

Η Απάντηση

1

18 Αυγούστου 2016

11:43

Στο κεφάλαιο αυτό θα γνωρίσουμε τα βασικά είδη εντολών που θα επιτρέπουν στα προγράμματά μας να *αλληλεπιδρούν* με το χρήστη, δηλαδή να του εμφανίζουν μηνύματα στην οθόνη και να του ζητούν να εισάγει τιμές από το πληκτρολόγιο. Θα κάνουμε επίσης μια πρώτη γνωριμία με ορισμένες θεμελιώδεις έννοιες τις οποίες θα συναντήσουμε συχνά και στα επόμενα κεφάλαια. Κυρίως όμως θα μάθουμε την Απάντηση για τη Ζωή, το Σύμπαν και τα Πάντα!

Πληκτρολογήστε τις εντολές που ακολουθούν. Ίσως να μην είναι άμεσα κατανοητό πώς λειτουργούν, αλλά αυτό είναι φυσιολογικό και η βαθύτερη κατανόηση θα έρθει με την εξάσκηση και τον πειραματισμό. Σκοπός είναι να εξοικειωθείτε με τη γλώσσα, τις λεπτομέρειες της σύνταξής της και τα βασικά της χαρακτηριστικά. Πειραματιστείτε ελεύθερα και μην απογοητεύεστε αν κάνετε λάθη: η προσπάθεια να τα εντοπίσετε και να τα εξαλείψετε είναι κι αυτή κομμάτι του προγραμματισμού.

Έννοιες: είσοδος, έξοδος, μεταβλητές, δομή επιλογής.

Ο Douglas Adams, στο βιβλίο *The Hitchhiker's Guide to the Galaxy*, γράφει για μια υπερεφυγή φυλή η οποία, κουρασμένη από τις διαφονίες σχετικά με το νόημα της ζωής, αποφάσισε να φτιάξει έναν υπολογιστή που θα έδινε οριστικά την απάντηση που αναζητούσαν. Ο υπολογιστής χρειάστηκε 7.5 εκατομμύρια χρόνια για να υπολογίσει και να ελέγξει την Απάντηση για τη Ζωή, το Σύμπαν και τα Πάντα. Η Απάντηση ήταν... σαράντα δύο.

Ο υπολογιστής λέγονταν *Deer Thought* και μπορούμε να φτιάξουμε κι εμείς ένα πρόγραμμα σαν το δικό του. Επειδή δεν διαθέτουμε ανάλογη υπολογιστική ισχύ κι επειδή γνωρίζουμε ήδη την Απάντηση, θα κλέψουμε λίγο: το πρόγραμμά μας δεν θα υπολογίζει την Απάντηση, αλλά μόνο θα την ανακοινώνει στο χρήστη.

Πες Τουλάχιστον Μια Καλημέρα

```
1 # χαιρετισμός  
2 print("Καλημέρα.")
```

Ο,τιδήποτε ακολουθεί το σύμβολο # είναι ένα σχόλιο. Δεν αφορά τον υπολογιστή και αγνοείται κατά την εκτέλεση του προγράμματος. Απευθύνεται σε εκείνους που διαβάζουν τον κώδικα και (πρέπει να) χρησιμεύει στην καλύτερη κατανόησή του.

Για να εμφανίσουμε ένα μήνυμα στην οθόνη χρησιμοποιούμε την `print()`, δίνοντας ανάμεσα στις παρενθέσεις το μήνυμα που θα εμφανιστεί.

Είναι εκπληκτικό ότι σχεδόν σε κάθε βιβλίο για τον προγραμματισμό, αυτό είναι το πρώτο παράδειγμα που συναντάει κανείς: πως να κάνει τον υπολογιστή του να πει μια καλημέρα.

