Οι Περιοχές τους

Στις παρακάτω εικόνες (Εικόνα 7, Εικόνα 8) μπορούμε να δούμε το ίδιο πρόγραμμα, στο περιβάλλον με πλακίδια και στο περιβάλλον με κώδικα. Διακρίνουμε τις περιοχές δημιουργίας προγράμματος, αποτελέσματος και δοκιμών.



Εικόνα 7 Περιοχές περιβάλλοντος με πλακίδια



Εικόνα 8: Περιοχές περιβάλλοντος με κώδικα

Περιοχή αποτελέσματος

Η περιοχή του αποτελέσματος είναι ίδια είτε είμαστε στο περιβάλλον με πλακίδια είτε είμαστε στο περιβάλλον με κώδικα. Στο φόντο υπάρχουν ως καμβάς, τετραγωνάκια με μήκος πλευράς 25 εικονοστοιχεία, ώστε ο χρήστης να διευκολυνθεί στο σχεδιασμό γεωμετρικών σχημάτων. Η χελώνα βρίσκεται σε ένα μη ορατό σύστημα αξόνων. Η αρχική της θέση είναι στην αρχή των αξόνων (0,0). Στο κάτω δεξιά μέρος του παραθύρου μπορούμε να βλέπουμε τις καρτεσιανές συντεταγμένες του σημείου που δείχνει το ποντίκι μας.

Περιοχή δοκιμών

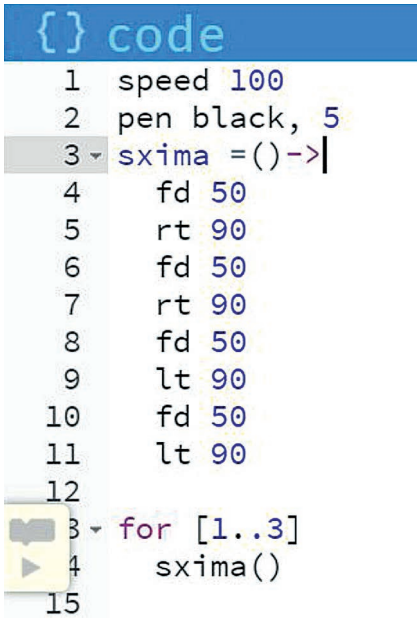
Στην περιοχή κάτω δεξιά (4) μπορούμε να δοκιμάζουμε εντολές και να έχουμε υπόδειξη για τη σύνταξη τους.

Περιοχή προγράμματος στο περιβάλλον κώδικα

Στην περιοχή προγράμματος πληκτρολογούμε τις εντολές και συντάσσουμε το πρόγραμμα μας.

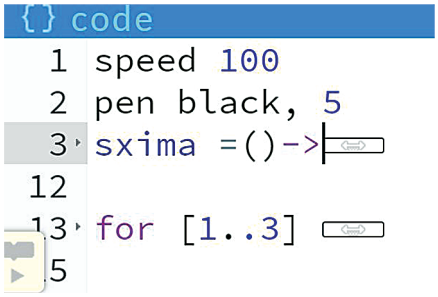
Ορατά τμήματα κώδικα

Όπως θα δούμε στη συνέχεια, κάποιες προγραμματιστικές δομές, όπως για παράδειγμα η δομή επανάληψης, είναι απαραίτητο να γράφουμε τις εντολές που θέλουμε να επαναληφθούν με μια εσοχή. Υπάρχει η δυνατότητα αυτές τις εντολές να τις εμφανίζουμε ή να τις κρύβουμε.



Εικόνα 9: Ορατό όλο το πρόγραμμα

Για παράδειγμα, στο πρόγραμμα που φαίνεται στην Εικόνα 9, αν πατήσουμε πάνω στα βελάκια που φαίνονται αριστερά των λέξεων `sxima` (συνάρτηση) και `for` (δομή επανάληψης), τότε οι «περιεχόμενες» εντολές σε αυτές τις δομές θα κρυφτούν και θα έχουμε την Εικόνα 10. Τα βελάκια αριστερά των λέξεων μας πλέον «κοιτούν» δεξιά. Οι εντολές γίνονται πάλι ορατές, είτε πατώντας τα βελάκια αυτά, είτε δεξιά στα μικρά πλαίσια με το σύμβολο της ισοδυναμίας.



Εικόνα 10 Μη ορατές εντολές του προγράμματος

Σχόλια

Σε κάθε γλώσσα προγραμματισμού χρησιμοποιούνται σχόλια για τη διευκρίνηση του κώδικα που χρησιμοποιούμε. Στην CoffeeScript τα σχόλια είναι δύο ειδών:

- Τα σχόλια που καταλαμβάνουν μια γραμμή και αρχίζουν με τον χαρακτήρα #. Π.χ.

  # αυτό είναι ένα σχόλιο

- Τα σχόλια που καταλαμβάνουν πολλές γραμμές και ορίζονται από τους χαρακτήρες ###. Π.χ.

###

Αυτό είναι ένα σχόλιο που καταλαμβάνει περισσότερες από μια γραμμές. Μπορείτε να γράψετε όσες γραμμές θέλετε αρκεί να τις βάλετε ανάμεσα σε τρεις χαρακτήρες σχολίων.

###

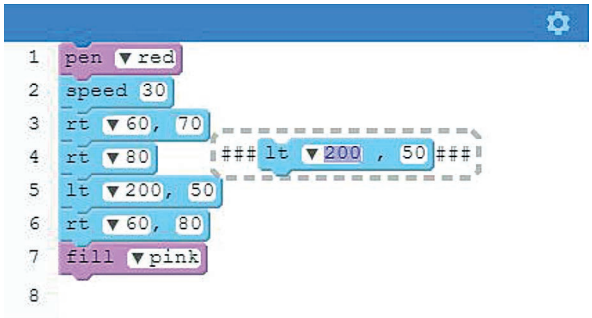
Αν βάλουμε # μπροστά από μια εντολή, τότε την κάνουμε «ανεργή», δεν είναι πλέον εκτελέσιμη δηλαδή είναι σα σχόλιο. Αντίστοιχα βάζοντας μια ομάδα εντολών ανάμεσα σε δύο ###, οι εντολές αυτές δεν είναι εκτελέσιμες.

Περιοχή προγράμματος στο περιβάλλον με πλακίδια

Για τη δημιουργία προγράμματος στο περιβάλλον με πλακίδια, σύρουμε πλακίδια από την περιοχή πλακιδίων στην περιοχή δημιουργίας προγράμματος. Για να εκτελεστεί ένα πρόγραμμα θα πρέπει να πατήσουμε το βέλος  που βρίσκεται στο μέσο της οθόνης (βλέπε Εικόνα 7).

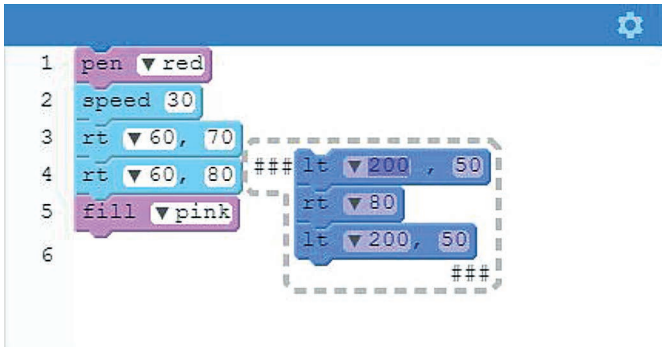
Τα πλακίδια συχνά έχουν υποδοχές οι οποίες επιτρέπεται να γεμίζουν με άλλα πλακίδια ή προεξοχές και εσοχές για να "συνδέονται" με άλλα πλακίδια. Για παράδειγμα, μια δομή επανάληψης μπορεί να έχει υποδοχές για τις εντολές που θα επαναληφθούν. Τα πλακίδια αυτά μπορεί να αντιπροσωπεύουν δομές υψηλού επιπέδου, όπως επαναλήψεις ή χαμηλού επιπέδου όπως η απόδοση τιμής σε μια μεταβλητή. Κάθε πλακίδιο έχει μια ταμπέλα, όπου αναγράφεται η εντολή που αντιπροσωπεύει και συχνά έχει διαφορετικό σχήμα και χρώμα (το οποίο σχετίζεται με την ομάδα που ανήκει η εντολή) ώστε να βοηθηθεί ο χρήστης να το τοποθετήσει στην κατάλληλη θέση.

Μπορούμε να προσθέτουμε πλακίδια στο πρόγραμμα μας –σύροντάς τα από την περιοχή πλακιδίων–, να αφαιρούμε –σα να τα «πετάμε» στην περιοχή πλακιδίων– ή να τα αφαιρούμε προσωρινά –μεταφέροντας τα εντός της περιοχής σύνταξης του προγράμματος αλλά στην άκρη της, εκτός ροής σα σημείωση– (Εικόνα 11) ή μεταφέροντάς τα σε άλλο σημείο του προγράμματος με την τεχνική του «σύρε και άφησε».



Εικόνα 11: Προσωρινή διαγραφή

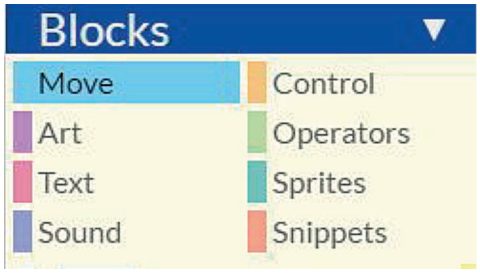
Μπορούμε να μετακινήσουμε/ μεταφέρουμε περισσότερα από ένα πλακίδια αρκεί να επιλέξουμε/ μαρκάρουμε με το ποντίκι μας την περιοχή τους.



Η περιοχή πλακιδίων

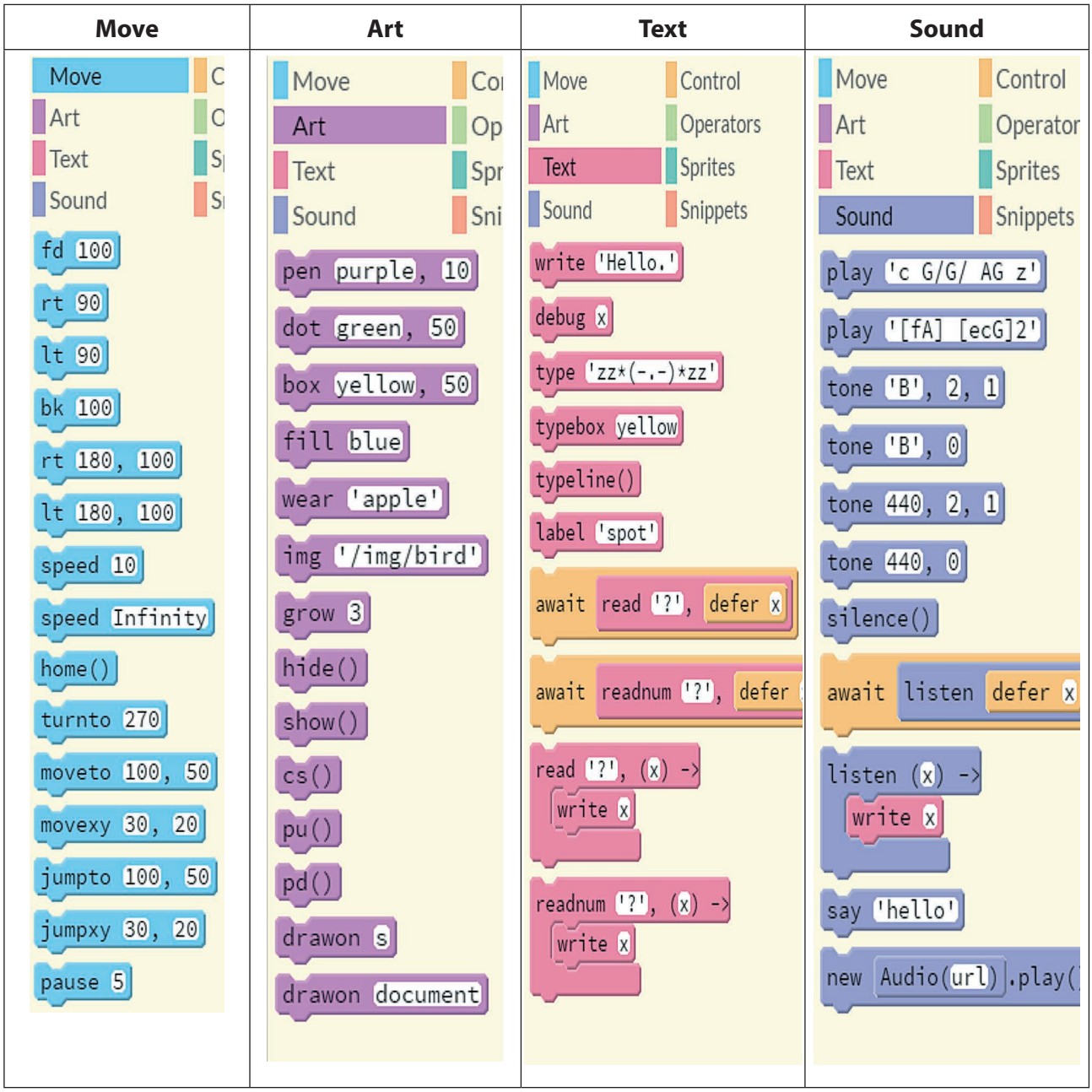
Η περιοχή εντολών πλακιδίων περιέχει οκτώ διαφορετικές ομάδες πλακιδίων με τις οποίες μπορούμε να δημιουργήσουμε ένα πρόγραμμα. (Εικόνα 13, Εικόνα 14)

Κάθε ομάδα πλακιδίων έχει συγκεκριμένο χρώμα ώστε να είναι εύκολη η ομαδοποίηση και ο εντοπισμός συγκεκριμένου πλακιδίου. Κάθε πλακίδιο αντιστοιχεί σε μία εντολή, σε μία δομή, σε μία συνθήκη και γενικότερα σε μία λειτουργία ενός προγράμματος. Οι ομάδες των πλακιδίων είναι:

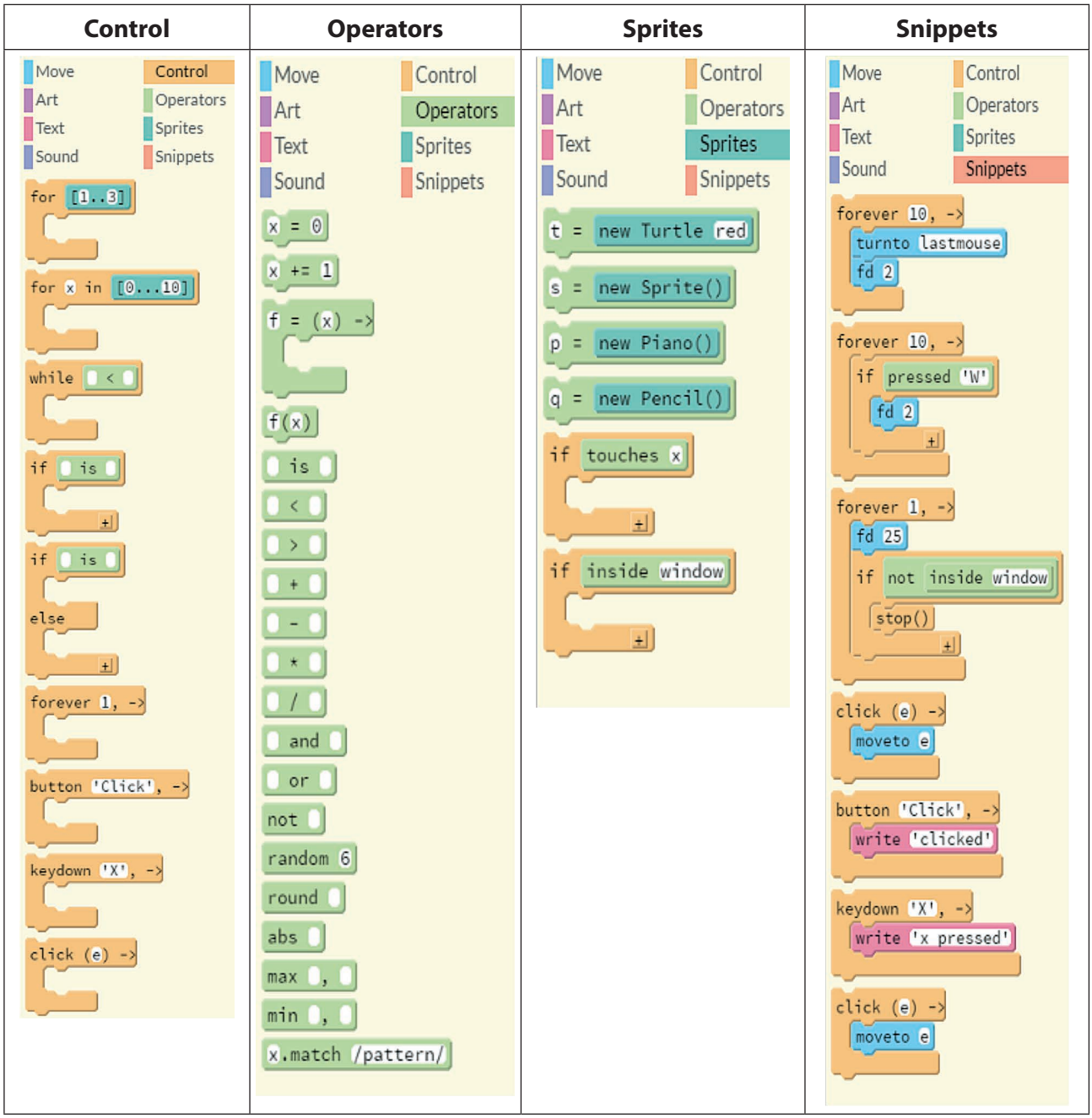


Εικόνα 12: Οι ομάδες πλακίδιων

Τα διαθέσιμα πλακίδια - εντολές εμφανίζονται συνοπτικά στις ακόλουθες δύο εικόνες:



Εικόνα 13: Οι διαθέσιμες εντολές των ομάδων Move, Art, Text, Sound



Εικόνα 14: Οι διαθέσιμες εντολές των ομάδων Control, Operators, Sprites, Snippets

Κάποιες από τις εντολές που αναγράφονται στα πλακίδια θα αναλυθούν στην επόμενη ενότητα.

Βασικές εντολές της Coffeescript

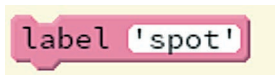
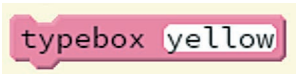
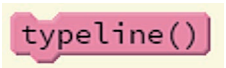
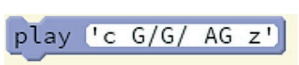
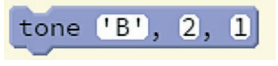
Κάποιες εντολές της Coffeescript είναι διαθέσιμες σε μορφή πλακιδίων. Υπάρχουν όμως και εντολές οι οποίες μπορούν να χρησιμοποιηθούν στο περιβάλλον με κώδικα αλλά δεν υπάρχουν στις αρχικές οικογένειες πλακιδίων. Αν χρησιμοποιήσουμε μια τέτοια εντολή στο περιβάλλον με κώδικα κατά τη δημιουργία ενός προγράμματος, τότε, αν δούμε το πρόγραμμα μας με πλακίδια, θα παρατηρήσουμε ότι το περιβάλλον δημιουργεί προσωρινά ένα αντίστοιχο πλακίδιο.