Πες και την Απάντηση

Στη συνέχεια, ο Deep Thought μπορεί να εμφανίσει την Απάντηση.

```
# εμφάνιση της Απάντησης
print("Η Απάντηση είναι... 42")
```

Το ίδιο αποτέλεσμα μπορεί να επιτευχθεί και με διαφορετικό τρόπο.

```
3 # ορισμός και εμφάνιση της Απάντησης
4 answer = 42
5 print("Η Απάντηση είναι...", answer)
```

answer/src/answer.1.py

Ενώ ο χρήστης βλέπει το ίδιο μήνυμα στην οθόνη, οι δύο `print()` δεν κάνουν το ίδιο πράγμα. Η πρώτη εμφανίζει πάντα την τιμή 42, ενώ η δεύτερη εμφανίζει την τιμή `answer`, όποια κι αν είναι αυτή.

Η `answer` είναι μια μεταβλητή. Μπορούμε να πούμε ότι δίνουμε στην τιμή 42 το όνομα `answer`. Μπορούμε επίσης να πούμε ότι δίνουμε στο όνομα `answer` την τιμή 42. Και οι δύο περιγραφές είναι ορθές, είναι απλά θέμα οπτικής γωνίας. Σημασία έχει ότι οι μεταβλητές επιτρέπουν στα προγράμματά μας να διατηρούν, να “θυμούνται” τις τιμές που είναι σημαντικές. Όταν συσχετίζουμε μια τιμή μ’ ένα όνομα (όπως κάνουμε εδώ με το όνομα `answer` και την τιμή 42) μπορούμε ν’ αναφερθούμε σε αυτή κι αργότερα, διαφορετικά δεν έχουμε τρόπο ανάκτησής της.

Η εντολή `answer = 42` δεν διατυπώνει κάτι που πρέπει να ισχύει για πάντα, είναι απλά μια εντολή που θα αντιστοιχίσει το όνομα `answer` με την τιμή 42 όταν έρθει η σειρά της να εκτελεστεί. Θα μπορούσαμε με μια αντίστοιχη εντολή στη συνέχεια ν’ αλλάξουμε την τιμή της μεταβλητής `answer`, δηλαδή να συσχετίσουμε το όνομα με μια νέα τιμή (αν και αυτό δεν θα χρειαστεί για την `answer`, γιατί η Απάντηση είναι μία).

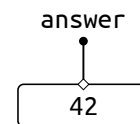
Σημειώστε ότι ο χρήστης δε γνωρίζει τίποτα για το όνομα `answer` ή την αντίστοιχη τιμή. Για την ακρίβεια δε γνωρίζει καν για την ύπαρξη της μεταβλητής. Ο χρήστης γνωρίζει μόνο ό,τι του εμφανίζει το πρόγραμμα με εντολές εξόδου.

Για Περίμενε Λιγάκι

Πώς γίνεται να προσθέσουμε λίγο σασπένς; Μπορούμε να εισάγουμε μια καθυστέρηση λίγο πριν την ανακοίνωση της Απάντησης;

Επειδή οι βασικές εντολές της Python δεν προσφέρουν κάποιο μηχανισμό καθυστέρησης, θα χρησιμοποιήσουμε μια βιβλιοθήκη, η οποία παρέχει τη λειτουργικότητα που μας χρειάζεται. Οι βιβλιοθήκες πε-

Μπορούμε να εμφανίσουμε με την `print()` πολλές τιμές, αρκεί να τις χωρίσουμε με κόμμα. Ανάμεσά στις τιμές εμφανίζεται ένα κενό, όμως αυτή η προκαθορισμένη συμπεριφορά είναι δυνατόν ν’ αλλάξει.



Σχήμα 1.1: Η μεταβλητή `answer` έχει την τιμή 42. Εναλλακτικά, θα λέγαμε ότι στην τιμή 42 έχει δοθεί το όνομα `answer`.

ριέχουν έτοιμο κώδικα και τις συναντάμε στις περισσότερες γλώσσες προγραμματισμού: είναι συλλογές από μικρά προγράμματα που μπορούμε να χρησιμοποιήσουμε στα προγράμματά μας.

```

1 import time
2 # χαιρετισμός
3 print("Καλημέρα.")
4 # καθυστέρηση
5 time.sleep(3)
6 # ορισμός και εμφάνιση της Απάντησης
7 answer = 42
8 print("Η Απάντηση είναι...", answer)