Εντολές εξόδου και αριθμητικές πράξεις

Μια βασική δυνατότητα που έχουν οι υπολογιστές και κατ' επέκταση τα προγραμματιστικά περιβάλλοντα είναι η εκτέλεση αριθμητικών πράξεων. Ιστορικά μάλιστα οι υπολογιστές χρησιμοποιήθηκαν κυρίως για την εκτέλεση σύνθετων υπολογισμών. Φυσικά σε ένα προγραμματιστικό περιβάλλον θα πρέπει να υπάρχει δυνατότητα να πάρουμε με κάποιο τρόπο το αποτέλεσμα μιας πράξης. Το έργο αυτό το αναλαμβάνουν οι εντολές εξόδου. Οι εντολές εξόδου αναλαμβάνουν να εξάγουν κάτι είτε στην οθόνη είτε στα ηχεία, είτε σε χαρτί. Κάθε εφαρμογή διαθέτει εντολές για εισαγωγή δεδομένων από το πληκτρολόγιο και το ποντίκι και εξαγωγή του αποτελέσματος σε οθόνη και ηχεία.

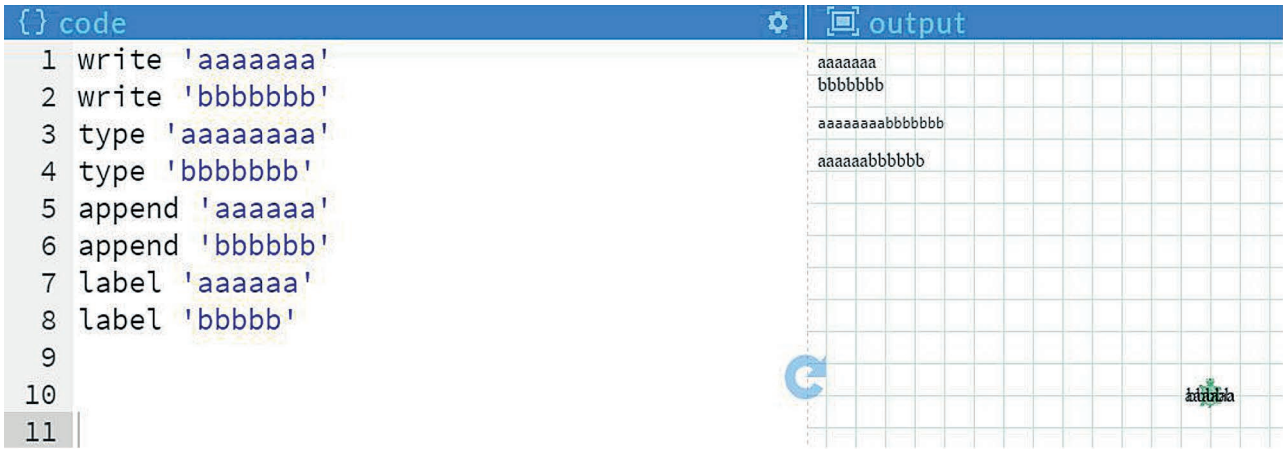
Οι εντολές εξόδου στην οθόνη βρίσκονται στην οικογένεια πλακιδίων **Text** και είναι χρωματισμένες ροζ, ενώ οι εντολές εξόδου στα ηχεία βρίσκονται στην οικογένεια πλακιδίων **Sound** και είναι γαλάζιο-μωβ.

Οι κυριότερες εντολές είναι:

`write`	Εμφανίζει το κείμενο μας στην οθόνη	
`label`	Εμφανίζει το κείμενο μας στη θέση που είναι η χελώνα. Συντάσσεται όμοια με τη write	
`type`	Εμφανίζει το κείμενο μας να είναι από γραφομηχανή	
`typeline`	Αλλαγή γραμμής. Εμφανίζεται νέα γραμμή	
`append`	Προσθέτει κείμενο χωρίς αλλαγή γραμμής	Δεν υπάρχει αντίστοιχο πλακίδιο
`say`	Έξοδο στα ηχεία. Συντάσσεται όμοια με τη write.	
`play`	Παίζει μελωδία	
`tone`	Ακούγεται τόνος	

`alert`	Εμφανίζει το κείμενο μας σε ένα ειδικό παράθυρο. Συντάσσεται όμοια με τη `write`.	Δεν υπάρχει αντίστοιχο πλακίδιο[2]

Στην επόμενη εικόνα μπορούμε να συγκρίνουμε τα διαφορετικά αποτελέσματα που προκύπτουν κατά τη χρήση των εντολών `write`, `type`, `append`, `label`



Χρήση της εντολής `write`

I. Εμφάνιση αποτελέσματος αριθμητικών πράξεων (αριθμητική έκφραση)

Οι αριθμητικές πράξεις πραγματοποιούνται μεταξύ αριθμών με τη βοήθεια των γνωστών συμβόλων + για πρόσθεση, - για αφαίρεση, * για πολλαπλασιασμό και / για διαίρεση

Για την εμφάνιση του αποτελέσματος χρησιμοποιούμε την εντολή `write` και δίπλα την αριθμητική έκφραση. Π.χ. `write 20 / 4` (αποτέλεσμα 5)

Ισχύει η προτεραιότητα πράξεων όπως την ξέρουμε στα μαθηματικά αλλά σε περιπτώσεις σύνθετων πράξεων, αν υπάρχει αμφιβολία, καλό είναι να χρησιμοποιούνται παρενθέσεις.

Εκτός από αριθμητικές εκφράσεις δίπλα στην `write` μπορούν να χρησιμοποιηθούν μαθηματικές συναρτήσεις. Π.χ. `write round 6,5`

II. Εμφάνιση αλφαριθμητικών

Τα *αλφαριθμητικά* (strings), μπορεί να είναι φράσεις, λέξεις ή και αριθμοί που δε γίνονται αριθμητικές πράξεις.

2. Π.χ. Πλακίδιο `alert` δεν υπάρχει στα αρχικά πλακίδια, αν όμως χρησιμοποιήσετε την εντολή στο περιβάλλον κώδικα και στη συνέχεια μεταβείτε στο περιβάλλον πλακιδίων θα διαπιστώσετε ότι έχει εμφανιστεί ένα πλακίδιο με όνομα `alert` στην περιοχή που έχουμε δημιουργήσει το πρόγραμμά μας.

Α) Αν θέλουμε η εμφάνιση να γίνει σε μια παράγραφο, τοποθετούμε το αλφαριθμητικό μέσα σε μονά ή διπλά εισαγωγικά. Π.χ. `write 'Είμαι μαθητής της Γ' Γυμνασίου. Είμαι 14 χρονών. Μου αρέσει το μπάσκετ. Μένω στην Ελλάδα. Αγαπημένο μου χρώμα είναι το κόκκινο'.`

Β) Αν θέλουμε να έχουμε περισσότερες παραγράφους (στην οθόνη), θα πρέπει εκεί που αρχίζει και τελειώνει κάθε παράγραφος να τοποθετούμε τις ενδείξεις και .

Υπάρχουν διαθέσιμες και άλλες ενδείξεις (τύπου HTML) για τη μορφοποίηση κειμένου όπως τα ... για πλάγια γραφή ή ... για έντονα γράμματα (bold).

III. Ενώνοντας αλφαριθμητικά

Αν θέλουμε να συνενώσουμε δύο ή περισσότερα αλφαριθμητικά, βάζουμε το σύμβολο + ανάμεσα σε αυτά. Π.χ. `write 'a' + '3'` θα δώσει a3.

Παραδείγματα

Εντολή `write`	αποτέλεσμα
`write 'καλημέρα'`	καλημέρα

`write '<p>Σήμερα είναι Δευτέρα. </p>`
`<p> Είναι Νοέμβριος.</p>`
 	
`<p>Πηγαίνω στη Γ γυμνασίου.</p>'`
	Σήμερα είναι Δευτέρα.
	
	Είναι Νοέμβριος.
	
	Πηγαίνω στη Γ Γυμνασίου.

`write '10+5'`	10 + 5
`write 10+5`	15
`write 2+3*5`	17
`write (2+3)*5`	25
`write 'a' + 3 + 5`	a35
`write 'a' + (3 + 5)`	a8
`write '12 + 8 =' + (12 + 8)`	12 + 8 =20
`write '12 + 8 =', 12 + 8`	12 + 8 = 20
`write "12 + 8 = #{12 + 8}"`	12 + 8 = 20
`write Number('5') + Number('3')`	8
`write 3 > 2`	true
`write '<i> Μου αρέσουν τα <u> φρούτα </u></i>'`	*Μου αρέσουν τα <u>φρούτα</u>*
`write "Σ'ευχαριστώ πολύ."`	Σ' ευχαριστώ πολύ.
`write 'Ο Λεωνίδας είπε: "Μολών λαβέ!"'`	Ο Λεωνίδας είπε: "Μολών λαβέ!"

Οι μεταβλητές κι η χρήση τους

Η μεταβλητή είναι ένα όνομα για μια τιμή που θέλουμε να αποθηκεύσουμε με σκοπό να τη χρησιμοποιήσουμε αργότερα. Οι μεταβλητές και η τιμή τους αποθηκεύονται στη μνήμη του υπολογιστή.

Δημιουργώ μεταβλητές – δίνω τιμές σε αυτές

Δημιουργούμε μια μεταβλητή και αποδίδουμε τιμή σε αυτήν με το =

Παραδείγματα

```
x=5
write x*x
```

```
onoma='Maria'
write onoma
say onoma
write 'onoma'
```

```
message = 'καλώς ήρθατε.'
write 'message'
see message
```

```
x = 20
box black, x * 5
box white, x * 4
box black, x * 3
box white, x * 2
```

```
x= 'Μαρία'
write 'Γεια σου '+ x
```

Στον προγραμματισμό η εκχώρηση `x = τιμή` αντιπροσωπεύει μια αποθήκευση στη μνήμη με όνομα x. Στα μαθηματικά το $y= x^2$ ερμηνεύεται ως ένας ορισμός.

Συνηθισμένες παρανοήσεις:

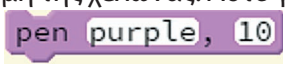
`x=5`
`y=4`
`x=y`
	Η τελευταία εντολή αποδίδει στο x την τιμή του y
	(στα μαθηματικά αυτό είναι λάθος)

`x=(random 6)` `write = x + x`	Ο μαθητής μπορεί να θεωρήσει ότι το x ορίζεται σε κάθε εμφάνισή του να είναι τυχαίος αριθμός, οπότε να περιμένει ότι θα εμφανιστεί ένα αποτέλεσμα όπως 4+5=9. Αυτό όμως δεν ισχύει μέσα στην ίδια εντολή
`write x + 6` `x = 5`	Η απόδοση τιμής σε μια μεταβλητή θα πρέπει να προηγείται από τη χρήση της μεταβλητής. Θα πρέπει να τεθεί πρώτα η εντολή `x = 5` και μετά η `write x + 6`

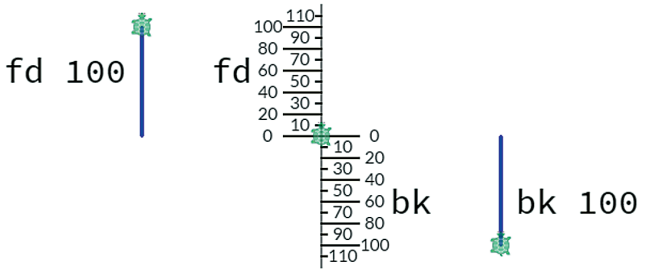
Βασικές εντολές σχεδίασης

Εντολές κίνησης - στροφής

Η μονάδα μέτρησης μετακίνησης (ευθύγραμμα τμήματα) είναι τα pixels (εικονοστοιχεία), ενώ η μονάδα μέτρησης των γωνιών είναι οι μοίρες. Για παράδειγμα η εντολή `fd 100` θα προκαλέσει τη μετακίνηση της χελώνας μπροστά κατά 100 pixels.

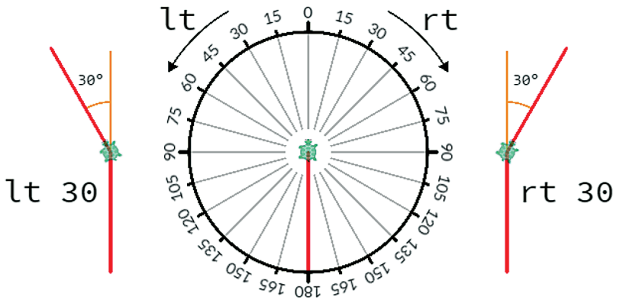
Σημείωση: Οι εντολές σχεδίασης μετακινούν τη χελώνα αλλά δεν αφήνουν ίχνος. Για να δημιουργηθεί ίχνος (γραμμή κ.λπ) θα πρέπει να έχει ενεργοποιηθεί η γραμμή της χελώνας. Αυτό γίνεται με την εντολή `pen, χρώμα,πάχος γραμμής` π.χ. `pen blue, 10` ή 

Η Εικόνα 15 δείχνει την έννοια της μετακίνησης που πραγματοποιείται με την εντολή `fd` (forward - μπροστά) πάνω στην οθόνη:



Εικόνα 15: Η έννοια της μετακίνησης της χελώνας μπροστά ή προς τα πίσω

Η αλλαγή στην κατεύθυνση της κίνησης της χελώνας πραγματοποιείται με τις εντολές `lt` (left turn - αριστερή στροφή) και `rt` (right turn - δεξιά στροφή). Η παρακάτω εικόνα δείχνει την έννοια της στροφής που πραγματοποιεί η χελώνα.

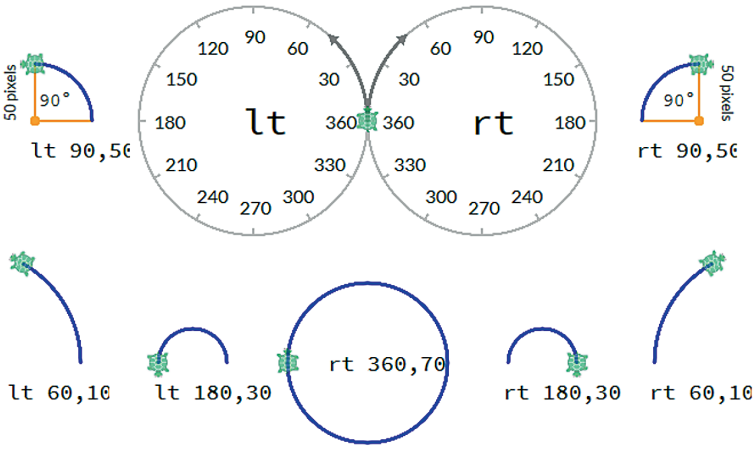


Εικόνα 16: Η έννοια της στροφής στην κίνηση της χελώνας προς τα δεξιά (rt) ή πρός τα αριστερά (lt)

Εντολές για δημιουργία τόξων - κύκλων

Παρατηρούμε ότι οι εντολές είναι ίδιες με τις εντολές αλλαγής διεύθυνσης αλλά στην περίπτωση αυτή συντάσσονται με δύο παραμέτρους. Η πρώτη παράμετρος αφορά το μήκος του τόξου σε μοίρες και η δεύτερη την ακτίνα του κύκλου που ανήκει το τόξο που θα δημιουργηθεί. Στη συγκεκριμένη περίπτωση θα δημιουργηθεί ένα ημικύκλιο (τόξο 180 μοιρών) με ακτίνα 100.

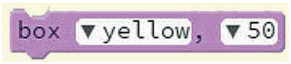
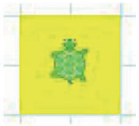
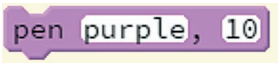
Η παρακάτω Εικόνα 17 διευκρινίζει τη δημιουργία αριστερόστροφων και δεξιόστροφων τόξων.



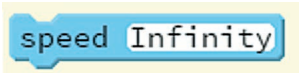
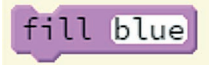
Εικόνα 17: Η έννοια της δημιουργίας τόξων και κύκλων

Συνοπτικά

`fd 100`	fd 100	Κίνηση μπροστά 100 εικονοστοιχεία
`rt 90`	rt 90	Δεξιά στροφή 90 μοίρες
`lt 90`	lt 90	Αριστερή στροφή 90 μοίρες
`bk 100`	bk 100	Κίνηση πίσω 100 εικονοστοιχεία
`home()`	home()	Επιστροφή της χελώνας στην αρχική της θέση
`rt 180, 100` `lt 180, 100`	rt 180, 100 lt 180, 100	Δημιουργία ενός τόξου 180 μοιρών με ακτίνα (κύκλου) 100 εικονοστοιχεία rt από αριστερά προς δεξιά lt από δεξιά προς αριστερά
`moveto 100, 50` `movexy 30, 20`	moveto 100, 50 movexy 30, 20	Η `moveto` σε μεταφέρει στις οριζόμενες συντεταγμένες `x, y` ενώ η `movexy` προκαλεί μετατόπιση `x, y` σε σχέση με το σημείο που βρίσκεσαι
`jumpto 100, 50` `jumpxy 30, 20`	jumpto 100, 50 jumpxy 30, 20	Όμοια με τη `moveto` και `movexy` αλλά δεν αφήνει ίχνος

`dot green, 50`		Σχεδιάζει ένα κύκλο με συγκεκριμένο χρώμα (πράσινο) και διάμετρο (50) στη θέση που βρίσκεται η χελώνα 
`box red, 50`		Σχεδιάζει ένα τετράγωνο με συγκεκριμένο χρώμα (κίτρινο) και πλευρά (50) στη θέση που βρίσκεται η χελώνα 
`pen purple, 10` `pen erase, 10`		Ορίζει το χρώμα και το μέγεθος του στυλό που θα χρησιμοποιήσει η χελώνα στις επόμενες κινήσεις της. Με το `erase` λειτουργεί ως γόμα.
`pu()` ή `pen up` ή `pen off`		Η χελώνα κατά τη μετακίνησή της δεν αφήνει ίχνος. Όμοια η εντολή `pen up`
`pd()` ή `pen on`		Επαναφέρει τη χελώνα στο να αφήνει ίχνος κατά τη μετακίνησή της

Βοηθητικές εντολές

	`speed 10`	Αύξηση ταχύτητας σχεδίασης σε 10 εντολές ανά δευτερόλεπτο
	`speed Infinity`	Άμεση εκτέλεση όλων των εντολών (βλέπουμε το τελικό αποτέλεσμα)
	`hide()`	Κρύβει τη χελώνα από την οθόνη
	`show()`	Εμφανίζει ξανά τη χελώνα
	`fill blue`	Γεμίζει την περιοχή με το αναφερόμενο χρώμα
	`cs()`	Καθαρίζει την περιοχή σχεδίασης.

Γεμίζοντας με χρώμα ένα σχήμα

Η εντολή που γεμίζει ένα σχήμα είναι η `fill`, αλλά θα πρέπει να κατανοηθεί η λειτουργία της.

Εντολές	**Αποτέλεσμα**

`pen purple, 5`
`fd 100`
`rt 90`
`rt 180, 50`
`fill pink`
	

Η εντολή `fill` ελέγχει τη διαδρομή που έκανε η χελώνα με την τρέχουσα πένα (pen) και γεμίζει τη διαδρομή αυτή με το χρώμα που καθορίζεται.