```

answer/src/answer.2.py

Αν θέλουμε να υπάρχει πραγματικό σασπένς και να είμαστε πιστοί στο βιβλίο του Douglas Adams, θα πρέπει να εισάγουμε μια καθυστέρηση της τάξης των 7.5 εκατομμυρίων ετών (σε δευτερόλεπτα). Ίσως δεν θα έπρεπε να δοκιμάσετε τον κώδικα που ακολουθεί, μπορεί να βαρεθείτε λιγάκι μέχρι να εμφανιστεί η Απάντηση.

```

4 # καθυστέρηση
5 wait = 7500000 * 365 * 24 * 60 * 60
6 time.sleep(wait)
7 # ορισμός και εμφάνιση της Απάντησης
8 answer = 42
9 print("Η Απάντηση είναι...", answer)

```

answer/src/answer.3.py

Η τιμή της μεταβλητής `wait` αντιστοιχεί στα δευτερόλεπτα καθυστέρησης. Σε αντίθεση με την `answer`, η τιμή της `wait` δεν καθορίζεται απευθείας αλλά προκύπτει από τον υπολογισμό της τιμής του γινομένου $7500000 * 365 * 24 * 60 * 60$.

Πιάσαμε την Κουβέντα

Μέχρι στιγμής ο Deep Thought είναι λίγο απρόσωπος: απλά ανακοινώνει την Απάντηση. Ο χρήστης δεν θα αλληλεπιδρά με το πρόγραμμα;

Η ροή της πληροφορίας ανάμεσα στο πρόγραμμα και τον χρήστη είναι μονόδρομη. Στις περισσότερες περιπτώσεις ένα πρόγραμμα δεν περιορίζεται στην εμφάνιση μηνυμάτων, αλλά χρειάζεται και να κάνει ερωτήσεις στον χρήστη, να του ζητήσει τιμές.

Θα προγραμματίσουμε τον Deep Thought έτσι ώστε να ζητάει το όνομα του χρήστη και να τον καλημερίζει κατάλληλα. Έτσι θα έχουν ένα στοιχειώδη διάλογο, πριν από την ανακοίνωση της Απάντησης.

```

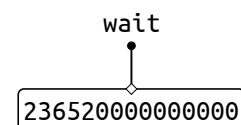
2 # είσοδος ονόματος
3 print("Πώς σε λένε;")
4 name = input()
5 # χαιρετισμός
6 print("Καλημέρα", name)

```

Για να χρησιμοποιήσουμε μια βιβλιοθήκη θα πρέπει πρώτα να την εισάγουμε (**import**). Εδώ θα εισάγουμε τη βιβλιοθήκη `time` (που περιέχει έτοιμα υποπρογράμματα για τη διαχείριση του χρόνου) και θα χρησιμοποιήσουμε την `sleep()`, που καθυστερεί την εκτέλεση του προγράμματος για όσα δευτερόλεπτα καθορίσουμε.

Το σύμβολο `*` αντιστοιχεί στην πράξη του πολλαπλασιασμού. Σε άλλες εκφράσεις μπορείτε επίσης να χρησιμοποιήσετε τα κλασικά `+` και `-`, καθώς επίσης και το `/` για τη διαίρεση, το `//` για το πηλίκο της ακέραιας διαίρεσης και το `%` για το υπόλοιπο της ακέραιας διαίρεσης.

Αν θέλετε να διακόψετε την εκτέλεση ενός προγράμματος, χρησιμοποιήστε τον συνδυασμό πλήκτρων `Ctrl + C`.



Σχήμα 1.2: Η τιμή της μεταβλητής `wait` προέκυψε από τον υπολογισμό της τιμής μας αριθμητικής έκφρασης.

Για να ζητήσουμε μια τιμή από το πληκτρολόγιο χρησιμοποιούμε την `input()`, η οποία επιστρέφει μια *αλφαριθμητική* τιμή: το κείμενο που πληκτρολογήθηκε από τον χρήστη. Εδώ, στην τιμή αυτή δίνεται το όνομα `name`.

answer/src/answer.4.py

Εναλλακτικά:

```
# είσοδος ονόματος
name = input("Πώς σε λένε; ")
# χαιρετισμός
print("Καλημέρα", name)
```

Η τιμή της μεταβλητής `name` είναι το κείμενο που πληκτρολογεί ο χρήστης και φυσικά δεν είναι γνωστό εκ των προτέρων. Ο `Deep Thought` χρησιμοποιεί την τιμή της μεταβλητής `name` για να καλημερίσει τον χρήστη, όποια κι αν είναι αυτή η τιμή.