Η εντολή `fill` γεμίζει μια περιοχή που καθορίζεται από μια συνεχόμενη γραμμή, ακόμα και όταν δεν είναι κλειστή. Το γέμισμα αρχίζει από την αρχή της σχεδίασης μέχρι το τελευταίο σημείο. Για παράδειγμα:

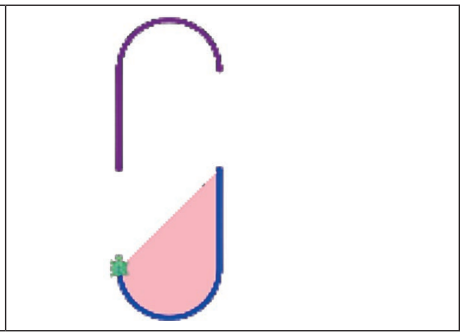
```
pen purple, 5
fd 100
rt 180, 50
fill pink
```

Αν στην ακολουθία των εντολών χρησιμοποιηθούν οι εντολές `jump`, `pen up`, ή `pen down`, θεωρείται ότι εξακολουθεί να είναι η ίδια διαδρομή και το γέμισμα εφαρμόζεται στα διαφορετικά τμήματα.

```
pen purple, 5
fd 100
rt 180, 50
jumpto 100, 0
fd 100
rt 180, 50
fill pink
```

Προσοχή! Αν στην ακολουθία των εντολών αλλάξουμε χρώμα πένας ή χρησιμοποιήσουμε την εντολή `pen null`, τότε διακόπτεται η πρώτη διαδρομή και ξεκινά μια νέα διαδρομή. Τώρα το γέμισμα δεν θα αφορά όλο το σχήμα συνολικά αλλά κάθε τμήμα του ξεχωριστά. Αν τοποθετηθεί μια εντολή fill στο δεύτερο μέρος, αυτή ισχύει για τη νέα διαδρομή.

```
pen purple, 5
fd 100
rt 180, 50
jumpto 100, 0
pen blue, 5
fd 100
rt 180, 50
fill pink
```



Αν θέλαμε να γεμίσουμε και το πρώτο τμήμα του σχεδίου, θα έπρεπε να προσθέσουμε μια εντολή `fill` αμέσως μετά τον σχεδιασμό του.

Δομή Επανάληψης: Η εντολή for

Η επανάληψη επιτρέπει σε ένα πρόγραμμα να εκτελέσει ένα πλήθος εντολών πολλές φορές χωρίς να χρειάζεται να αναφέρει αναλυτικά τις εντολές για κάθε επανάληψη.

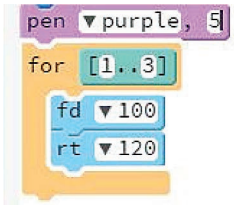
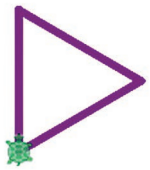
Η επανάληψη στο Pencil Code γίνεται με τρεις τρόπους από τους οποίους θα παρουσιαστεί εδώ μόνο ο ένας, ο οποίος πραγματοποιείται με τη δομή `for`.

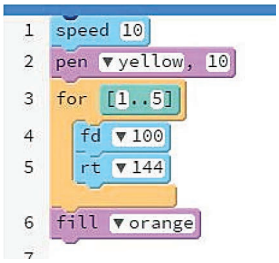
Η δομή `for` επαναλαμβάνει μια ομάδα εντολών για ένα σταθερό αριθμό επαναλήψεων. Ο αριθμός των επαναλήψεων δηλώνεται με μια λίστα.

Το πλακίδιο `for` βρίσκεται στην ομάδα πλακιδίων **Control**. Αν χρησιμοποιηθεί το περιβάλλον κώδικα, οι εντολές που βρίσκονται μέσα στη `for` (και θα επαναληφθούν) πρέπει να γραφούν με μια εσοχή δύο κενών στην αρχή κάθε γραμμής.

Παραδείγματα

`for [1..3]` `  write 'Hello'`		Hello Hello Hello

`pen purple, 5`
`for [1..3]`
`  fd 100`
`  rt 120`
		

`speed 10`
`pen yellow, 10`
`for [1..5]`
`  fd 100`
`  rt 144`
`fill orange`
		

`wear 'car'`
`grow 0.2`
`pu()`
`for [1..10]`
`  movexy -5, 0`
		Εμφανίζει την εικόνα ενός αυτοκινήτου και δημιουργεί την αίσθηση της κίνησης

Υποπρογράμματα–συναρτήσεις

Μια συνάρτηση είναι ένα ανεξάρτητο τμήμα κώδικα που αποτελεί ένα υποπρόγραμμα, δηλαδή μπορεί να αποτελέσει τμήμα ενός άλλου προγράμματος. Οι συναρτήσεις χρησιμοποιούνται από τους προγραμματιστές για να διαιρέσουν μεγάλα προγράμματα σε ένα πλήθος από μικρότερα. Ορισμένα από τα πλεονεκτήματα των συναρτήσεων είναι:

- επιτρέπουν την επαναχρησιμοποίηση του κώδικα. Αν δημιουργήσουμε μια συνάρτηση, ο κώδικάς της μπορεί να χρησιμοποιηθεί σε πολλά σημεία του προγράμματος χωρίς να χρειάζεται να ξαναγράψουμε τις εντολές.
- επιτρέπουν τον έλεγχο της εκτέλεσης του κώδικα. Η διάσπαση ενός μεγάλου μήκους κώδικα σε μικρότερα επιτρέπει τον καλύτερο έλεγχο των εντολών.

Όπως και στη δομή επανάληψης for οι εντολές που βρίσκονται στο σώμα της συνάρτησης γράφονται με μια αρχική εσοχή δύο κενών

Προσοχή: Οι εντολές που βρίσκονται σε μια συνάρτηση δεν εκτελούνται αμέσως με την ολοκλήρωση της συνάρτησης αλλά αργότερα, όταν κληθεί η συνάρτηση αυτή.

Ενσωματωμένες συναρτήσεις

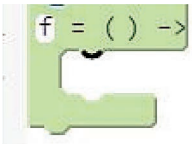
Η CoffeeScript περιέχει αρκετές ενσωματωμένες συναρτήσεις, όπως για παράδειγμα οι παρακάτω μαθηματικές συναρτήσεις.

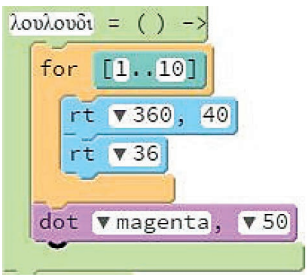
`write round 6.7`	6
`write abs -7`	7
`write max 10, 3`	10

Δημιουργία συνάρτησης (χωρίς μεταβλητή)

Στην ομάδα πλακιδίων Operators υπάρχει το αντίστοιχο πλακίδιο ορισμού μιας συνάρτησης.

Σε περιβάλλον κώδικα, μια συνάρτηση ορίζεται από το όνομα της συνάρτησης, τον χαρακτήρα "=", δύο παρενθέσεις "()"μέσα στις οποίες τοποθετούνται οι μεταβλητές της συνάρτησης και τέλος οι χαρακτήρες "-->". Μετά το "βέλος" ακολουθούν οι εντολές που περιλαμβάνει η συνάρτηση, αν οι εντολές είναι περισσότερες από μια, τότε γράφονται σε νέα γραμμή με εσοχή δύο χαρακτήρων στην αρχή της ή ένα tab.

	Το πλακίδιο ορισμού συνάρτησης

	`λουλουδι = () ->`
	`  for [1..10]`
	`  rt 360, 40`
	`  rt 36`
	`  dot magenta, 50`

Οι παρενθέσεις, μπορούν να παραλειφθούν, αν δώσουμε τις εντολές με κώδικα.

Για να καλέσουμε τη συνάρτηση υπάρχει αντίστοιχο πλακίδιο στην ομάδα πλακιδίων **Operators**

	Πλακίδιο κλήση της συνάρτησης f
	`λουλουδι()`
Δεν υπάρχει αντίστοιχο πλακίδιο	`(ή)do λουλουδι`

Συναρτήσεις με μεταβλητές (χρήση παραμέτρων)

Οι μεταβλητές που χρησιμοποιούμε στον ορισμό μιας συνάρτησης ονομάζονται παράμετροι, ενώ, όταν καλούμε τη συνάρτηση με μια τιμή ή μια μεταβλητή, αυτή ονομάζεται όρισμα.

Παράδειγμα 1: Το x στη συνάρτηση tetragono είναι παράμετρος, ενώ το 100 στην κλήση tetragono(100) είναι όρισμα.

```
tetragono=(x)->
  for [1..4]
    fd x
    rt 90
```

την καλούμε ως εξής

```
tetragono(100)
     ή
tetragono 100
     ή
x=100
tetragono(x)
```

Παράδειγμα 2: Συνάρτηση με τρεις παραμέτρους.

```
τρίγωνο = (color, n, x) ->
  pencolor, n
  for [1..3]
    fd x
    rt 120
```

την καλούμε ως εξής

```
τρίγωνο(red, 4, 100)
     ή
τρίγωνο red, 4, 100
     ή
color=red
n=4
x=100
τρίγωνο(color, n, x)
```

Αν έχουμε γράψει και τις 2 παρακάτω συναρτήσεις με τη σειρά που φαίνονται:

```
τρίγωνο= ->
  for [1..3]
    fd 400
    rt 120
```

```
τρίγωνο = (color, n, x) ->
  pen color, n
  for [1..n]
    fd x
    rt 360/n
```

Αν καλέσουμε τη διαδικασία

```
τρίγωνο()
```

προφανώς θα καλέσει τη δεύτερη συνάρτηση, ο ορισμός της οποίας αντικατέστησε την πρώτη, και στις μεταβλητές θα χρησιμοποιήσει τις default τιμές. Για το color η τιμή αυτή είναι black, για τις άλλες μεταβλητές η τιμή τους θα είναι άγνωστη (πρακτικά μηδέν).

Τοπικές μεταβλητές και γενικές μεταβλητές

Μέσα σε ένα πρόγραμμα οι μεταβλητές που ορίζουμε και η τιμή τους ισχύουν για όλο το πρόγραμμα. Δηλαδή είναι γενικές. Αυτό όμως δεν ισχύει, όταν το πρόγραμμα περιέχει συναρτήσεις. Τότε οι μεταβλητές που ορίζονται μέσα σε συναρτήσεις είναι τοπικές, δηλαδή ισχύουν μόνο μέσα στη συνάρτηση, ενώ δεν αναγνωρίζονται έξω από αυτήν. Οι γενικές μεταβλητές ισχύουν και μέσα στις συναρτήσεις και η τιμή τους μπορεί να τροποποιηθεί μέσα σε αυτές.

Παραδείγματα:

```
x = 5
f = () ->
  x = 10
  write x
f()
write x
```

βγάζει `10 10` η μεταβλητή x αναγνωρίστηκε και τροποποιήθηκε μέσα στη συνάρτηση

```
x = 5
f = (y) ->
  y = 10
  write y
f(x)
write x
```

βγάζει (φυσιολογικά) 10 5

Αν προστεθεί στο τέλος

```
x = 5
f = () ->
  y = 10
  write x
f()
write x
write y
```

τότε θα εμφανίσει 5 5 (Τα δύο `write x`) αλλά θα βγάλει μήνυμα ότι η μεταβλητή `y` δεν είναι ορισμένη

(ορίστηκε μέσα στη συνάρτηση)

Το ίδιο ισχύει, αν χρησιμοποιηθεί η ίδια μεταβλητή εσωτερικά και εξωτερικά της συνάρτησης

```
x = 5
f = (x) ->
  x = 10
write x
f(x)
write x
```

Θα εμφανίσει 5 10

Είσοδος δεδομένων

Οι εντολές εισόδου δεδομένων στο περιβάλλον του Pencil Code υλοποιούνται με συναρτήσεις. Υπάρχουν δύο βασικές εντολές εισόδου δεδομένων η `read` και η `readnum`.

Η `read` διαβάζει ένα αλφαριθμητικό από το πληκτρολόγιο και η σύνταξη της είναι
`read 'Κείμενο επεξήγησης', (Μεταβλητή) ->` έξοδος συνάρτησης
Το κείμενο που θα εισαγάγουμε θα αποθηκευτεί στη `Μεταβλητή`
Π.χ.

```
read 'Πως σε λένε;', (x) ->
  write 'γεια σου' + x
```

Η `readnum` διαβάζει ένα αριθμό από το πληκτρολόγιο και η σύνταξή της είναι
`readnum 'Κείμενο επεξήγησης', (Μεταβλητή) ->` έξοδος συνάρτησης
Ο αριθμός που θα εισαγάγουμε θα αποθηκευτεί στη μεταβλητή
Π.χ.

```
readnum 'Your age?', (n) ->
  write 'Next year you will be ' + (n + 1)
```

Τα αντίστοιχα πλακίδια των εντολών `read`και `readnum` βρίσκονται στην ομάδα εντολών **Text**

Εκτός από κείμενο και αριθμούς υπάρχει η δυνατότητα εισαγωγής δεδομένων από το μικρόφωνο (ηχητικά).

Η αντίστοιχη εντολή είναι η `listen` με σύνταξη
`listen 'Κείμενο', (Μεταβλητή) ->` έξοδος συνάρτησης
Η πρόταση που θα πούμε θα αποθηκευτεί στη μεταβλητή
Π.χ.

```
listen 'Say something', (t) ->
  say 'Yousaid: ' + t
```

Εκτός από τις εντολές αυτές υπάρχουν διαθέσιμες εντολές (συναρτήσεις) οι οποίες ενεργοποιούνται μόλις γίνει μια συγκεκριμένη ενέργεια είτε από το ποντίκι είτε από το πληκτρολόγιο.

Μια τέτοια εντολή είναι η `click` η οποία ενεργοποιείται μόλις πατηθεί το πλήκτρο του ποντικιού. Η σύνταξή της είναι

```
click () -> εντολές
```

Π.χ.

```
click () ->
  say 'hello'
```

Αντίστοιχη εντολή που ενεργοποιείται από το πληκτρολόγιο είναι η `keydown`, η οποία ελέγχει αν έχει πατηθεί κάποιο πλήκτρο. Η σύνταξή της είναι

```
keydown (Πλήκτρο) -> εντολές
```

Π.χ.

```
keydown 'A', ->
  tone 'C'
```

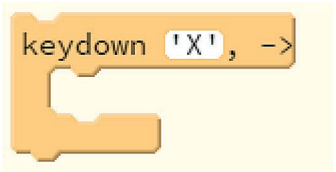
Τέλος μπορούμε να δημιουργήσουμε ένα κουμπί στην οθόνη το πάτημα του οποίου θα εκτελεί μια εργασία. Η εντολή δημιουργίας του κουμπιού είναι η `button`. Η σύνταξή της είναι

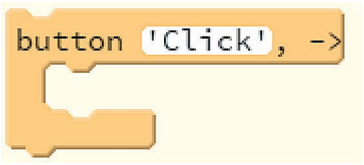
```
button (Ταμπέλα κουμπιού) -> εντολές
```

Π.χ.

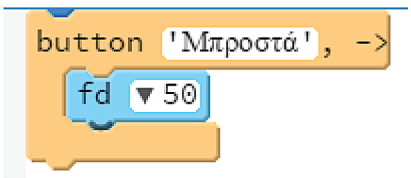
```
button 'good morning', ->
  say 'good morning'
```

Τα αντίστοιχα πλακίδια των εντολών `click`, `keydown` και `button` βρίσκονται στην ομάδα εντολών Control



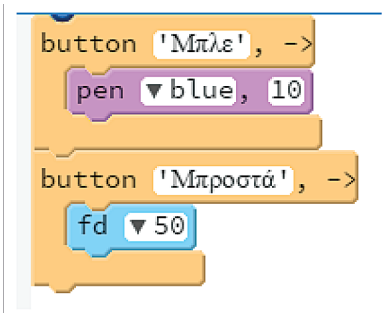


Παράδειγμα δημιουργίας μιας συνάρτησης που θα μετακινεί τη χελώνα 50 Pixels μπροστά με το πάτημα ενός κουμπιού.

Με πλακίδια	**Με κώδικα**
	`button 'Μπροστά', ->` `  fd 50`

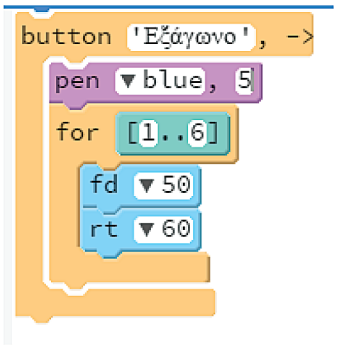
Μπορούμε να δημιουργήσουμε περισσότερα κουμπιά με συναρτήσεις για να ελέγχουμε τη χελώνα. Για παράδειγμα δημιουργία δύο κουμπιών που το ένα βάζει χρώμα στην πένα και το άλλο μετακινεί τη χελώνα.

Με πλακίδια	Με εντολές	

	`button 'Μπλε', ->`
	`  pen blue, 10`
	`button 'Μπροστά', ->`
	`  fd 50`
		θα δημιουργηθούν τα κουμπιά
		 

Μπορούμε να δημιουργήσουμε κουμπιά που να σχεδιάζουν οποιοδήποτε σχήμα. Για παράδειγμα ένα εξάγωνο.

Με πλακίδια	Με εντολές

	`button 'Εξάγωνο', ->`
	`  pen blue, 5`
	`  for [1..6]`
	`  fd 50`
	`  rt 60`

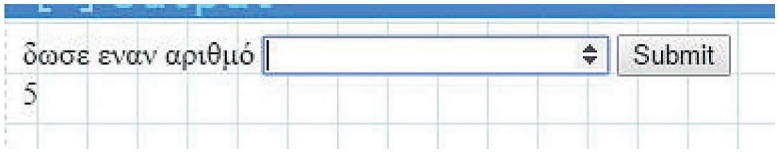
Δυσκολίες στη χρήση συναρτήσεων

```
x=5
readnum 'δωσε εναν αριθμό', (x) ->
  write x
write x
```

Η τελευταία εντολή `write` θα εμφανίσει το αποτέλεσμα 5 πριν από την `readnum`

Οι συναρτήσεις εκτελούνται παράλληλα με τα προγράμματα ακόμα και αν είναι ενσωματωμένες σε αυτά.

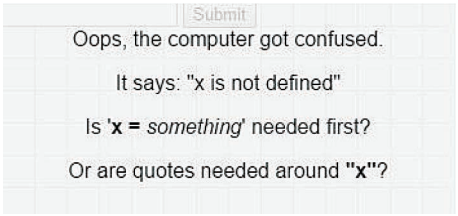
Έτσι εφόσον της έχουμε δώσει μια τιμή της μεταβλητής x προηγουμένως, η τελευταία `write` θα την εμφανίσει.



Έστω ότι του δίνουμε τον αριθμό 4. Τότε θα προκύψει το ακόλουθο αποτέλεσμα.

Αν δεν έχουμε δώσει εξ' αρχής τιμή για τη μεταβλητή x, θα εμφανιστεί μήνυμα λάθους.

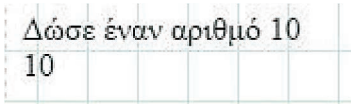
```
readnum 'δωσε εναν αριθμό', (x) ->
  write x
write x
```



Αν θέλουμε η ερώτηση να περιμένει μέχρι να εισάγει απάντηση ο χρήστης πριν προχωρήσει στην εκτέλεση της επόμενης εντολής, τότε χρησιμοποιούμε την παρακάτω σύνταξη:

```
x=5
await readnum ‹Δώσε έναν αριθμό›, defer x
write x
```

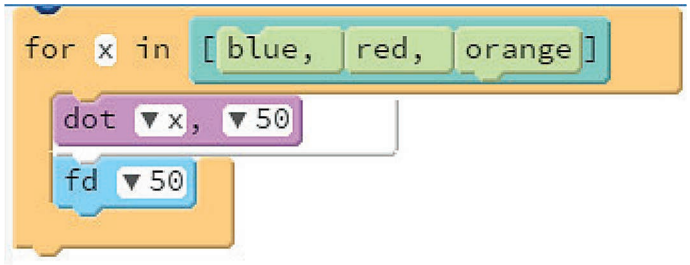
Έστω ότι του δίνουμε τον αριθμό 10. Τότε θα προκύψει το ακόλουθο αποτέλεσμα.

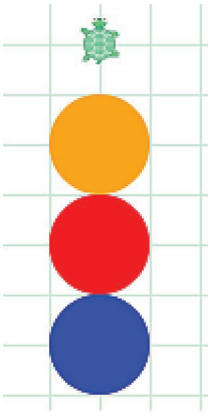


Εισαγωγή στις λίστες

Οι λίστες είναι μια δομή δεδομένων που υποστηρίζονται από όλες τις σύγχρονες γλώσσες προγραμματισμού. Οι δομές δεδομένων είναι αντικείμενα που χρησιμοποιεί η γλώσσα προγραμματισμού (και ο προγραμματιστής) για την ταυτόχρονη οργάνωση πολλών στοιχείων. Οι λίστες μπορούν να χρησιμοποιηθούν για την αναπαράσταση μονοδιάστατων πινάκων. Ένα παράδειγμα λίστας και της χρήσης της υπάρχει στο παρακάτω πρόγραμμα (το οποίο μπορείτε να γράψετε μόνο σε περιβάλλον κειμένου):

```
for x in [blue, red, orange]
  dot x, 50
  fd 50
```





προκαλεί την εμφάνιση του σχήματος που βρίσκεται δίπλα.

Το αντικείμενο `[blue, red, orange]` είναι μια δομή δεδομένων που ονομάζεται λίστα. Μια λίστα ορίζεται με ένα ζευγάρι από ορθογώνιες αγκύλες `[]` οι οποίες περιέχουν ορισμένα στοιχεία που χωρίζονται με κόμμα. Τα στοιχεία αυτά μπορεί να είναι αριθμοί, λέξεις ή γενικότερα αλφαριθμητικά. Στο συγκεκριμένο παράδειγμα η μεταβλητή x του `for` σε κάθε επανάληψη παίρνει διαδοχικά τιμή από τα περιεχόμενα της λίστας. Άρα θα γίνουν τρεις επαναλήψεις και το x θα έχει σε κάθε μια από αυτές την τιμή ενός χρώματος.

Μια λίστα μπορεί να έχει ένα συγκεκριμένο όνομα. Για παράδειγμα

```
colors =[blue, red, orange]
```

Στην περίπτωση αυτή έχουμε αποδώσει στη μεταβλητή colors τη λίστα `[blue, red, orange]`.

Η παραπάνω λίστα θα μπορούσε να γραφεί χωρίς να διαχωρίζουμε τα στοιχεία με κόμμα, αν τα διαχωρίζαμε γράφοντας το καθένα σε νέα γραμμή, Για παράδειγμα:

```
colors =[
 blue
 red
 orange
 ]
```

Κάθε λίστα έχει ένα συγκεκριμένο αριθμό στοιχείων με καθορισμένη σειρά. Η αρίθμηση της θέσης (index) των στοιχείων αρχίζει από το 0. Δηλαδή στη λίστα colors το στοιχείο με θέση 0 είναι το blue, το στοιχείο με θέση 1 είναι το red και το στοιχείο με θέση 2 είναι το orange. Μπορούμε να αναφερόμαστε σε ένα συγκεκριμένο στοιχείο της λίστας χρησιμοποιώντας τη θέση του. Για παράδειγμα η έκφραση colors[0] αναφέρεται στο πρώτο στοιχείο της λίστας κ.λπ. Συνεπώς αν για τη λίστα colors χρησιμοποιήσουμε την εντολή

`write colors[1]` θα πάρουμε ως αποτέλεσμα το red

Ο αριθμός θέσης ενός στοιχείου μπορεί να χρησιμοποιηθεί για την αλλαγή των περιεχομένων μιας λίστας. Για παράδειγμα η εντολή

`color[1] = yellow` θα αλλάξει το δεύτερο στοιχείο της λίστας colors από red σε `yellow`.

Με τον όρο μήκος της λίστας αναφερόμαστε στον αριθμό των στοιχείων που περιέχει. Μπορούμε να αναφερθούμε στο μήκος της λίστας με την έκφραση colors.length.

Συνεπώς αν γράψουμε

`write colors.length` θα πάρουμε τον αριθμό 3.

Το ίδιο αποτέλεσμα θα πάρουμε με τις εντολές

```
mikos = colors.length
write mikos
```

ΠΑΡΑΤΗΡΗΣΗ: Ίσως φανεί περίεργο που το πρώτο στοιχείο της λίστας είναι στη θέση 0 και όχι στη θέση 1. Αυτό είναι ένα χαρακτηριστικό που το έχουν πολλές γλώσσες προγραμματισμού όπως π.χ η Pyhton. Μια αιτιολογία είναι ότι ο αριθμός θέσης εκφράζει την απόσταση του στοιχείου από την αρχή της λίστας, έτσι το δεύτερο στοιχείο είναι στην πρώτη θέση άρα η απόστασή του από την αρχή είναι 1. Το τελευταίο στοιχείο μια λίστας είναι στη θέση length - 1.

Όταν μία λίστα περιέχει ακέραιους συνεχόμενους αριθμούς, τότε μπορεί να γραφεί με πιο σύντομο τρόπο. Για παράδειγμα η λίστα {1, 2, 3, 4, 5] μπορεί να γραφεί [1...6]. Παρατηρήστε ότι η λίστα [1...6] δεν περιλαμβάνει το τελευταίο στοιχείο. Μαθηματικά μια τέτοια λίστα μπορεί να περιγραφεί ως ένα ημίκλειστο διάστημα τιμών (κλειστό από τα αριστερά και ανοιχτό από τα δεξιά).

Μια λίστα μπορεί να είναι κενή - δηλαδή να μην περιέχει κανένα στοιχείο, φυσικά μια κενή λίστα έχει μήκος (length) ίσο με μηδέν. Αυτό μπορούμε να το διαπιστώσουμε με το πρόγραμμα.

```
Lista = []
```

```
write 'length of Lista is' + Lista.length
```

Επανάληψη εννοιών

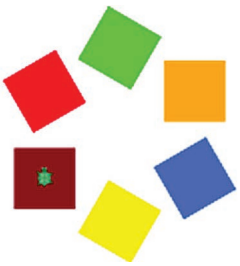
Εντολές	**Αποτέλεσμα**
`onomata = ['Μαρία', ' Ελένη', 'Γιώργος']`	(Ορισμός μιας λίστας)
`write onomata`	Μαρία, Ελένη, Γιώργος
`write onomata.length`	3 (υπάρχουν 3 στοιχεία στη λίστα).
`write onomata[0]`	Μαρία (Το 0 αντιστοιχεί στην πρώτη θέση)
`write onomata[1]`	Ελένη (Το 1 αντιστοιχεί στη δεύτερη θέση)
`write onomata[2]`	Γιώργος
`onomata[0] = 'Σοφία'`	(Απόδοση τιμής στο πρώτο στοιχείο της λίστας)
`write onomata`	Σοφία, Ελένη, Γιώργος

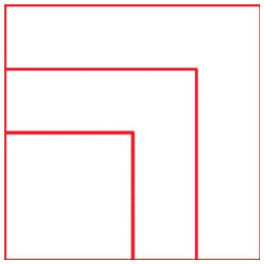
Παραδείγματα επαναλήψεων με τιμές μεταβλητής από λίστα

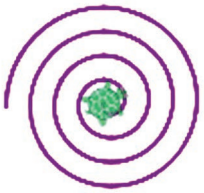
`for n in [1..5]` `  write 3 * n`	3 6 9 12 15
`for x in [10, 43, 6, 27]` `  write 'ο αριθμός', x`	ο αριθμός 10 ο αριθμός 43 ο αριθμός 6 ο αριθμός 27
`for x in ['Ελένη', 'Γιάννη', 'Μαρία']` `  write 'Γεια σου', x`	Γεια σου Ελένη Γεια σου Γιάννη Γεια σου Μαρία

`for color in [red, orange, yellow, blue]`
`  pen color, 4`
`  fd 100`
`  rt 90`
	

`for d in [50, 100, 50, 100]`
`  fd d`
`  rt 90`
	

`mycolors = [darkred, red, lime, orange, royalblue, yellow]`
`for x in mycolors`
`  box x, 70`
`  fd 100`
`  rt 60`
	

`tetragonox = (x)->`
`  for [1..4]`
`  fd x`
`  rt 90`
 	
`pen red`
`for x in [ 100, 150, 200]`
`  tetragonox(x)`
	

`pen purple`
`for x in [50..1] by -1`
`  rt 30, x`
	

Δομή επιλογής

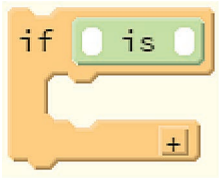
Εισαγωγή

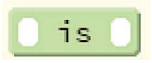
Στον προγραμματισμό, όπως και στη ζωή, συχνά παρουσιάζεται το φαινόμενο να αλλάξουμε τις ενέργειες που πρέπει να κάνουμε ανάλογα, αν ισχύει κάποια συνθήκη ή δεν ισχύει. Για παράδειγμα θέλουμε να επισκεφτούμε ένα φίλο μας αλλά δεν γνωρίζουμε αν είναι στο σπίτι του. Τότε η σκέψη μας είναι «Θα τηλεφωνήσω και, αν τον βρω, θα πάω να τον επισκεφτώ». Αν διαπιστώσουμε λοιπόν

ότι βρίσκεται στο σπίτι του θα κάνουμε κάποιες συγκεκριμένες ενέργειες οι οποίες δεν θα γίνουν, αν ο φίλος δεν βρεθεί. Έτσι και στον προγραμματισμό είναι συχνό το φαινόμενο να θέλουμε ένα τμήμα του προγράμματος με συγκεκριμένες εντολές να εκτελείται ή να μην εκτελείται ανάλογα με το αν ισχύει κάποια συνθήκη.

Η διαδικασία αυτή στο Pencil Code πραγματοποιείται με τη δομή επιλογής.

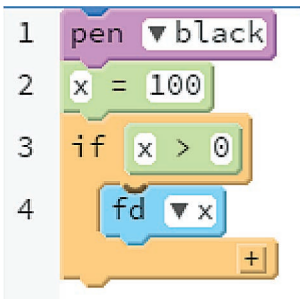
Η απλή δομή επιλογής έχει στα εικονίδια τη μορφή:



Η συνθήκη  μπορεί να αλλάξει και να γίνει  ή  . Τα εικονίδια αυτά υπάρχουν στην ομάδα **Operators**.

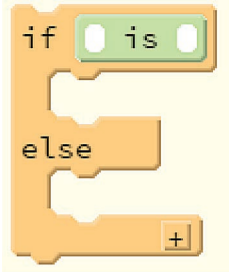
Παράδειγμα: Αν ο αριθμός x είναι θετικός, προχώρα μπροστά κατά x βήματα

Στα Εικονίδια	**Στον Κώδικα**

	`pen black`
	`x = 100`
	`if x > 0`
	`  fd x`

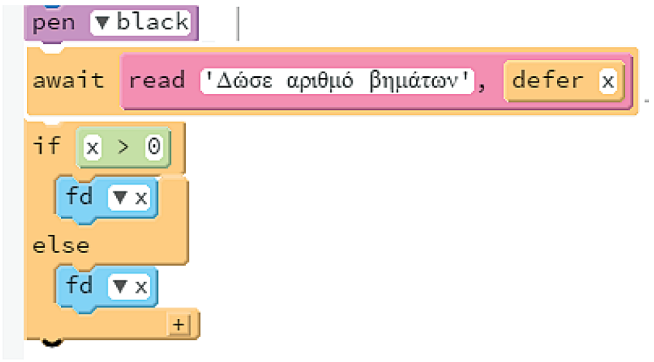
Η παραπάνω δομή ελέγχου ονομάζεται απλή, γιατί υπάρχουν πιο σύνθετες μορφές της. Για παράδειγμα μπορούμε να καθορίσουμε από την αρχή τι θα γίνει, αν δεν ισχύει η συνθήκη που βάλαμε. Στο παράδειγμα με το τηλεφώνημα στον φίλο μας, η σκέψη μας θα μπορούσε να είναι «Θα τηλεφωνήσω στο φίλο μου και, αν τον βρω, θα πάω να τον επισκεφτώ αλλιώς (αν δεν τον βρω) θα παίξω στον υπολογιστή».

Η δομή αυτή έχει τη μορφή



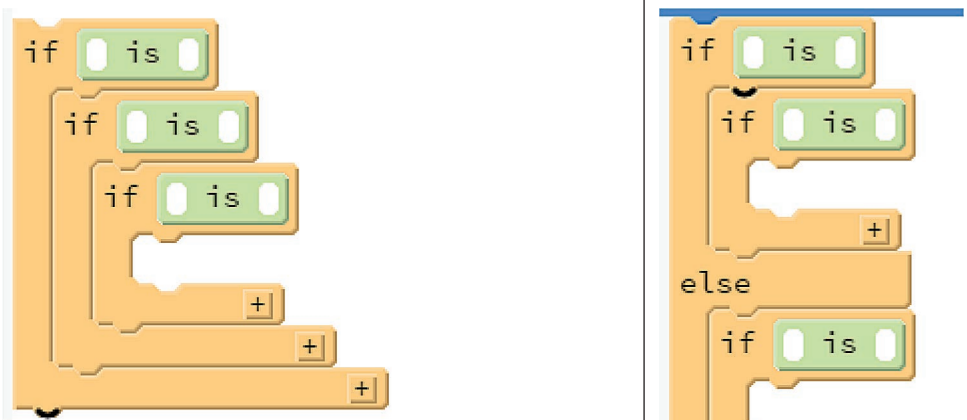
Παράδειγμα: Δώσε ένα αριθμό. Αν ο αριθμός x είναι θετικός, προχώρα μπροστά κατά x βήματα, αλλιώς (αν δεν είναι θετικός) πήγαινε προς τα πίσω κατά x βήματα

Στα Εικονίδια	Στον Κώδικα

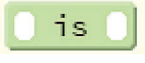
	`pen black`
	`await read 'Δώσε αριθμό βημάτων', defer x`
	`if x > 0`
	`  fd x`
	`else`
	`  fd x`

Παρατήρηση: Μετά το else παρατηρούμε ότι πάλι υπάρχει η εντολή fd (μπροστά) αλλά επειδή ο αριθμός είναι αρνητικός η χελώνα θα κινηθεί προς τα πίσω.

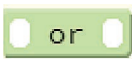
Η δομή επιλογής επιτρέπει την "εμφώλευση" δηλαδή την τοποθέτηση μιας δομής επιλογής μέσα σε μία άλλη. Για παράδειγμα



Αντίστοιχα σε μορφή κώδικα η εμφώλευση γίνεται με τις κατάλληλες εσοχές.

Η συνθήκη που βάζουμε μετά το `if` μπορεί να είναι απλή όπως  π.χ `x is 5`, ή σύνθετη. Μια σύνθετη συνθήκη υλοποιείται με τη βοήθεια λογικών συνδέσμων.

Λογικοί σύνδεσμοι

Στο Pencil Code χρησιμοποιούνται οι λογικοί σύνδεσμοι  ,  και  (Σε κώδικα αντίστοιχα and, or και not). Σε κάθε κενό ενός λογικού συνδέσμου μπορούμε να τοποθετήσουμε μια απλή συνθήκη π.χ.  . Σε κώδικα σύνθετες συνθήκες μπορεί να έχουν τη μορφή `x > 0 and y > 0`, `x is 1 or y is 1`.

Υπενθυμίζεται ότι η αλήθεια μιας λογικής συνθήκης που περιέχει ένα λογικό σύνδεσμο υπολογίζεται από τον παρακάτω πίνακα (όπου Α και Β συνθήκες π.χ `x>0`).

A	B	not A	not B	A and B	A or B
Αληθής	Αληθής	Ψευδής	Ψευδής	Αληθής	Αληθής
Αληθής	Ψευδής	Ψευδής	Αληθής	Ψευδής	Αληθής
Ψευδής	Αληθής	Αληθής	Ψευδής	Ψευδής	Αληθής
Ψευδής	Ψευδής	Αληθής	Αληθής	Ψευδής	Ψευδής

Ο πίνακας δίνεται για ενημέρωση του εκπαιδευτικού και δεν πρέπει να διδαχθεί στους μαθητές. Στους μαθητές αρκεί να ειπωθεί ότι το `not` αντιστρέφει τη συνθήκη, το and ισχύει, όταν και οι δύο συνθήκες είναι αληθινές και το or ισχύει πάντα εκτός αν και οι δύο συνθήκες είναι ψευδείς. Μπορούν να χρησιμοποιηθούν παραδείγματα από την πραγματική ζωή π.χ Αύριο θα πάω κινηματογράφο και θα διαβάσω Ιστορία (αντίστοιχα θα πάω κινηματογράφο ή θα διαβάσω Ιστορία) πότε θα έχω πει ψέματα και πότε αλήθεια;

Δομή επανάληψης `forever`

Η δομή επανάληψης `forever` χρησιμοποιείται, όταν θέλουμε κάποια επανάληψη να εκτελείται συνεχώς (περίπου 30 επαναλήψεις το δευτερόλεπτο). Για παράδειγμα μια δομή επανάληψης που ρυθμίζει ένα παιχνίδι. Η δομή `while` δεν είναι κατάλληλη για ατέρμονες επαναλήψεις, γιατί η χρήση της θα προκαλέσει το "κρέμασμα" του φυλλομετρητή, επειδή οι διαδοχικές επαναλήψεις γίνονται άμεσα η μία μετά την άλλη. Σε αντίθεση η `forever` κάνει μια μικρή διακοπή ανάμεσα στις επαναλήψεις και επιτρέπει την αλληλεπίδραση με τον χρήστη.

```
forever ->
  pen random color
  fd 3
  rt 2
```

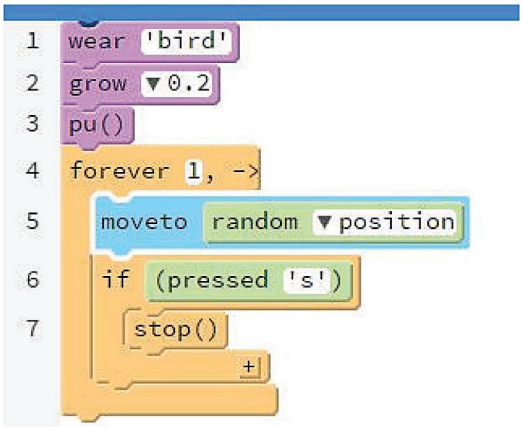
Η δομή `forever` βρίσκεται στην ομάδα πλακιδίων **Control** με τη μορφή

Ο αριθμός που βρίσκεται μετά το forever καθορίζει το κενό χρόνου που θα υπάρχει μεταξύ των επαναλήψεων. Συγκεκριμένα ο αριθμός εκφράζει πόσες επαναλήψεις θα εκτελεστούν σε ένα δευτερόλεπτο.

Διακοπή - τερματισμός της forever

Ένας ατέρμων βρόχος μπορεί να διακοπεί (και να τερματιστεί) με την εντολή `stop()`. Η εντολή αυτή θα πρέπει να τοποθετηθεί είτε σε μια δομή επιλογής `if`, είτε σε μια δομή ελέγχου του πληκτρολογίου όπως η `keydown`.