Έχεις Ώρα;

Αν ένας χρήστης εκτελέσει το βράδι το πρόγραμμα που έχουμε φτιάξει για τον `Deep Thought`, τότε θα δυσκολευτεί να πιστέψει ότι πρόκειται για έναν υπερυπολογιστή που γνωρίζει την Απάντηση για τη Ζωή, το Σύμπαν και τα Πάντα, αφού θα τον χαιρετίσει βραδιάτικα με ένα "Καλημέρα".

Το πρόγραμμα πρέπει να γνωρίζει την ώρα της ημέρας. Ευτυχώς ο υπολογιστής μας ξέρει τι ώρα είναι, οπότε απλά θα τον ρωτήσουμε.

```
5 # ανάκτηση τοπικής ώρας συστήματος
6 hour = time.localtime().tm_hour
```

Εδώ όμως το ουσιαστικό πρόβλημα είναι ότι οι εντολές που θα πρέπει να εκτελέσει το πρόγραμμά μας δεν είναι πάντα οι ίδιες. Θα πρέπει να προγραμματίσουμε τον `Deep Thought` έτσι ώστε να ελέγχει την ώρα της ημέρας και να εμφανίζει διαφορετικό μήνυμα ανάλογα με το αποτέλεσμα του ελέγχου.

```
7 # εμφάνιση χαιρετισμού ανάλογα με την ώρα
8 if hour < 14:
9     print("Καλημέρα", name)
10 else:
11     print("Καλησπέρα", name)
```

answer/src/answer.5.py

Ο κώδικας ελέγχει την συνθήκη `hour < 14`, δηλαδή αν η τρέχουσα ώρα είναι πριν τις 2 το μεσημέρι. Αν η συνθήκη ισχύει τότε εμφανίζει το μήνυμα "Καλημέρα", αλλιώς το μήνυμα "Καλησπέρα". Με άλλα λόγια, το πρόγραμμα επιλέγει την συμπεριφορά του ανάλογα με τις συνθήκες που επικρατούν την ώρα της εκτέλεσής του. Αυτό το νέο προγραμματιστικό εργαλείο ονομάζεται *δομή επιλογής*.

Μια πρακτική συμβουλή: θα πρέπει πάντα να εκτελούμε τα προγράμματά μας, ελέγχοντας ότι η συμπεριφορά τους είναι ορθή. Εδώ το πρόγραμμά μας περιέχει μια δομή επιλογής με δύο πιθανές περιπτώσεις και θα πρέπει να ελέγξουμε ότι λειτουργεί σωστά και στις δύο. Αν το εκτελέσουμε πρώι και μας χαιρετίσει μ' ένα "Καλημέρα", δεν είναι καλή ιδέα να περιμένουμε μέχρι το βράδι για να ελέγξουμε αν αλλάζει η συμπεριφορά του.

Η εμφάνιση ενός μηνύματος πριν την ανάγνωση τιμών είναι τόσο συνηθισμένη που η `input()` μπορεί να δεχθεί ανάμεσα στις παρενθέσεις το μήνυμα που πρέπει να εμφανιστεί.

Η `localtime()` της βιβλιοθήκης `time` επιστρέφει πληροφορίες για την ημερομηνία και την ώρα του συστήματος. Οι συντακτικές λεπτομέρειες δεν έχουν σημασία αυτή την στιγμή.

Η `if` συνοδεύεται από μια *συνθήκη*, η οποία ελέγχεται κατά την εκτέλεση του προγράμματος και μπορεί να είναι αληθής (`True`) ή ψευδής (`False`).