Παράδειγμα χρήσης




```
wear 'bird'

grow 0.2

pu()

forever 1, ->

  moveto random position

  if (pressed 's')

    stop()
```

Στο παραπάνω παράδειγμα η εκτέλεση του προγράμματος σταματά μόλις πατηθεί το γράμμα "s" (Η επιλογή του γράμματος είναι τυχαία από το "stop")

Μέρος Δεύτερο: Σχέδια μαθημάτων - Φύλλα εργασίας

Οι χρόνοι για την υλοποίηση των φύλλων εργασίας είναι ενδεικτικοί. Προφανώς έχει σημασία η πρότερη εμπειρία των μαθητών με τον Προγραμματισμό. Προτιμήσαμε να δώσουμε περισσότερες δραστηριότητες ανά ώρα από όσες μπορεί να κάνει ένας αρχάριος προγραμματιστής μέσα σε μια διδακτική ώρα ώστε να είναι διαθέσιμες για τους μαθητές που έχουν ευκολία στον προγραμματισμό, οι οποίοι θα ολοκληρώνουν πιο γρήγορα τις πρώτες δραστηριότητες και θέλουν να κάνουν κάτι ακόμη. Η τελευταία δραστηριότητα μπορεί σε κάθε περίπτωση να μπαίνει ως «υλικό για σκέψη» μέχρι το επόμενο μάθημα. Εξάλλου όσοι μαθητές θέλουν και έχουν πρόσβαση στο διαδίκτυο από το σπίτι μπορούν να τις υλοποιούν εκεί.

Προτείνουμε στους μαθητές να αποθηκεύουν τις δραστηριότητες των φύλλων εργασίας σε ξεχωριστό αρχείο κάθε φορά, ανά διδακτική ώρα.

Επισημαίνουμε στους μαθητές ότι μπορούμε να κάνουμε ανενεργή κάθε εντολή, αν προσθέσουμε στην αρχή της το σύμβολο # ή πολλές εντολές μαζί αρκεί να μπουν ανάμεσα σε δύο ###. Έτσι δε χρειάζεται να σβήσουν τη μια δραστηριότητα πριν κάνουν την επόμενη, παρά μόνο να την «απενεργοποιήσουν» δηλαδή να μην είναι εκτελέσιμη.

1η Διδακτική ώρα: Γνωριμία με το περιβάλλον

- Παρουσίαση του περιβάλλοντος από τον καθηγητή.
- Παρουσίαση χρήσης από τον καθηγητή.
- Δημιουργία λογαριασμών μαθητών στο περιβάλλον.
- Εξοικείωση με το περιβάλλον του Pencil Code: Αφήνουμε τους μαθητές να δοκιμάσουν μόνοι τους όσα διαθέσιμα πλακίδια θέλουν.

Προτείνουμε στους μαθητές να αποθηκεύουν τις δραστηριότητες των φύλλων εργασίας σε ξεχωριστό αρχείο κάθε φορά, ανά διδακτική ώρα.

Επισημαίνουμε στους μαθητές ότι μπορούμε να κάνουμε ανενεργή κάθε εντολή, αν προσθέσουμε στην αρχή της το σύμβολο #. Έτσι δε χρειάζεται να σβήσουν τη μια δραστηριότητα πριν κάνουν την επόμενη, παρά μόνο να την «απενεργοποιήσουν».

2η Διδακτική ώρα: Εντολές εξόδου

ΣΤΟΧΟΙ

Στόχοι του μαθήματος είναι οι μαθητές να:

- αναφέρουν τις εντολές εξόδου που διαθέτει το περιβάλλον του Pencil Code
- περιγράφουν τη χρήση των εντολών εξόδου
- χρησιμοποιούν τις εντολές εξόδου

Σχέδιο μαθήματος

Μοιράζεται το φύλλο εργασίας (έντυπα) και ζητείται από τους μαθητές να το συμπληρώσουν.

Οι μαθητές δουλεύουν είτε μόνος του ο καθένας είτε συνεργατικά ανάλογα με τον αριθμό των μαθητών και των διαθέσιμων υπολογιστών.

Σε κάθε περίπτωση επιτρέπεται η επικοινωνία ανάμεσα στους μαθητές για την πραγματοποίηση του φύλλου εργασίας.

Ο διδάσκων κατά τη διάρκεια του μαθήματος κάνει ερωτήσεις στους μαθητές, για να διαπιστωθεί αν αντιληφθήκαν τη διαφοροποίηση στη σύνταξη των εντολών.

Σχολείο ..... Ημερομηνία.....

Ονοματεπώνυμο..... Τμήμα ..... Αρ. Καταλόγου .....

Φύλλο εργασίας – Εντολές εξόδου: Κείμενο, αριθμοί, ήχος

Σημείωση: Η εντολή `write` και `label` υπάρχουν στην ομάδα πλακιδίων **Text**

Η εντολή `say`, `play` και `tone` υπάρχουν στην ομάδα πλακιδίων **Sound**

Δοκιμάστε μια μια τις παρακάτω εντολές στο Pencil Code και συμπληρώστε στον πίνακα το αποτέλεσμα (Output) των εντολών.

Εντολή	**αποτέλεσμα**
`write 'καλημέρα'`	
`write 'προσθέτω 3 + 5'`	
`write 3+5`	
`write 2+4*5`	
`write (2 + 4)/3`	
`write '3' + '5'`	
`write Number('3') + Number('5')`	
`write "3 + 5 = #{3 + 5}"`	
`write "3 + 5 =", 3 + 5`	
`write 'Είμαι <u> μαθητής </u>'`	
`write '<p> Σήμερα έχει κρύο. </p> <p> Είναι Χειμώνας.</p>'`	
`alert 'καλημέρα'`	
`say 'My name is John'`	
`play 'EDCDEEEzDDDzEEE'`	
`tone 'A'`	

Δώστε τις κατάλληλες εντολές ώστε να εμφανιστούν, με τη μορφοποίηση που σας ζητείται, τα εξής:

	Εντολή
Η λέξη ΕΛΛΑΔΑ (με έντονη γραφή)	
Το γινόμενο του αριθμού 27 με τον αριθμό 35	
Να γραφεί ως ταμπέλα πάνω στη χελώνα η φράση: είμαι μια έξυπνη χελώνα	
Να ακούσετε τη φράση: Game over	

Πειραματιστείτε με τη σύνθεση μιας δικής σας μελωδίας. ☺

3η Διδακτική ώρα: Εντολές εισόδου δεδομένων

ΣΤΟΧΟΙ

Στόχοι του μαθήματος είναι οι μαθητές να:

- εισάγουν δεδομένα μέσω πλήκτρων και κουμπιών στην οθόνη
- εισάγουν κείμενο και ομιλία

Σχέδιο μαθήματος

Ο διδάσκων δείχνει παραδείγματα στους μαθητές για τη χρήση των εντολών `read` και `readnum`, `click`, `button` και `keydown`. Στο σημείο αυτό δεν πρέπει να γίνει αναφορά στην έννοια των συναρτήσεων αλλά να διδαχθούν ως σύνταξη της εντολής. Γίνεται αναφορά στο γεγονός ότι η απάντηση στην εντολή `read` και `readnum`, αποθηκεύεται προσωρινά σε μια μεταβλητή χωρίς να γίνει περαιτέρω συζήτηση στο τι είναι μεταβλητή.

Στη συνέχεια μοιράζεται το φύλλο εργασίας όπου οι μαθητές καλούνται να δουν άλλα παραδείγματα των παραπάνω εντολών, εφαρμόζοντάς τα μόνοι τους στο Pencil Code, και στη συνέχεια να φτιάξουν κάποια δικά τους.

Σχολείο ..... Ημερομηνία.....

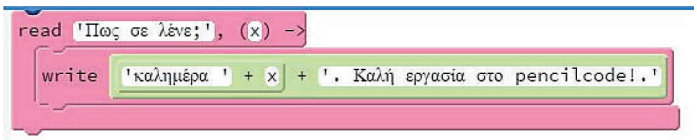
Ονοματεπώνυμο..... Τμήμα ..... Αρ. Καταλόγου .....

Φύλλο εργασίας – Είσοδος και έξοδος δεδομένων

Στις εικόνες βλέπετε τις εντολές με πλακίδια και στη συνέχεια δίνονται με κώδικα.

ΠΡΟΣΟΧΗ! Προσέχουμε η εντολή `write` να γραφτεί πιο μέσα από την εντολή `read` στο περιβάλλον με τον κώδικα είτε με 2 κενά είτε με ένα tab.

Δοκιμάστε τις επόμενες εντολές




```
read ‹Πως σε λένε;›, (x) ->
  write 'καλημέρα ' + x + '. Καλή εργασία στο Pencil Code!.'
```

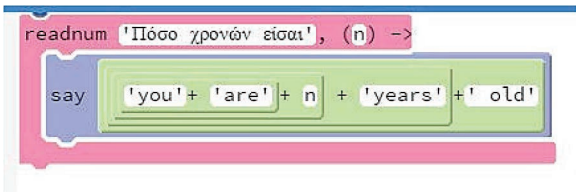
Δώστε την απάντησή σας και πατήστε Submit

Τροποποιήστε τις παραπάνω εντολές και φτιάξτε τη δική σας ερώτηση.

.....

.....

Δοκιμάστε τις επόμενες εντολές




```
readnum 'Πόσο χρονών είσαι', (n) ->
  say 'you'+ 'are'+ n + 'years'+' old'
```

Δώστε την απάντησή σας και πατήστε Submit

Τροποποιήστε τις παραπάνω εντολές και φτιάξτε τη δική σας ερώτηση.

.....

.....

.....

Δοκιμάστε τις επόμενες εντολές και δημιουργήστε και ανάλογες νέες δικές σας

```
click (e) ->
  tone 'B', 2
keydown 'A', ->
  write 'πάτησες το πληκτρο A'
button 'Good morning', ->
  say 'good morning'
```

4η-5η Διδακτική ώρα: Εντολές σχεδίασης

ΣΤΟΧΟΙ

Στόχος του μαθήματος είναι οι μαθητές να:

- δημιουργούν απλά σχέδια δίνοντας εντολές κίνησης, σε μια χελώνα, είτε χρησιμοποιώντας τη γεωμετρία της χελώνας είτε χρησιμοποιώντας καρτεσιανές συντεταγμένες.

Σχέδιο μαθήματος 4ης διδακτικής ώρας

Προτείνεται να μη μοιραστεί φύλλο εργασίας, αλλά οι επόμενες δραστηριότητες να ζητηθούν από τον διδάσκοντα γράφοντας τες στον πίνακα είτε με προβολή.

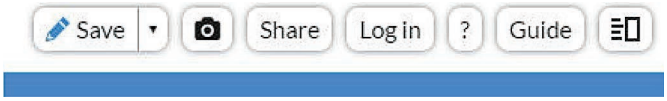
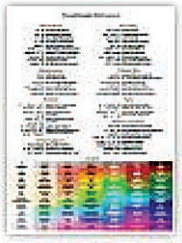
Λέμε κάθε φορά σε ποια ομάδα εντολών πλακιδίων θα τις βρουν. Καλό είναι επίσης να εξηγούμε τη σημασία των εντολών, καθώς και τη μετάφρασή τους στα ελληνικά.

Δραστηριότητες

– Ζητάμε από τους μαθητές να δοκιμάσουν τις εντολές

```
dot black, 100
dot orange, 50
```

– Τους λέμε ότι ο αριθμός δηλώνει την ακτίνα του κύκλου με μονάδα μέτρησης το εικονοστοιχείο και τους ζητάμε να δοκιμάσουν δικές τους τιμές και με διαφορετικά χρώματα. Τους δείχνουμε που μπορούν να βρουν περισσότερα διαθέσιμα:

τα διαθέσιμα χρώματα μπορείτε να τα βρείτε αν επιλέξετε Guide	
και στη συνέχεια QuickReferenceSheet http://pencilcode.net/material/reference.pdf) ή κατευθείαν πάνω στη διπλανή εικόνα	

– Ζητάμε από τους μαθητές να δοκιμάσουν τις παρακάτω εντολές:

```
dot pink, 70
fd 100
box gold, 60
dot midnightblue, 25
hide()
```

– Ζητάμε από τους μαθητές να μας πουν τι νομίζουν ότι κάνουν οι νέες εντολές που συνάντησαν παραπάνω

– Ας δοκιμάσουν τώρα την εντολή

```
cs()
```

Καλό είναι να χρησιμοποιούμε αυτή την εντολή ανάμεσα στις δοκιμές

– Ζητάμε τώρα να δοκιμάσουν τις εξής εντολές:

```
show()
home()
pen blue, 10
fd 100
rt 90
bk 150
lt 90
fd 50
pu()
fd 50
dot red, 40
jumpto 100,100
```

– Συζητάμε με τους μαθητές τι κάνουν οι παραπάνω εντολές.

– Ζητάμε από τους μαθητές να φτιάξουν ένα τετράγωνο με μήκος πλευράς 100 pixels, πάχος γραμμής 5 και με διαφορετικό χρώμα στην κάθε πλευρά.

Ας δοκιμάσουν τις επόμενες εντολές (επισημαίνουμε ότι η δεύτερη εντολή γράφεται πατώντας στην αρχή δύο κενά ή ένα tab).

```
click (e) ->
  moveto e.x, e.y
```

Εξηγούμε ότι το x και y είναι καρτεσιανές συντεταγμένες. Στη συνέχεια λέμε στους μαθητές να κάνουν "κλικ" σε διάφορα σημεία του χώρου σχεδίασης.

– Ζητάμε να δοκιμάσουν τις εντολές

```
pen olive, 5
rt 180, 50
```

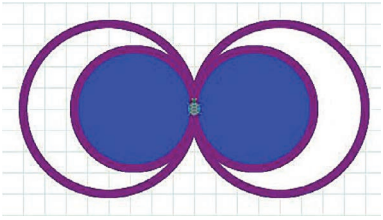
– Ζητάμε να φτιάξουν έναν κύκλο ακτίνας 100 εικονοστοιχείων

– Αφού φτιάξουν τον κύκλο, χωρίς να τον σβήσουν να δώσουν την εντολή

```
fill red
```

– Ζητάμε από τους μαθητές να φτιάξουν το παρακάτω σχήμα:

(μπορεί να υλοποιηθεί με διαφορετικούς τρόπους, το συζητάμε αυτό στην τάξη και κάθε τρόπο που βλέπουμε να χρησιμοποιείτε ζητάμε από τον μαθητή να καταγράψει τις εντολές που χρησιμοποίησε στον πίνακα, με σωστή σύνταξη, όταν η πλειοψηφία της τάξης ολοκληρώσει τη δραστηριότητα.)



Προτείνεται, όσοι μαθητές ολοκληρώσουν τις προηγούμενες δραστηριότητες, να τους ζητηθεί να φτιάξουν ένα δικό τους πρωτότυπο σχήμα

Σχέδιο μαθήματος 5ης διδακτικής ώρας

Μοιράζεται το επόμενο φύλλο εργασίας.

Επισημαίνεται η αρχική και η τελική θέση της χελώνας. Επίσης ο διδάσκων υπενθυμίζει στους μαθητές στοιχεία από τη γεωμετρία για τη σχεδίαση του 2ου σχήματος.

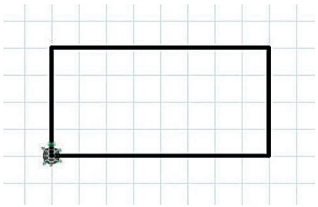
Σημείωση: Στην περίπτωση που οι μαθητές έχουν εξοικειωθεί με τη χρήση της χελώνας σε κάποιο άλλο logolike περιβάλλον, αυτή η διδακτική ώρα μπορεί να παραλειφθεί.

Σχολείο ..... Ημερομηνία.....

Ονοματεπώνυμο..... Τμήμα ..... Αρ. Καταλόγου .....

Φύλλο εργασίας – Εντολές σχεδίασης

1) Να δώσετε τις εντολές για τον σχηματισμό του παρακάτω ορθογωνίου παραλληλογράμμου με πλευρές 50 και 100 pixels. Καταγράψτε στο φύλλο εργασίας τις εντολές.



.....

.....

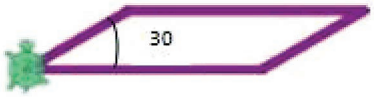
.....

.....

.....

.....

2) Να δώσετε τις εντολές για τον σχηματισμό του παρακάτω παραλληλογράμμου με πλευρές 100 και 200 pixels. Καταγράψτε στο φύλλο εργασίας τις εντολές.



.....

.....

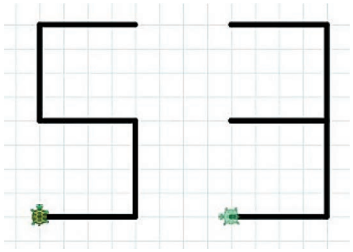
.....

.....

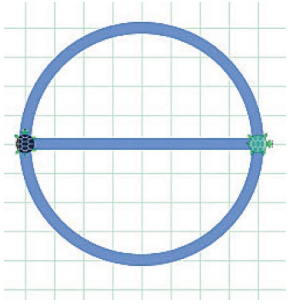
.....

.....

3) Γράψτε τις εντολές προκειμένου η χελώνα να ζωγραφίσει, τον ακόλουθο αριθμό, όπως φαίνεται. Η χελώνα θα ξεκινήσει από το κάτω μέρος του αριθμού 5 και θα τελειώσει στο κάτω μέρος του αριθμού 3, όπως φαίνεται στο σχήμα. Τα ευθύγραμμα τμήματα έχουν μήκος 50 pixels.



4) Γράψτε τις εντολές προκειμένου η χελώνα να ζωγραφίσει το ακόλουθο σχήμα, όπου φαίνεται η αφετηρία της χελώνας και το τέλος. Η ακτίνα του κύκλου είναι 50 pixels. Καταγράψτε στο φύλλο εργασίας τις εντολές.



.....

.....

.....

.....

.....

.....

6η Διδακτική ώρα: Δομή Επανάληψης: Η εντολή for

ΣΤΟΧΟΙ

Στόχοι του μαθήματος είναι οι μαθητές να:

- ξεχωρίζουν πότε μπορούν να χρησιμοποιήσουν δομή επανάληψης
- χρησιμοποιούν την εντολή `for`

Σχέδιο μαθήματος

Προτείνεται οι μαθητές να εργαστούν συνεργατικά ανά δυο.

Μοιράζεται το φύλλο εργασίας. Ο διδάσκων λέει στους μαθητές ότι η δομή επανάληψης `for` βρίσκεται στα πλακίδια στην ομάδα εντολών **Control**–ελέγχου με χρώμα πορτοκαλί ανοιχτό.

Ζητείται από τους μαθητές, να κάνουν την πρώτη δραστηριότητα χρησιμοποιώντας τα πλακίδια.

Ο διδάσκων συζητά με τους μαθητές τη χρήση της εντολής `for` .

Τους ζητά να δουν την εντολή στο περιβάλλον κειμένου και να προσέξουν τη σύνταξη της.

Είναι σημαντικό να αντιληφθούν ότι οι εντολές που είναι μέσα στο βρόχο και επαναλαμβάνονται, γράφονται με μια εσοχή στην αρχή της γραμμής, δηλαδή όχι στην ίδια "κάθετο" με την εντολή `for`. Η εσοχή αυτή δημιουργείται είτε πατώντας δύο κενά είτε το πλήκτρο tab.

Κατά τη συμπλήρωση του φύλλου εργασίας, ο διδάσκων ενθαρρύνει τους μαθητές στη συνεργασία με τον συμμαθητή τους.

Σχολείο ..... Ημερομηνία.....

Ονοματεπώνυμο..... Τμήμα ..... Αρ. Καταλόγου .....

Φύλλο εργασίας – Δομή επανάληψης- η εντολή for

Δοκιμάστε τις παρακάτω εντολές χρησιμοποιώντας τα πλακίδια και στη συνέχεια δείτε τις ως κώδικα. Προσέξτε τη σύνταξή τους.

Εντολές- Κώδικας	**Πλακίδια**	**Αποτέλεσμα - output**
`for [1..5]` `  write 'καλημέρα'`		

Χρησιμοποιώντας την εντολή `for`, δώστε τις κατάλληλες εντολές ώστε να ακουστεί 4 φορές η φράση GOOD MORNING.

Γράψτε τις εντολές στο φύλλο εργασίας, προσέχοντας τη σύνταξη τους.

.....

.....

Δώστε κατάλληλες εντολές για να φτιαχτεί το παρακάτω σχήμα, το ύψος κάθε σκαλιού είναι 50 εικονοστοιχεία και το πλάτος τους 80 εικονοστοιχεία. Η χελώνα ξεκινάει από κάτω και τελειώνει πάνω, όπως φαίνεται στην εικόνα. Γράψτε τις εντολές στο φύλλο εργασίας, προσέχοντας τη σύνταξή τους.

	Γράψτε τις εντολές
α) αναλυτικά οι εντολές χωρίς τη χρήση της εντολής `for`	....
β) με την εντολή `for`.	....

Δοκιμάστε τις παρακάτω εντολές στο περιβάλλον του κώδικα. Προσέξτε τη σύνταξή τους.

κώδικας	**Αριθμός επαναλήψεων**

`for [1..4]`
`  box purple, 30`
`  fd 50`	

`for [1...4]`
`  box purple, 30`
`  fd 50`	

Δώστε κατάλληλες εντολές (Χρησιμοποιήστε την εντολή `for`) για να φτιαχτεί το παρακάτω σχήμα.

.....

.....

.....

.....

.....

.....

.....

.....

7η Διδακτική ώρα: Η εντολή for – Κανονικά πολύγωνα

ΣΤΟΧΟΙ

Στόχοι του μαθήματος είναι οι μαθητές να:

- χρησιμοποιούν την εντολή `for` για τη δημιουργία κανονικών πολυγώνων
- αναγνωρίζουν ότι η χελώνα στρίβει κατά την εξωτερική γωνία του πολυγώνου
- υπολογίζουν την εξωτερική γωνία του κανονικού πολυγώνου από το πλήθος των κορυφών του

Σχέδιο μαθήματος

Μοιράζεται το φύλλο εργασίας στους μαθητές.

Ο διδάσκων ζητά από τους μαθητές να κάνουν τη πρώτη δραστηριότητα. Με αφορμή το αποτέλεσμα της πρώτης δραστηριότητας, γίνεται συζήτηση με τους μαθητές, για την τιμή της στροφής, με τη μαιευτική μέθοδο.

Οι μαθητές καλούνται να συμπληρώσουν το υπόλοιπο φύλλο εργασίας.

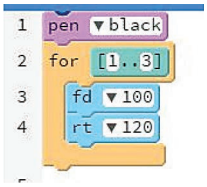
Σχολείο ..... Ημερομηνία.....

Ονοματεπώνυμο..... Τμήμα ..... Αρ. Καταλόγου .....

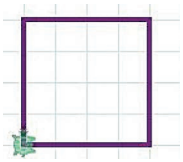
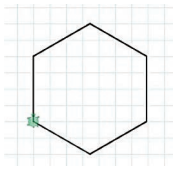
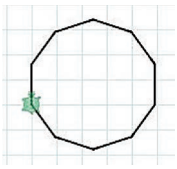
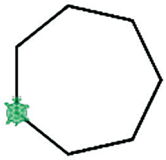
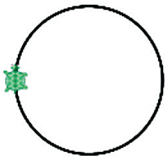
Φύλλο εργασίας – Δομή επανάληψης- Κανονικά Πολύγωνα

1) Δοκιμάστε τις παρακάτω εντολές. Τι σχήμα προκύπτει; **Τι παρατηρείτε;**

Εντολές- Κώδικας	Πλακίδια	Σχήμα

`pen black`
`for [1..3]`
`  fd 100`
`  rt 120`
		

2) Δώστε τις κατάλληλες εντολές για τη δημιουργία των παρακάτω σχημάτων, **χρησιμοποιώντας την εντολή** `for` (η χελώνα αρχίζει και τελειώνει στην ίδια θέση που φαίνεται). Γράψτε τις εντολές στο φύλλο εργασίας, προσέχοντας τη σύνταξή τους.

		εντολές
Τετράγωνο		
Κανονικό εξάγωνο		
Κανονικό δεκάγωνο		
Κανονικό επτάγωνο		
Κύκλος **(προσεγγιστικά ως ένα κανονικό πολύγωνο με πολλές γωνίες)**		

☞ Για όσους έχουν χρόνο ☺: Δοκιμάστε να φτιάξετε το παρακάτω σχήμα

8η-10η Διδακτική ώρα: Υποπρογράμματα-συναρτήσεις

ΣΤΟΧΟΙ

Στόχοι του μαθήματος είναι οι μαθητές να:

- δημιουργούν συναρτήσεις
- δημιουργούν προγράμματα τα οποία καλούν συναρτήσεις

Σχέδιο μαθήματος

Τα υποπρογράμματα και η χρήση τους είναι μια απαραίτητη βασική γνώση και δεξιότητα που πρέπει να αποκτήσουν οι μαθητές. Στο περιβάλλον του Pencil Code τα υποπρογράμματα υλοποιούνται με συναρτήσεις.

Πριν οι μαθητές αρχίσουν να εργάζονται, ο διδάσκων δείχνει ένα παράδειγμα για τη δημιουργία υποπρογράμματος στον υπολογιστή του, πρώτα χρησιμοποιώντας πλακίδια και στη συνέχεια τους δείχνει τον κώδικα και επισημαίνει τη σύνταξη: Οι εντολές που περιλαμβάνονται στο υποπρόγραμμα γράφονται είτε με δύο κενά στην αρχή κάθε γραμμής εντολής είτε με ένα πάτημα του πλήκτρου tab στην αρχή.

Επισημαίνει στους μαθητές ότι, όταν δημιουργούν ένα υποπρόγραμμα, δεν εκτελούνται οι εντολές που περιέχει, παρά μόνο αποθηκεύονται και με το όνομα της συνάρτησης μπορούμε να τις καλέσουμε και να εκτελεστούν.

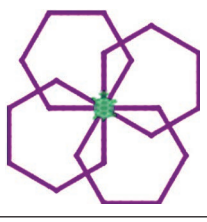
Στη συνέχεια ο διδάσκων καλεί τη συνάρτηση, χρησιμοποιώντας πλακίδια. Επισημαίνει στους μαθητές τα διαφορετικά πλακίδια για τη δημιουργία και τη χρήση συνάρτησης.

Έπειτα ο διδάσκων με τη χρήση των κατωτέρω παραδειγμάτων και με χρήση πλακιδίων αλλά και με κώδικα δημιουργεί πρώτα τη συνάρτηση, "προσπαθεί" να την εκτελέσει για να δείξει στους μαθητές ότι οι εντολές μιας συνάρτησης εκτελούνται μόνο, όταν την καλέσεις. Στη συνέχεια κάνει διαδοχικές κλήσεις της συνάρτησης για τη δημιουργία του επιθυμητού σχήματος

Παράδειγμα 1

`εξάγωνο = () ->`
`  pen purple, 3`
`  for [1..6]`
`  fd 40`
`  rt 60`
	Ο κώδικας δε βγάζει αποτέλεσμα
	(Είναι μια συνάρτηση)

`εξάγωνο()`	

`for [1..4]`
`  εξάγωνο()`
`  rt 90`
	

Παράδειγμα2

`circle1=()->`
`  pen black, 10`
`  rt 360, 70`
`  fill red`
`circle2 = ()->`
`  pen green, 6`
`  rt 360, 50`
`  fill yellow`
`circle3 = ()->`
`  pen purple, 3`
`  rt 360, 30`
`  fill cyan`
	Ο κώδικας δε βγάζει αποτέλεσμα (ορισμός συναρτήσεων)

`speed 20`
`circle1()`
`jumpxy 0, 120`
`circle2()`
`jumpxy 0, 80`
`circle3()`
	

Μοιράζεται το 1ο φύλλο εργασίας στους μαθητές. Επισημαίνεται στους μαθητές να είναι προσεκτικοί στην ανάγνωση των οδηγιών του φύλλου εργασίας.

Υπενθυμίζουμε στους μαθητές να αποθηκεύσουν τις συναρτήσεις κάθε μαθήματος σε διαφορετικό αρχείο. Έχουν όλες τις συναρτήσεις μαζί και επιλέγουν κάθε φορά ποια θέλουν να καλέσουν.

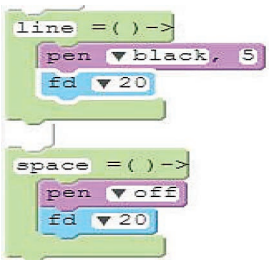
Μπορούν να διατεθούν άλλες 2 διδακτικές ώρες με τα επόμενα 2 φύλλα εργασίας. Αφήνουμε τους μαθητές να συνεργαστούν για την υλοποίησή τους.

Σχολείο ..... Ημερομηνία.....

Ονοματεπώνυμο..... Τμήμα ..... Αρ. Καταλόγου .....

Φύλλο εργασίας – Συναρτήσεις

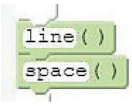
⌨ Δώστε τις παρακάτω εντολές που θα δημιουργήσουν τις συναρτήσεις `line` και `space`. Τα πλακίδια των συναρτήσεων που θα χρειαστείτε είναι στην ομάδα Operators, με πράσινο χρώμα




```
line =()->
  pen black, 5
  fd 20

space =()->
  pen off
  fd 20
```

Μόλις δημιουργήσατε τις πρώτες σας συναρτήσεις.

⌨ **Χωρίς να σβήσετε** τις παραπάνω εντολές, δοκιμάστε τις εξής. Τι παρατηρείτε;




```
line()
space()
```

☞ **Σβήστε τις τελευταίες 2 εντολές, αλλά ΜΗ σβήσετε τις συναρτήσεις `line` και `space`**. Χρησιμοποιήστε τις εντολές `line()`, `space()` και `for` και όποια άλλη εντολή χρειάζεται και δημιουργήστε το παρακάτω σχήμα. Σημειώστε τις εντολές στο φύλλο εργασίας, προσέχοντας τη σύνταξή τους.

.....

.....

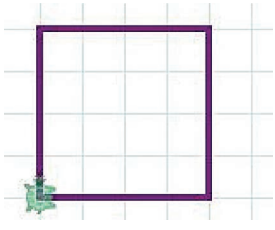
.....

.....

.....

.....

☞ Δημιουργήστε μια συνάρτηση με το όνομα `tetragono` για τη δημιουργία ενός τετραγώνου και καλέστε την ώστε να το δημιουργήσει. Προσοχή! Η χελώνα θα καταλήγει στην αρχική της θέση. Χρησιμοποιήστε την εντολή for. Σημειώστε τις εντολές στο φύλλο εργασίας, **προσέχοντας τη σύνταξή τους**.



.....

.....

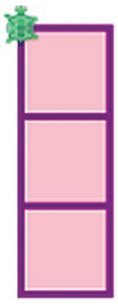
.....

.....

.....

.....

☞ Χρησιμοποιήστε τη συνάρτηση `tetragono()` σε συνδυασμό με την εντολή `for` και όποιες άλλες εντολές χρειάζονται για να δημιουργήσετε το παρακάτω σχήμα. Σημειώστε τις εντολές στο φύλλο εργασίας, **προσέχοντας τη σύνταξή τους**.



.....

.....

.....

.....

.....

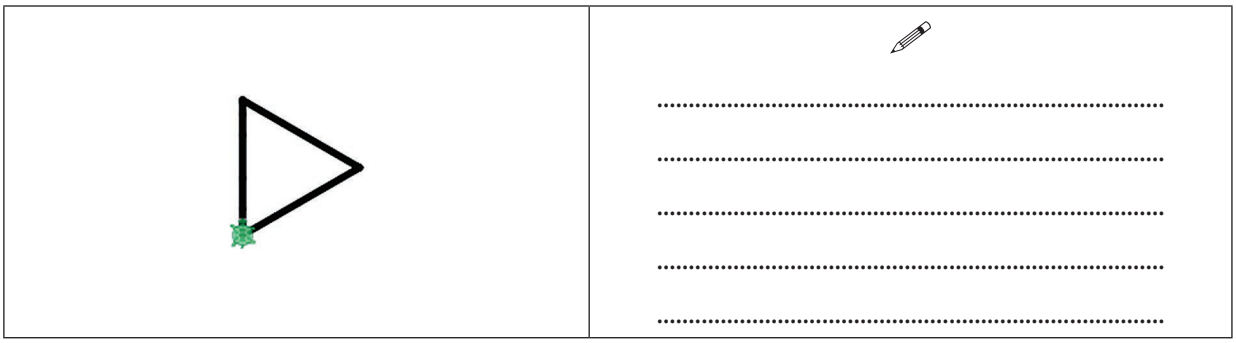
.....

Σχολείο ..... Ημερομηνία.....

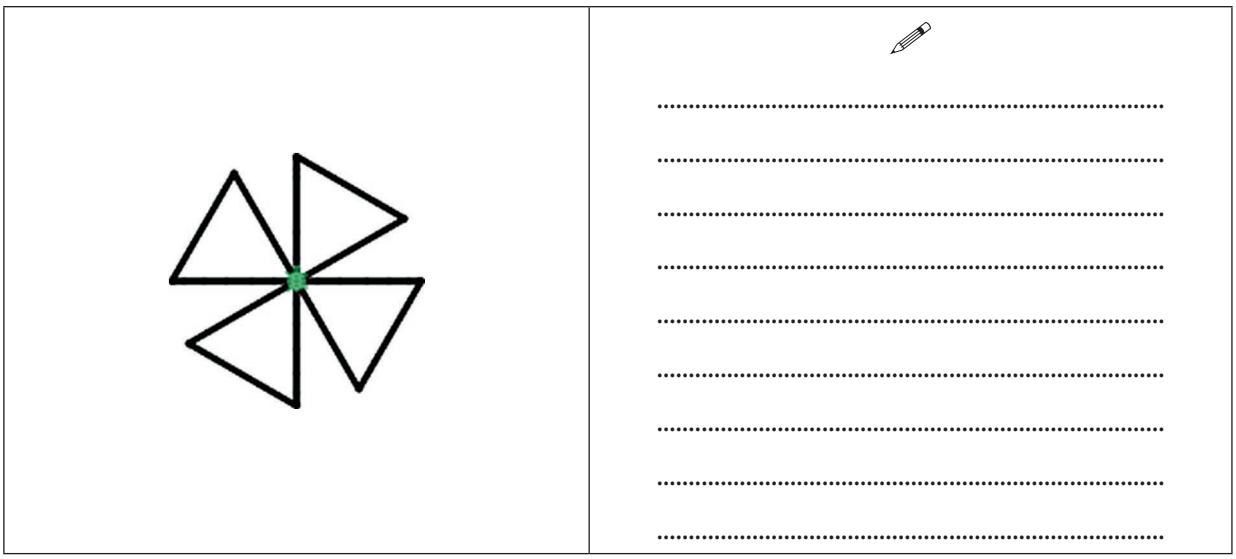
Ονοματεπώνυμο..... Τμήμα ..... Αρ. Καταλόγου .....

Φύλλο εργασίας – Συναρτήσεις-σχήματα εκ περιστροφής

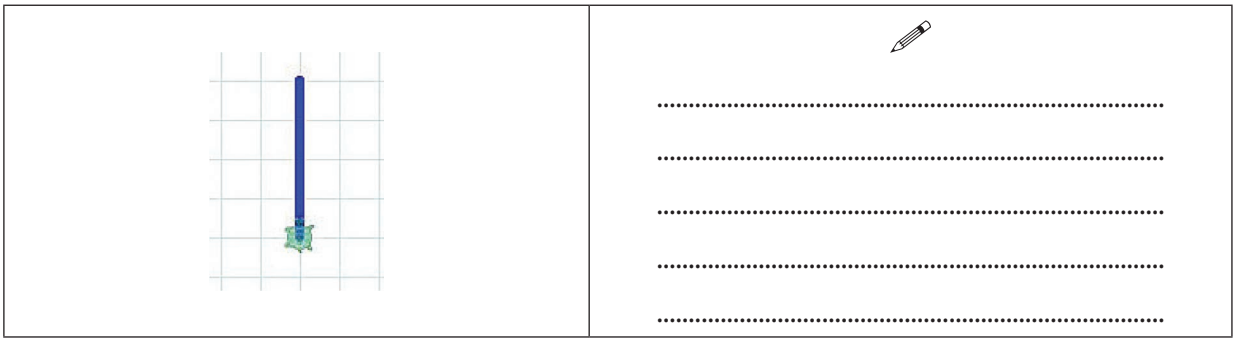
☞ Δημιουργήστε συνάρτηση με το όνομα `isopleuro` για τη δημιουργία ισοπλεύρου τριγώνου, με πλευρές μήκους 60 εικονοστοιχείων, πάχους γραμμής 5, και χρώματος μαύρο. Καλέστε την ώστε να δημιουργηθεί το σχήμα που βλέπετε.



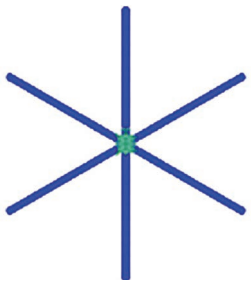
☞ Στη συνέχεια χρησιμοποιώντας τη συνάρτηση `isopleuro` σε συνδυασμό με την εντολή `for` και άλλες εντολές δημιουργήστε το παρακάτω σχήμα.



☞ Δημιουργήστε συνάρτηση με το όνομα `aktina` για τη δημιουργία ευθυγράμμου τμήματος ακτίνας μήκους 50 εικονοστοιχείων, χρώματος μπλε και πάχος γραμμής 3 (Προσοχή! Η χελώνα αρχίζει και τελειώνει στην ίδια θέση).



☞ Χρησιμοποιώντας τη συνάρτηση `aktina`, την εντολή `for` και όποιες άλλες εντολές νομίζετε, φτιάξτε το επόμενο σχήμα.



.....

.....

.....

.....

.....

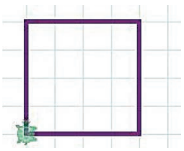
☞ Δημιουργήστε συνάρτηση για τη δημιουργία του σχήματος που επαναλαμβάνεται και στη συνέχεια καλέστε τη, χρησιμοποιώντας την εντολή `for` για τη δημιουργία του παρακάτω σχήματος.

Σχολείο ..... Ημερομηνία.....

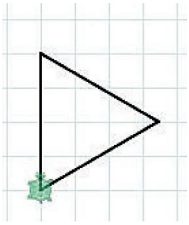
Ονοματεπώνυμο..... Τμήμα ..... Αρ. Καταλόγου .....

Φύλλο εργασίας – Υπερδιαδικασίες- Σπίτι

☞ Δημιουργήστε μια συνάρτηση `tetragono` για τη δημιουργία τετραγώνου, πλευράς 100 εικονοστοιχείων.