Οι εντολές που ακολουθούν την `if` είναι στοιχισμένες δεξιάτερα από αυτή. Η στοιχισμένη επιτυχάνεται εισάγοντας κενά πριν από τις εντολές, τα οποία υποδηλώνουν ότι αυτές οι εντολές θα εκτελεστούν μόνο αν η συνθήκη είναι αληθής. Παρομοίως, οι εντολές που ακολουθούν την `else` είναι στοιχισμένες δεξιάτερα και θα εκτελεστούν μόνο εφόσον η συνθήκη είναι ψευδής.

Το σύμβολο `<` χρησιμοποιείται για την σύγκριση τιμών, όπως επίσης και τα `<=`, `>` και `>=`. Για να ελέγξουμε αν δύο τιμές είναι ίσες χρησιμοποιούμε το `==` και για να ελέγξουμε αν είναι διαφορετικές το `!=`.

Μην παραλείπετε το σύμβολο `:` μετά την συνθήκη της `if` και την `else`.

Η λύση είναι να τροποποιήσουμε *προσωρινά* το πρόγραμμά μας και να καθορίσουμε εμείς την ώρα, αντί να χρησιμοποιήσουμε την ώρα του συστήματος. Καθορίζοντας ότι είναι 10 το πρωί, το πρόγραμμα θα εμφανίσει το πρώτο μήνυμα.

```
# ΓΙΑ ΕΛΕΓΧΟ: ορισμός ώρας  
hour = 10
```

Όταν αλλάξουμε την τιμή της μεταβλητής `hour` σε 20 τότε, εκτός συγκλονιστικού απροόπτου, θα δούμε το δεύτερο μήνυμα.

```
# ΓΙΑ ΕΛΕΓΧΟ: ορισμός ώρας  
hour = 20
```

Πλήρες Τελικό Πρόγραμμα

```
1 import time  
2 # είσοδος ονόματος  
3 print("Πώς σε λένε;")  
4 name = input()  
5 # ανάκτηση τοπικής ώρας συστήματος  
6 hour = time.localtime().tm_hour  
7 # εμφάνιση χαιρετισμού ανάλογα με την ώρα  
8 if hour < 14:  
9     print("Καλημέρα", name)  
10 else:  
11     print("Καλησπέρα", name)  
12 # καθυστέρηση  
13 wait = 3  
14 time.sleep(wait)  
15 # ορισμός και εμφάνιση της Απάντησης  
16 answer = 42  
17 print("Η Απάντηση είναι...", answer)
```

[answer/src/answer.final.py](#)

Και Τώρα Τι;

Δεν μάθατε και λίγα για αρχή. Ίσως μάλιστα να αισθάνεστε λίγο όπως ο Παπαγάλος, του Ζαχαρία Παπαντωνίου.

Σαν έμαθε τη λέξη *καλησπέρα*
ο παπαγάλος, είπε ξαφνικά:
«Είμαι σοφός, γνωρίζω ελληνικά.
Τι κάθομαι εδώ πέρα;»

Όπου *ελληνικά*, αντικαταστήστε με *Pythonia*. Όμως είναι ανάγκη να εμπεδώσετε τις έννοιες που συναντήσατε χρησιμοποιώντας τις σε νέα προβλήματα (δηλαδή λύνοντας τις ασκήσεις που ακολουθούν). Διαφορετικά, θα αισθανθείτε σίγουρα όπως ο Παπαγάλος:

«Κυρ παπαγάλε, θα 'χομε την τύχη
ν' ακούσωμε τις λες και πάρα πέρα;»
Ο παπαγάλος βήχει, ξεροβήχει...
μα τι να πει; Ξανάπε: «καλησπέρα».

Τροποποιήσεις – Επεκτάσεις

Στο πρόγραμμα που κατασκευάσαμε χρησιμοποιήσαμε την `input()` για να ζητήσουμε το όνομα του χρήστη. Η τιμή που επιστρέφει η `input()` είναι αλφαριθμητική, έχει δηλαδή την μορφή κειμένου. Αυτό δεν είναι πρόβλημα όταν ζητάμε το όνομα του χρήστη γιατί κι αυτό έχει την μορφή κειμένου. Όταν όμως θέλουμε να ζητήσουμε από το χρήστη έναν ακέραιο ή έναν πραγματικό αριθμό (κάτι που θα χρειαστεί στις ασκήσεις που ακολουθούν), τότε θα πρέπει να μετατρέψουμε την αλφαριθμητική τιμή που επιστρέφει η `input()` σε αριθμητική, με τον τρόπο που φαίνεται παρακάτω:

```
# το κείμενο που πληκτρολογεί ο χρήστης
# μετατρέπεται σε ακέραιο αριθμό και
# αποθηκεύεται στην μεταβλητή number
number = int(input())
```

```
# το κείμενο που πληκτρολογεί ο χρήστης
# μετατρέπεται σε πραγματικό αριθμό και
# αποθηκεύεται στην μεταβλητή number
number = float(input())
```