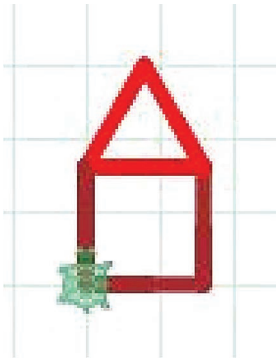


☞ Δημιουργήστε μια συνάρτηση με το όνομα `isopleuro` (ΜΗΣΒΗΣΕΤΕΤΗΣΥΝΑΡΤΗΣΗ `tetragono`) και στη συνέχεια καλέστε τη για να δημιουργήσετε το εξής ισόπλευρο τρίγωνο με μήκος πλευράς 100 εικονοστοιχείων.



☞ Χρησιμοποιώντας τις συναρτήσεις `tetragono` και `isopleuro` και όποιες άλλες εντολές χρειάζονται δημιουργήστε το παρακάτω σπίτι. Φροντίστε η χελώνα να αρχίσει και να τελειώσει στη θέση που φαίνεται. Το τετράγωνο να γίνει καφέ και η σκεπή κόκκινη. Αν θέλετε μπορείτε να γεμίσετε τη σκεπή με κόκκινο χρώμα.

☞ Δημιουργήστε μια συνάρτηση με το όνομα `spiti`, η οποία θα περιλαμβάνει τις συναρτήσεις και τις εντολές που χρησιμοποιήσατε προηγουμένως.



Δημιουργήστε μια συνάρτηση με το όνομα `village`, που θα φτιάχνει 4 σπίτια, το ένα δίπλα στο άλλο.

11η -12η Διδακτική ώρα: Συναρτήσεις με μεταβλητές

ΣΤΟΧΟΙ

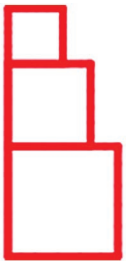
Στόχοι του μαθήματος είναι οι μαθητές να:

- δημιουργούν συναρτήσεις που να περιέχουν παραμέτρους
- δημιουργούν προγράμματα τα οποία καλούν συναρτήσεις με χρήση ορισμάτων

Σχέδιο μαθήματος

11η ώρα:

Ο διδάσκων μοιράζει το φύλλο εργασίας στους μαθητές. Τους ζητά να κοιτάξουν το πρώτο σχήμα:



Τους ζητά να του πουν πόσα τετράγωνα βλέπουν και πόσες συναρτήσεις πιστεύουν ότι πρέπει να δημιουργήσουν για τα τετράγωνα αυτά.

Ο διδάσκων γράφει στον πίνακα το εξής:

```
tetragonom = (x)->
  for [1..4]
    fd x
    rt 90
```

Γίνεται συζήτηση στην τάξη για τη μεταβλητή x.

Με μαιευτική μέθοδο, προκύπτει ποια πρέπει να είναι η εντολή με την οποία θα καλέσουμε τη συνάρτηση για τη δημιουργία τετραγώνου με συγκεκριμένο μήκος πλευράς.

```
tetragonom (20)
```

ή

```
tetragonom 20
```

ή

```
x=20
tetragonom (x)
```

Στη συνέχεια ο διδάσκων γράφει στον πίνακα

```
tetragono3m = (color, x, y)->
  pen color, y
  for[1..4]
    fd x
    rt 90
```

Γίνεται συζήτηση στην τάξη για τις νέες μεταβλητές

Με μαιευτική μέθοδο, προκύπτει ποια θα πρέπει να είναι η εντολή για να καλέσουμε τη συνάρτηση για τη δημιουργία τετραγώνου με συγκεκριμένο μήκος, χρώματος και πάχους πλευράς.

```
tetragono3m (red, 100, 5)
```

ή

```
tetragono3m red, 100, 5
```

Καλούνται τώρα οι μαθητές να συμπληρώσουν το φύλλο εργασίας.

12η ώρα:

Μοιράζεται το 2ο φύλλο εργασίας (είναι 2 σελίδες – το μοιράζουμε σε διαφορετικά φύλλα). Δεν αποτελεί στόχο η υλοποίηση ολόκληρου του φύλλου εργασίας σε μια διδακτική ώρα. Η δεύτερη σελίδα μπορεί να δοθεί στους μαθητές, ως προαιρετική εργασία για το σπίτι.

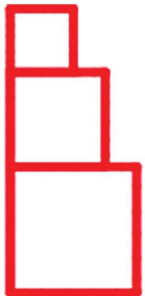
Σχολείο ..... Ημερομηνία.....

Ονοματεπώνυμο..... Τμήμα ..... Αρ. Καταλόγου .....

Φύλλο εργασίας (1) – Συναρτήσεις με Μεταβλητές

Έστω η παρακάτω συνάρτηση για τη δημιουργία τετραγώνου με μεταβλητό μήκος πλευράς. Αφού τη γράψετε στο περιβάλλον του Pencil Code, χρησιμοποιήστε τη για να δημιουργήσετε το παρακάτω σχήμα. Γράψτε στο φύλλο εργασίας τις εντολές που δώσατε για τη δημιουργία του. Τα τετράγωνα έχουν μήκη 40, 60 και 80 εικονοστοιχεία. Μπορείτε να το κάνετε με ότι χρώμα θέλετε.

```
tetragonom = (x)->
  for [1..4]
    fd x
    rt 90
```

....	

Δημιουργήστε μια συνάρτηση με το όνομα `kyklosm` για τη δημιουργία ενός κύκλου μαύρου χρώματος με μεταβλητή την ακτίνα του κύκλου. Σημειώστε τις εντολές στο φύλλο εργασίας, προσέχοντας τη σύνταξη.

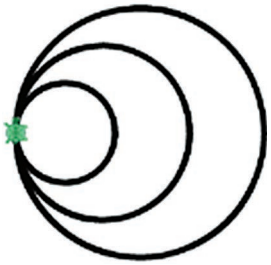
.....

.....

.....

.....

Χρησιμοποιήστε τη συνάρτηση `kyklosm` για τη δημιουργία του παρακάτω σχήματος. Οι κύκλοι έχουν ακτίνες 40, 60 και 80 εικονοστοιχεία αντίστοιχα. Σημειώστε τις εντολές στο φύλλο εργασίας.

	....

 Δημιουργήστε μια συνάρτηση με το όνομα `kyklos3m` για τη δημιουργία ενός κύκλου με μεταβλητές την ακτίνα του κύκλου, το χρώμα και το πάχος της γραμμής. Σημειώστε τις εντολές στο φύλλο εργασίας, προσέχοντας τη σύνταξη.

.....

.....

.....

.....

.....

.....

 Να καλέσετε τη συνάρτηση `kyklos3m` ώστε να εμφανιστούν:

i. ένας κόκκινος κύκλος με πάχος γραμμής 5 και ακτίνα 40 εικονοστοιχεία

.....

ii. ένας πράσινος κύκλος με πάχος γραμμής 10 και ακτίνα 60 εικονοστοιχεία

.....

iii. ένας μπλε κύκλος με πάχος γραμμής 15 και ακτίνα 80 εικονοστοιχεία

.....

Σχολείο ..... Ημερομηνία.....

Ονοματεπώνυμο..... Τμήμα ..... Αρ. Καταλόγου .....

Φύλλο εργασίας (2) – Συναρτήσεις με Μεταβλητές

1. Να δημιουργήσετε μια συνάρτηση με το όνομα `isopleuro3m` για τον σχεδιασμό ισοπλεύρου τριγώνου. (Μεταβλητές: το μήκος της πλευράς του τριγώνου, το πάχος της γραμμής και το χρώμα της γραμμής).

.....

.....

.....

.....

.....

.....

 Να καλέσετε τη συνάρτηση `isopleuro3m` ώστε να εμφανιστούν:

I. ένα κόκκινο τρίγωνο με πλευρά 100 εικονοστοιχεία και πάχος 5

.....

ii. ένα πορτοκαλί τρίγωνο με πλευρά 200 εικονοστοιχείακαι πάχος 4

.....

2. Γράψτε συνάρτηση με όνομα `polygono4m` που θα παίρνει ως μεταβλητές: τον αριθμό των πλευρών του πολυγώνου, το μήκος, το χρώμα και το πάχος των πλευρών και θα σχηματίζει κανονικό πολύγωνο.

.....

.....

.....

.....

.....

.....

 Πώς θα καλέσουμε τη συνάρτηση `polygono4m` προκειμένου να σχεδιάσουμε τα παρακάτω;

i. ένα πράσινο τρίγωνο με πλευρά 100 εικονοστοιχεία και πάχος γραμμής 15

.....

ii. ένα κόκκινο τετράγωνο με πλευρά 120 εικονοστοιχεία και πάχος γραμμής 10

.....

iii. ένα μπλε εξάγωνο με πλευρά 80 εικονοστοιχεία και πάχος γραμμής 5

.....

3. Να δημιουργήσετε συνάρτηση με το όνομα `skala3m` για τη δημιουργία μιας σκάλας που δέχεται ως μεταβλητές το πλήθος των σκαλοπατιών, το ύψος και το βάθος κάθε σκαλοπατιού, μαύρου χρώματος και πάχος γραμμής 5.

.....

.....

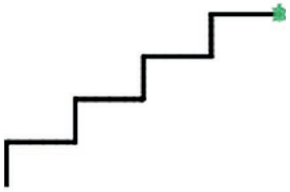
.....

.....

.....

.....

Πώς θα καλέσουμε τη συνάρτηση `skala3m` προκειμένου να σχεδιάσουμε το παρακάτω σχήμα (ύψος σκαλιού 50 εικονοστοιχεία, βάθος 80 εικονοστοιχεία).



.....

4. Να δημιουργήσετε μια συνάρτηση με το όνομα `ilios3m` που θα σχεδιάζει ένα σχήμα όπως το παρακάτω. Η διαδικασία θα δέχεται σαν παραμέτρους τον αριθμό των ακτίνων, το μήκος της ακτίνας και το χρώμα της. Το πάχος της γραμμής να είναι 3.

.....

.....

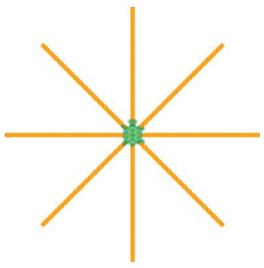
.....

.....

.....

.....

Πώς θα καλέσουμε τη συνάρτηση `ilios3m` προκειμένου να σχεδιάσουμε το παρακάτω σχήμα (μήκος ακτίνας 100 εικονοστοιχεία, χρώμα πορτοκαλί.



.....

13η-14η Διδακτική ώρα: Η εντολή for με μεταβλητές από λίστα

ΣΤΟΧΟΙ

Στόχος του μαθήματος είναι οι μαθητές να:

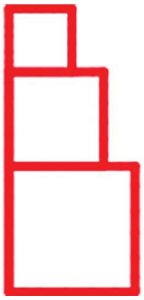
- δημιουργούν βρόχους επανάληψης με χρήση μεταβλητής που λαμβάνει τιμή από λίστα τιμών

Σχέδιο μαθήματος

13η ώρα:

Ο διδάσκων θυμίζει στους μαθητές μια προηγούμενη δραστηριότητα:

Σε προηγούμενη δραστηριότητα είχαμε δημιουργήσει συνάρτηση με το όνομα `tetragonom` για τη δημιουργία τετραγώνου (η χελώνα αρχίζει και τελειώνει στην ίδια θέση) με μεταβλητή το μήκος της πλευράς. Στη συνέχεια είχαμε καλέσει τη συνάρτηση για τη δημιουργία του παρακάτω σχήματος.Τα τετράγωνα έχουν μήκος πλευρών 40, 60 και 80 εικονοστοιχεία.



Οι εντολές για να γίνει το παραπάνω σχήμα, καλώντας τη διαδικασία `tetragonom` ήταν οι εξής:

```
tetragonom(80)
fd 80
tetragonom(60)
fd 60
tetragonom(40)
```

Ο διδάσκων με επίδειξη, δίνει στους μαθητές την εξής εναλλακτική (με κώδικα).

```
for x in [80, 60, 40]
  tetragonom(x)
  fd x
```

Το γράφει και στον πίνακα της τάξης.

Γίνεται μικρή συζήτηση για τη δυνατότητα της μεταβλητής να παίρνει τιμές από συγκεκριμένη λίστα τιμών. Αναφέρεται η λέξη λίστα, αλλά δεν εξηγούμε στην παρούσα φάση περισσότερα.

Μοιράζεται το 1ο φύλλο εργασίας

14η ώρα:

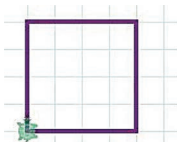
Μοιράζεται το 2ο φύλλο εργασίας

Σχολείο ..... Ημερομηνία.....

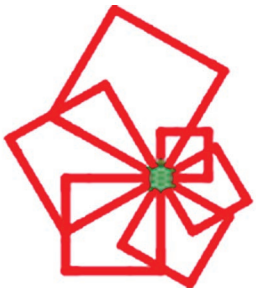
Ονοματεπώνυμο..... Τμήμα ..... Αρ. Καταλόγου .....

Φύλλο εργασίας (1) – Μεταβλητές και η εντολή for

Δημιουργήστε μια συνάρτηση με το όνομα `tetragonom` για τη δημιουργία τετραγώνου (η χελώνα αρχίζει και τελειώνει στην ίδια θέση) με μεταβλητή το μήκος της πλευράς.



Χρησιμοποιήστε τη συνάρτηση `tetragonom` και την εντολή `for`, ώστε να δημιουργήσετε το επόμενο σχήμα: (τα τετράγωνα έχουν μήκος πλευρών 30, 40, 50, 60, 70, 80 εικονοστοιχεία)



.....

.....

.....

.....

.....

Δημιουργήστε μια συνάρτηση με το όνομα `kyklosc` για τη δημιουργία κύκλου ακτίνας 100 εικονοστοιχείων με μεταβλητή το χρώμα του

.....

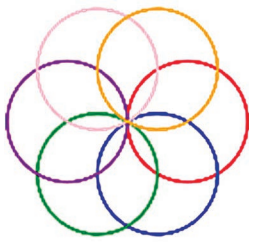
.....

.....

.....

.....

Χρησιμοποιήστε τη συνάρτηση `kyklosc` και την εντολή `for`, ώστε να δημιουργήσετε το επόμενο σχήμα: (οι κύκλοι έχουν 6 διαφορετικά χρώματα).



.....

.....

.....

.....

.....

Προσπαθήστε να δημιουργήσετε ένα παραλληλόγραμμο με πλευρές 50 και 100 εικονοστοιχεία, χρησιμοποιώντας την εντολή `for` με μεταβλητή που θα παίρνει τιμές από μια λίστα.

Σχολείο ..... Ημερομηνία.....

Ονοματεπώνυμο..... Τμήμα ..... Αρ. Καταλόγου .....

Φύλλο εργασίας(2) – Μεταβλητές και η εντολή for

Δοκιμάστε τις παρακάτω εντολές

```
for x in [13, 35, 2]
  write 'Κερδίζει ο αριθμός ', x
```

Πόσες επαναλήψεις έγιναν; ___

Δοκιμάστε τις παρακάτω εντολές

```
for x in [0..4]
  write 'ο αριθμός ', x
```

Πόσες επαναλήψεις έγιναν;___

Ποια ήταν τιμή του x στην 1η επανάληψη;___

Ποια ήταν τιμή του x στην τελευταία επανάληψη;___

Δοκιμάστε τις παρακάτω εντολές

```
for x in [0...4]
  write 'ο αριθμός ', x
```

Πόσες επαναλήψεις έγιναν;___

Ποια ήταν τιμή του x στην 1η επανάληψη;___

Ποια ήταν τιμή του x στην τελευταία επανάληψη;___

Δοκιμάστε τιςπαρακάτω εντολές

```
for x in [10..20] by 2
  write x
```

Ποιοι αριθμοί εμφανίστηκαν; _____

Με ανάλογο τρόπο, ζητήστε να εμφανιστούν όλοι οι αριθμοί από το 1 έως το 100 που είναι πολλαπλάσια του 5.

Γράψτε τις εντολές που χρησιμοποιήσατε:

.....

.....

.....

Χρησιμοποιήστε την εντολή `for` με μεταβλητή για να εμφανιστεί το παρακάτω. Γράψτε τις εντολές που χρησιμοποιήσατε.

ο αριθμός 10 ο αριθμός 43 ο αριθμός 6 ο αριθμός 27	....

Χρησιμοποιήστε την εντολή for με μεταβλητή για να εμφανιστεί το παρακάτω. Γράψτε τις εντολές που χρησιμοποιήσατε.

Γεια σου Ελένη Γεια σου Γιάννη Γεια σου Μαρία	....

Δοκιμάστε τις παρακάτω εντολές

```
for color in [red, orange, yellow, blue]
  pen color, 4
  fd 100
  rt 90
```

Δημιουργήστε με ανάλογο τρόπο ένα κανονικό οκτάγωνο με διαφορετικά χρώματα στην κάθε πλευρά.

Γράψτε τις εντολές που χρησιμοποιήσατε:

.....

.....

.....

Δοκιμάστε τις παρακάτω εντολές

```
pen purple
for x in [50..1] by -1
  rt 30, x
```

Τροποποιήστε τις παραπάνω εντολές ώστε η σπείρα να ξεκινήσει από μέσα προς τα έξω.

Γράψτε τις εντολές που χρησιμοποιήσατε:

.....

.....

.....

15η Διδακτική ώρα: Δημιουργώ μεταβλητές

ΣΤΟΧΟΙ

Στόχοι του μαθήματος είναι οι μαθητές να:

- δημιουργούν μεταβλητές
- περιγράφουν τη λειτουργία τους μέσα σε ένα πρόγραμμα
- προβλέπουν την τιμή μιας μεταβλητής μετά από την εκτέλεση του προγράμματος

Ο διδάσκων ζητά από τους μαθητές να κάνουν τις επόμενες δραστηριότητες.

Χρησιμοποιεί τον πίνακα και γίνεται συζήτηση.

Σχολείο ..... Ημερομηνία.....

Ονοματεπώνυμο..... Τμήμα ..... Αρ. Καταλόγου .....

Φύλλο εργασίας – Δημιουργώ μεταβλητές

Δοκιμάστε τις παρακάτω εντολές ΜΙΑ ΜΙΑ και γράψτε τι εμφανίζεται ή τι ακούγεται κάθε φορά. (αν δεν εμφανίζεται κάτι γράψτε -)

`x=3`	
`write x*x`	
`x='Maria'`	
`write x`	
`say x`	
`write 'x'`	

Για να δώσω την τιμή 5 στη μεταβλητή poso ποια εντολή θα δώσω;

.....

Για να εμφανίσω την τιμή της μεταβλητής ποια εντολή θα δώσω;

.....

Σβήστε τις παραπάνω εντολές και δοκιμάστε πάλι την εντολή

```
write x
```

Τι παρατηρούμε;

Ακολουθεί συζήτηση

Δοκιμάστε τις παρακάτω εντολές ΜΙΑ ΜΙΑ και γράψτε τι εμφανίζεται ή τι ακούγεται κάθε φορά. (Αν δεν εμφανίζεται κάτι γράψτε -)

`x=5`	
`x=x+3`	
`write x`	
`write x*2`	
`y=10`	
`for z in [1..5]` `  write y*z`	

Δοκιμάστε τις παρακάτω εντολές ΜΙΑ ΜΙΑ και γράψτε τι εμφανίζεται ή τι ακούγεται κάθε φορά. (Αν δεν εμφανίζεται κάτι γράψτε -)

`x=7`	
`y=x+6`	
`write y`	

Δοκιμάστε τις παρακάτω εντολές

```
await read 'Πώς σε λένε;', defer x
alert 'Γεια σου ' + x
```

Γράψτε μια διαδικασία με όνομα Ginomeno η οποία να περιέχει την εντολή `await` και όταν εκτελείται:

α) να ρωτάει με κατάλληλο μήνυμα ένα αριθμό και να τον αποθηκεύει σε μια μεταβλητή με το όνομα Α1

β) να ρωτάει με κατάλληλο μήνυμα ένα δεύτερο αριθμό και να τον αποθηκεύει σε μια μεταβλητή με τον όνομα Α2

γ) να υπολογίζει το γινόμενο των δύο αριθμών και να εκχωρεί τον αριθμό αυτό σε μια μεταβλητή με όνομα Γ.