Οι `int()` και `float()` χρησιμοποιούνται γενικότερα όταν χρειάζεται να μετατρέψουμε μια τιμή σε ακέραια ή πραγματική. Για παράδειγμα, η εκφραση `int(3.14159)` έχει την ακέραια τιμή 3.

Κάτι άλλο που θα φανεί χρήσιμο σε κάποιες από τις ασκήσεις είναι ο υπολογισμός του ηλίκου και του υπολοίπου της *ακέραιας* διαίρεσης δύο αριθμών. Αυτό γίνεται όπως στο παράδειγμα:

```
# ηλίκο ακέραιας διαίρεσης του number με το 10
q = number // 10
# υπόλοιπο ακέραιας διαίρεσης του number με το 2
r = number % 2
```

Τέλος, αν μετά από μια `print` δεν θέλετε ν' αλλάξει η γραμμή, ορίστε κατάλληλα την παράμετρο `end` της `print`. Δοκιμάστε το παρακάτω τμήμα κώδικα με και χωρίς την `end`, για να καταλάβετε τη διαφορά στον τρόπο λειτουργίας.

```
# μετά το μήνυμα τυπώνεται το περιεχόμενο της end
# (εδώ, ένα κενό ή τα αποσιωπητικά)
# και το επόμενο μήνυμα τυπώνεται στην ίδια γραμμή
print("Καλημέρα.", end = " ")
print("Η Απάντηση είναι", end = "... ")
print("42")
```

- 1.1 Μπορείτε να επεκτείνετε το πρόγραμμα του *Deep Thought* έτσι ώστε να ρωτά το χρήστη ποιο είναι το έτος γέννησής του και να υπολογίζει την ηλικία του. Για να το καταφέρετε αυτό, θα πρέπει το πρόγραμμά σας να γνωρίζει ποιο είναι το τρέχον έτος:

```
year = time.localtime().tm_year
```

Στη συνέχεια, μπορείτε απλά να εμφανίσετε στο χρήστη την ηλικία του (αν και πιθανότατα θα τη γνωρίζει) ή να χρησιμοποιήσετε την `if` για να εμφανίζετε διαφορετικό μήνυμα ανάλογα με την ηλικία. Θα μπορούσατε, για παράδειγμα, να μην εμφανίζετε την Απάντηση αν η ηλικία του χρήστη ξεπερνά κάποιο όριο.

[answer/exercises/answer-age.py](#)

- 1.2 Να τροποποιήσετε το πρόγραμμα του *Deep Thought* έτσι ώστε να ρωτάει τον χρήστη για την Απάντηση, αντί να την ανακοινώνει απευθείας.

Ποιά είναι η Απάντηση;

Το πρόγραμμα θα ελέγχει αν ο χρήστης πράγματι γνωρίζει την Απάντηση και θα του εμφανίζει ανάλογο μήνυμα.

Με το `==` μπορείτε να ελέγξετε αν δύο τιμές είναι ίσες.

Όταν ζητάτε από το χρήστη την Απάντηση, εκείνος δεν είναι υποχρεωμένος να πληκτρολογήσει έναν αριθμό. Έτσι, αν το πρόγραμμά σας χρησιμοποιεί την `int` για να μετατρέψει την απάντηση του χρήστη σε ακέραιο, έχετε υπόψη ότι αυτή η μετατροπή ίσως να μην είναι εφικτή και να οδηγήσει σε μήνυμα σφάλματος:

```
ValueError: invalid literal for int() with base 10
```

Μια απλή λύση είναι να θεωρήσετε ότι η Απάντηση είναι `"42"` αντί για `42`, δηλαδή κείμενο και όχι ακέραιος. Έτσι, θα μπορέσετε να την συγκρίνετε με την απάντηση του χρήστη, χωρίς να χρειαστεί να μετατρέψετε την απάντηση του χρήστη σε ακέραιο με την `int`.

[answer/exercises/answer-question.py](#)

Στη συνέχεια, να επεκτείνετε το πρόγραμμά σας έτσι ώστε να δίνεται μια δεύτερη ευκαιρία στο χρήστη, αν αυτός δεν γνωρίζει την Απάντηση. Πιθανώς, πριν ο χρήστης ερωτηθεί για δεύτερη φορά, το πρόγραμμα θα μπορούσε να του εμφανίζει κάποια υπόδειξη, π.χ. ότι η Απάντηση είναι το γινόμενο δύο διαδοχικών μονοψήφιων ακεραίων ή ότι είναι ο τέταρτος κατά σειρά θετικός ακέραιος που οι "γείτονές" του είναι πρώτοι αριθμοί.

[answer/exercises/answer-question-extended.py](#)

Ασκήσεις

- 1.3 Στη Σελήνη το βάρος ενός αντικειμένου είναι το $\frac{1}{6}$ του βάρους του στη Γη. Στην Αφροδίτη το βάρος ενός αντικειμένου είναι 0,9 φορές το βάρος του στη Γη. Στον Ήλιο το βάρος ενός αντικειμένου είναι 27,07 φορές το βάρος του στη Γη, αλλά όταν βρίσκεται κανείς εκεί η αύξηση του βάρους δεν είναι το βασικότερο πρόβλημα. Να γράψετε πρόγραμμα που θα ζητάει από το χρήστη το βάρος του στη Γη και θα εμφανίζει το βάρος του στη Σελήνη, την Αφροδίτη και τον Ήλιο.

[answer/exercises/weight.py](#)

- 1.4 Υπάρχουν πολλά παιχνίδια στα οποία κάποιος σας ζητά να σκεφτείτε έναν μυστικό αριθμό και, μετά από μερικές ερωτήσεις και υπολογισμούς, «μαντεύει» τον αριθμό που είχατε σκεφτεί. Ένα πολύ παλιό παράδειγμα βασίζεται σ' ένα θεώρημα του κινέζου Sun Tzu, που έζησε κάπου ανάμεσα στον 3^ο και τον 5^ο αιώνα. Ας υποθέσουμε ότι ο άγνωστος αριθμός είναι ο x . Αν ονομάσουμε α , β και γ τα υπόλοιπα της διαίρεσης του x με το 3, το 5 και το 7 αντίστοιχα, τότε για να βρούμε τον άγνωστο αριθμό θα πρέπει να υπολογίσουμε την τιμή της παράστασης $70\alpha + 21\beta + 15\gamma$ και στη συνέχεια να διαιρέσουμε με το 105. Το υπόλοιπο της διαίρεσης θα είναι ο μυστικός αριθμός. Να αναπτύξετε ένα πρόγραμμα το οποίο ζητάει από το χρήστη να σκεφτεί έναν μυστικό αριθμό από το 1 μέχρι και το 100. Στη συνέχεια, το πρόγραμμα ρωτάει το χρήστη ποιο είναι το υπόλοιπο της διαίρεσης αυτού του αριθμού με το 3, το 5 και το 7 και ανακοινώνει τον μυστικό αριθμό, αφήνοντας το χρήστη έκπληκτο με τις υπερφυσικές του ικανότητες.

[answer/exercises/suntzu.py](#)

- 1.5 Να γράψετε πρόγραμμα που θα ζητάει από το χρήστη το πλήθος των δευτερολέπτων που έχουν περάσει από τα μεσάνυχτα και θα εμφανίζει την τρέχουσα ώρα στη μορφή ώρες:λεπτά:δευτερόλεπτα. Για παράδειγμα, αν ο χρήστης καθορίσει ότι έχουν περάσει 42222 δευτερόλεπτα από τα μεσάνυχτα, το πρόγραμμα θα απαντά ότι η ώρα είναι 11:43:42.