δ) τέλος να βγάζει ένα παράθυρο όπου να εμφανίζεται μήνυμα:
το γινόμενο των αριθμών είναι Γ. (στη θέση Γ θα εμφανιστεί η τιμή του)

16η-17η Διδακτική ώρα: Δομή επιλογής if

ΣΤΟΧΟΙ

Στόχοι του μαθήματος είναι οι μαθητές να:

- περιγράφουν τις διάφορες μορφές της δομής επιλογής
- επιλέγουν την κατάλληλη μορφή της δομής επιλογής στα προγράμματα που αναπτύσσουν

Σχέδιο μαθήματος

1η διδακτική ώρα

Ο διδάσκων αντιγράφει το πρώτο φύλλο εργασίας σε ένα αρχείο doc και το μοιράζει στους υπολογιστές, ώστε να είναι διαθέσιμο στους μαθητές, σε ηλεκτρονική μορφή.

Το αρχείο περιέχει οδηγίες στους μαθητές για να δουλέψουν. Ενδεικτικές λύσεις:

```
button 'βουνο2',->
  await read ' ποιο είναι το ψηλότερο βουνό της Ελλάδας', defer x
  if x is 'Όλυμπος'
    wear 'mountain'
  else
   wear 'wrong'

button 'βαθμολογία', ->
  await read 'βαθμός α τετραμήνου', defer t1
  await read 'βαθμός β τετραμήνου', defer t2
  await read 'βαθμός εξετάσεων', defer g
  V=(t1+t2+g)/3
  if V <10
    write 'πρέπει να ξαναδώσεις εξετάσεις'
    wear 'sad'
  else
    write 'πέρασες το μάθημα'
    wear 'congratulations'
```

2η διδακτική ώρα

Μοιράζεται το φύλλο εργασίας. Ενδεικτικές λύσεις:

```
button 'πρωτοβάθμια', ->
  await read 'α', defer a
  await read 'β', defer b
  write 'η εξίσωση '+ a + ' * x = ' + b
  if a is 0
    if b is 0
      write ' η εξίσωση είναι αόριστη'
    else
      write ' η εξίσωση είναι αδύνατη'
    else
      x= b/a
      write 'η εξίσωση έχει λύση ' + x
```

```
button 'πρωτοβαθμια2', ->
  await read 'α', defer a
  await read 'β', defer b
  if a is 0 and b is 0
    write 'η εξίσωση είναι αόριστη'
  else if a is 0 and b isnt 0
    write 'η εξίσωση είναι αδύνατη'
  else
    write 'η εξίσωση έχει λύση' + b / a
```

Φύλλο εργασίας (1) – Η εντολή if

⌨ Αντιγράψτε τον παρακάτω κώδικα στο Pencil Code, κρατώντας την ίδια σύνταξη.

```
await read 'πόσο χρονών είσαι', defer x
if x > 17
  write 'είσαι ενήλικας'
else
  write 'είσαι παιδί'
```

⌨ Τρέξτε το πρόγραμμα 2-3 φορές δίνοντας ως απάντηση, διαφορετικές ηλικίες. Βάλτε τον προηγούμενο κώδικα σε ένα κουμπί:

```
button 'ηλικία', ->
  await read 'πόσο χρονών είσαι', defer x
  if x > 17
    write 'είσαι ενήλικας›
  else
    write 'είσαι παιδί'
```

☞ Αντιγράψτε στο ίδιο αρχείο τον παρακάτω κώδικα:

```
button 'ρίξε το ζάρι', ->
  x = random [1..6]
  write 'το ζάρι έδειξε' + x
  if x is 1
    write 'θα δουλέψω στο Pencil Code.'
    wear 'pencil'
  else if x is 2
    write 'θα διαβάσω Μαθηματικά!'
    wear 'math'
  else if x is 3
    write 'θα διαβάσω Λογοτεχνία.'
    wear 'book'
  else if x is 4
    write 'θα κάνω γυμναστική!'
    wear 'gym'
  else if x is 5
    write 'θα δω ταινία'
    wear 'cinema'
  else
    write 'θα πάω βόλτα'
    wear 'walk'
```

⌨ Πατήστε το κουμπί ρίξε το ζάρι

☞ Προσπαθήστε να φτιάξετε ένα κουμπί που θα εμφανίζει την ερώτηση: Ποιο είναι το ψηλότερο βουνό της Ελλάδας; Ο υπολογιστής θα ελέγχει την απάντηση που θα δώσει ο χρήστης. Αν απαντήσει σωστά, η χελώνα θα ντυθεί "mountain", αλλιώς θα ντυθεί "wrong".

☞ Προσπαθήστε να φτιάξετε να κουμπί που θα εμφανίζει ερωτήσεις ώστε ο χρήστης να δίνει τον βαθμό του στα Μαθηματικά στο α' τετράμηνο, στο β' τετράμηνο και στις εξετάσεις. Αν ο μέσος όρος είναι μεγαλύτερος του 10, θα γράφεται το κείμενο "Πέρασες το μάθημα", αλλιώς θα γράφεται το κείμενο «Πρέπει να ξαναδώσεις εξετάσεις».

Σχολείο ..... Ημερομηνία.....

Ονοματεπώνυμο..... Τμήμα ..... Αρ. Καταλόγου .....

Φύλλο εργασίας (2) – Η εντολή if

Έστω ο παρακάτω κώδικας στο Pencil Code. ΧΩΡΙΣ να τον αντιγράψετε στο περιβάλλον του Pencil Code, συμπληρώστε στον παρακάτω πίνακα, τι μήνυμα θα εμφανιστεί σε κάθε μια από τις παρακάτω περιπτώσεις:

```
button 'μάθημα',->
  await read ' πόσο είναι ο μέσος όρος?', defer m
  await read ' πόσο είναι ο βαθμός στην Ιστορία ?', defer h
  if m < 10 and h < 10
    write 'έμεινες στην ίδια τάξη'
  else if m > 10 and h > 10
    write 'πέρασες'
  else
    write 'ρώτα τον καθηγητή σου για το αποτέλεσμα'
```

Μέσος όρος	Βαθμός στην Ιστορία	Μήνυμα
16	17	
12	8	
9	12	
9	9	

Έστω ο παρακάτω κώδικας στο Pencil Code. ΧΩΡΙΣ να τον αντιγράψετε στο περιβάλλον του Pencil Code, συμπληρώστε στον παρακάτω πίνακα, ποιο θα είναι το κόστος μεταφοράς ενός οχήματος με το καράβι σε κάθε μια από τις παρακάτω περιπτώσεις:

```
button 'karavi',->
  await read 'τι εποχή έχουμε?', defer s
  await read 'βάρος οχήματος', defer v
  if s is 'Καλοκαίρι'
    if v > 1000
      kostos = 100
    else
      kostos = 80
  else
    if v > 1000
      kostos = 70
  else
    kostos = 50
  write 'το κόστος μεταφοράς θα είναι ' + kostos + ' ευρώ'
```

Εποχή	Βάρος	Κόστος
Καλοκαίρι	700	
Χειμώνας	1300	
Καλοκαίρι	1200	
Άνοιξη	600	

☞ Προσπαθήστε να φτιάξετε ένα κουμπί που όταν το πατάμε, θα μας ζητάει να του δώσουμε τους συντελεστές της πρωτοβάθμιας εξίσωσης $ax=b$ και να μας δίνει τη λύση της.
(θυμόμαστε από τα Μαθηματικά: όταν η εξίσωση είναι της μορφής $0x=0$ η εξίσωση είναι ταυτότητα, όταν η εξίσωση έχει $α=0$ και $β\neq0$ η εξίσωση είναι αδύνατη, σε κάθε άλλη περίπτωση η εξίσωση έχει λύση $x= β/α$)

18η – 20η Διδακτική ώρα: Επαναληπτική Δραστηριότητα

Μοιράζεται το φύλλο εργασίας στους μαθητές με μισοψημένους κώδικες. Οι μαθητές καλούνται να συνεργαστούν σε ομάδες των 2- 3 ατόμων ώστε να δημιουργήσουν το παιχνίδι Λαβύρινθος.

Οι μαθητές ξεκινούν φτιάχνοντας τον Λαβύρινθο σε μια απλή μορφή και σταδιακά, χρησιμοποιώντας τη φαντασία τους, τον βελτιώνουν. Ο διδάσκων υποστηρίζει τη διαδικασία.

Ενδεικτική λύση:

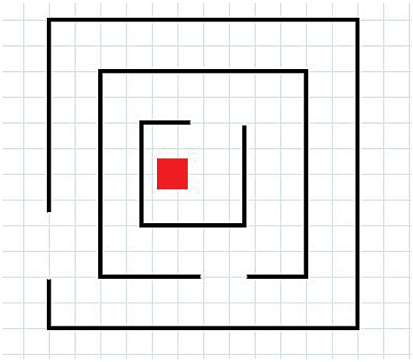
```
speed Infinity
pen black, 4
jumpto -200, -175
for [1..4]
  fd 300
  rt 90
fd 50
pen erase, 4
fd 60
rt 90
fd 50
lt 90
bk 60
pen black, 4
for [1..4]
  fd 200
  rt 90
rt 90
fd 100
pen erase, 4
fd 40
pu()
bk 100
lt 90
fd 50
pen black, 4
for [1..4]
  fd 100
  rt 90
fd 100
rt 90
fd 50
pen erase, 4
fd 50
home()
jumpto 200, 150
b = new Date
```

```

t = new Turtle red
t.jumpto -80, -30
t.wear 'cheese'
t.grow 0.2

write 'ΠΗΓΑΙΝΕ ΣΤΟΝ ΘΗΣΑΥΡΟ'
pause 2
wear  'mouse'
grow 0.1

forever  ->
  turnto lastmouse
  fd 3
  if touches t
    write 'ΚΕΡΔΙΣΕΣ '
    f = new Date
    time = (f - b) / 1000
    write "ο χρόνος σου ήταν " +  time + " δευτερόλεπτα !"
    t.hide()
    stop()
  if touches black
    jumpto 200, 150

```

Φύλλο εργασίας – Λαβύρινθος

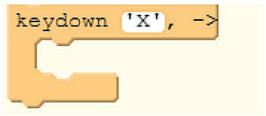
1) Χρησιμοποιώντας εντολές σχεδίασης, φτιάξτε έναν Λαβύρινθο.

2) Βάλτε σε κάποιο σημείο μέσα στο Λαβύρινθο σας, τον «θησαυρό», για παράδειγμα ένα τετράγωνο με διαφορετικό χρώμα από το χρώμα των τοιχωμάτων του Λαβύρινθου. όπως στην παρακάτω εικόνα.



3) Φέρτε τη χελώνα σας σε κάποιο σημείο εκτός Λαβυρίνθου.

4) Χρησιμοποιώντας τη συνάρτηση `keydown` δώστε κατάλληλες εντολές στα πλήκτρα που επιθυμείτε ώστε πατώντας τα, η χελώνα να κινείται μπροστά, δεξιά, αριστερά και πίσω.



5) Στο τέλος «ενεργοποιήστε» το παιχνίδι με τις παρακάτω εντολές, τροποποιημένες κατάλληλα για τον δικό σας Λαβύρινθο:

```
forever ->
  if touches red
    write 'ΚΕΡΔΙΣΕΣ'
    stop()
  if touches black
    home()
```

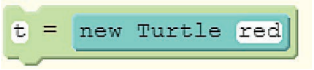
Παραλλαγή 1

Μπορείτε να κινείτε τη χελώνα σας με το ποντίκι: Μετά την `forever` προσθέτουμε τις εντολές

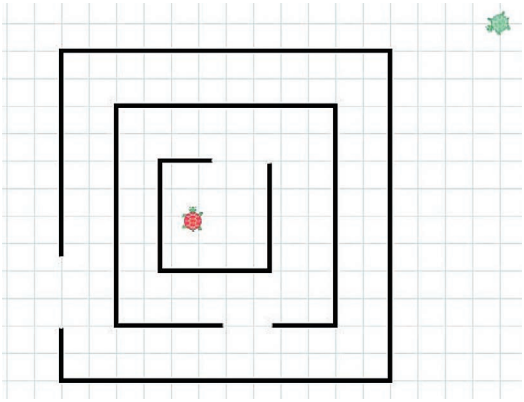
```
turnto lastmouse
fd 3
```

Παραλλαγή2

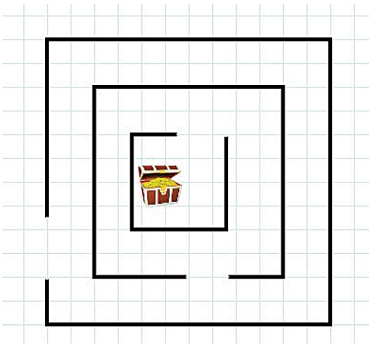
- Δημιουργήστε μια δεύτερη χελώνα με το πλακίδιο:



- «κρύψτε» τη μέσα στο Λαβύρινθο (για να δώσουμε εντολές στη χελώνα με όνομα t, γράφουμε `t.fd 40, t.rt 90, t.pu()` κ.ο.κ.)



- «Ντύστε» τη χελώνα σας να μοιάζει με θησαυρό με την εντολή `t.wear 'treasure'` και μικρύνετέ την με την εντολή `t.grow 0.2`



- Αντικαταστήστε την εντολή `if touches red` με την εντολή `if touches t`

Παραλλαγή 3

Δοκιμάστε να «ντύσετε» διαφορετικά τις 2 χελώνες. (Π.χ. γάτα- ποντίκι, ποντίκι-τυρί)

Παραλλαγή 4

Μπορούμε να μετράμε τον χρόνο μέχρι να βρεθεί ο θησαυρός.

Γράφουμε την εντολή `s = newDate` ΠΡΙΝ τη `forever`, για να αποθηκευτεί την ώρα έναρξης του παιχνιδιού και προσθέτουμε τις εντολές

```
e= newDate
time= (e-s)/1000
write "κέρδισες σε "+ time + "δευτερόλεπτα!"
```

ΜΕΤΑ την εντολή `if touches t`

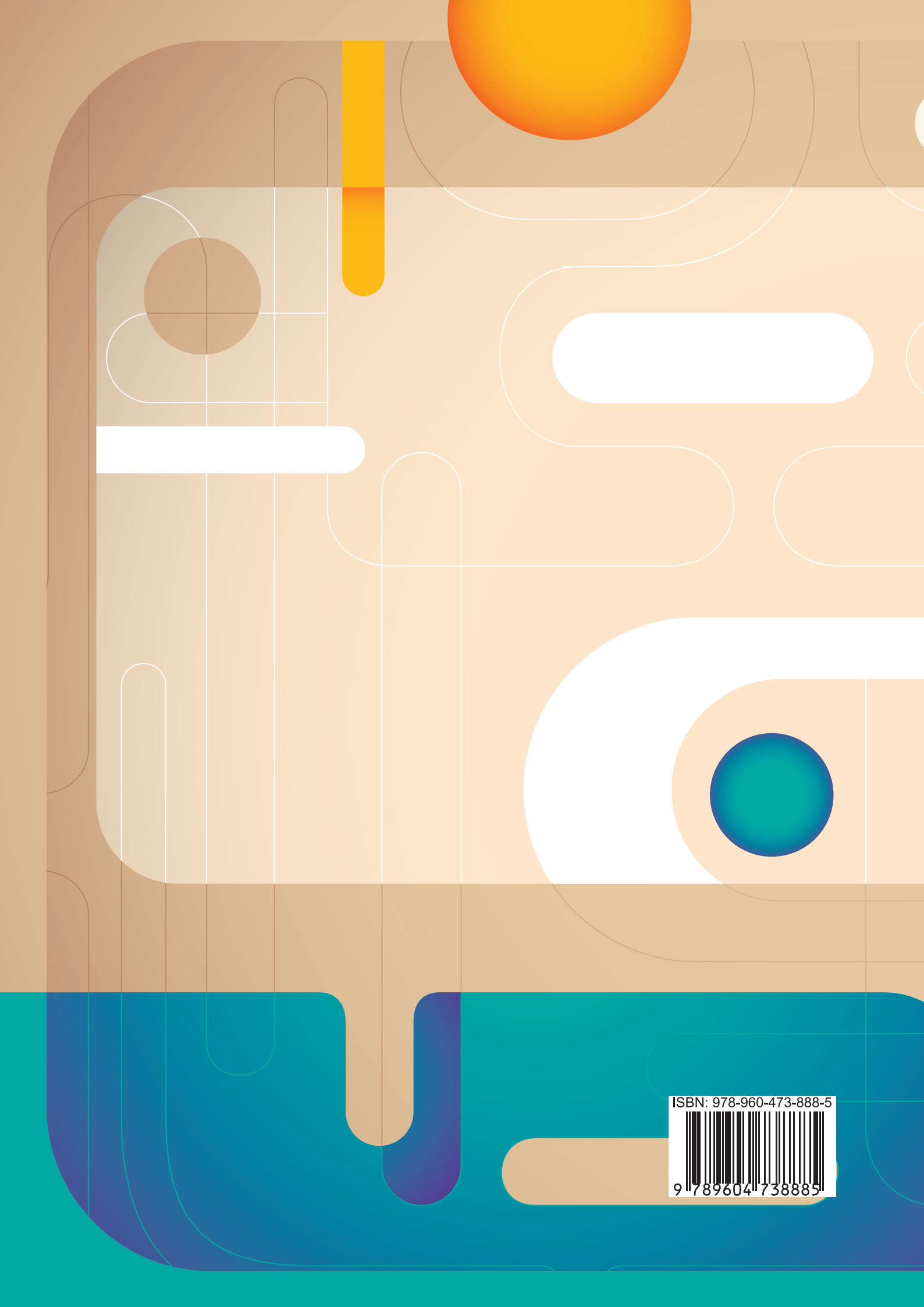
Παραλλαγή 5

Δημιουργήστε περισσότερους Λαβύρινθους- πίστες. Κάθε φορά που η χελώνα, βρίσκει τον θησαυρό, αρχίζει το ψάξιμο του θησαυρού σε άλλο Λαβύρινθο (πιθανά πιο δύσκολο)

Πρόταση υλοποίησης: Η κάθε πίστα-είναι μια συνάρτηση. Η νέα πίστα καλείται από την προηγούμενη πίστα –συνάρτηση, σε περίπτωση επιτυχίας.

Ευρετήριο εντολών

`alert`, 15
`append`, 14
`box`, 18
`click`, 29
`cs()`, 19
`dot`, 18
`fd`, 17
`fill`, 19
`hide()`, 19
`home()`, 18
`keydown`, 30
`label`, 14
`listen`, 29
`lt`, 17
`moveto`, 18
`movexy`, 18
`pd()`, 19
`pen`, 17
`pen erase`, 18
`pen off`, 19
`pen on`, 19
`play`, 14
`pu()`, 19
`read`, 29
`readnum`, 29
`rt`, 17
`rt 180, 100`, 18
`say`, 14
`show()`, 19
`speed`, 19
`Speed Infinity`, 19
`tone`, 14
`type`, 14
`typeline`, 14
`write`, 14


ISBN: 978-960-473-888-5