Σημείωση: Αν οι ώρες, τα λεπτά ή τα δευτερόλεπτα είναι μονοψήφιοι αριθμοί τότε η ώρα δεν θα εμφανίζεται «όμορφα». Αν αυτό προσβάλλει την αισθητική σας και δεν μπορείτε να το ανεχτείτε, έχετε υπόψη ότι μπορείτε να εμφανίσετε έναν αριθμό ως διψήφιο, με ένα αρχικό μηδέν όταν είναι απαραίτητο, με τον εξής φρικτό τρόπο:

```
# έστω num ένας ακέραιος, πιθανώς μονοψήφιος
# έτσι εμφανίζεται με αρχικό μηδέν, αν χρειάζεται
print("{:02}".format(num))
```

[answer/exercises/clock.py](#)

- 1.6 Στο βιβλίο του Yacov Perelman με τίτλο *Figures for Fun: Stories and Conundrums* υπάρχει το εξής πρόβλημα: Ένας άνθρωπος αδειάζει σ' ένα τραπέζι ένα κουτί που περιέχει 48 σπέρτα και τα χωρίζει σε τρεις άνισους σωρούς. Παίρνει από τον πρώτο σωρό τόσα σπέρτα όσα υπάρχουν στο δεύτερο σωρό και τα προσθέτει στον δεύτερο σωρό. Στη συνέχεια, παίρνει από το δεύτερο σωρό τόσα σπέρτα όσα υπάρχουν στον τρίτο σωρό και τα προσθέτει στον τρίτο σωρό. Τέλος, παίρνει από τον τρίτο σωρό τόσα σπέρτα όσα υπάρχουν στον πρώτο σωρό και τα προσθέτει στον πρώτο σωρό. Μετά από αυτές τις μετακινήσεις, όλοι οι σωροί περιέχουν το ίδιο πλήθος σπέρτων. Πόσα σπέρτα περιείχε κάθε σωρός αρχικά;

Να γράψετε ένα πρόγραμμα που υπολογίζει τη λύση του προβλήματος, ακολουθώντας τη διαδικασία που περιγράφεται αντιστρόφως. Χρησιμοποιήστε τρεις μεταβλητές $h1$, $h2$ και $h3$ για το πλήθος των σπέρτων που περιέχει κάθε σωρός. Αρχικά, οι τρεις μεταβλητές θα έχουν την ίδια τιμή, αντιστοιχώντας στην τελική κατάσταση. Ακολουθήστε με την αντίστροφη σειρά τα βήματα που περιγράφονται, γράφοντας τις αντίστοιχες εντολές. Όταν τελειώσετε και εκτελέσετε το πρόγραμμα, οι μεταβλητές

h_1 , h_2 και h_3 θα πρέπει να αντιστοιχούν στην αρχική κατάσταση των σωρών, δηλαδή στην απάντηση που αναζητούμε.

Στη συνέχεια, συμπληρώστε το πρόγραμμα με τις κατάλληλες εντολές έτσι ώστε να γίνεται και η επαλήθευση της απάντησης. Δηλαδή, με βάση το αρχικό πλήθος σπάρτων που υπολογίσατε για τους τρεις σωρούς, ακολουθήστε κανονικά τα βήματα όπως περιγράφονται, γράφοντας τις αντίστοιχες εντολές. Το πρόγραμμα θα πρέπει στο τέλος να ελέγχει αν οι μεταβλητές h_1 , h_2 και h_3 έχουν την ίδια τιμή, επαληθεύοντας τη λύση.

Δώστε ιδιαίτερη σημασία στη σειρά με την οποία πρέπει να εκτελεστούν οι εντολές που μεταβάλλουν τις μεταβλητές h_1 , h_2 και h_3 . Αν χρειάζεστε έναν “σκελετό” για τη λύση της άσκησης, ανατρέξτε στο answer/exercises/matches-incomplete.py

answer/exercises/matches.py

ΕΙΣΟΔΟΣ ΚΑΙ ΕΞΟΔΟΣ Η ανάγνωση των τιμών που πληκτρολογεί ο χρήστης και γενικότερα η ροή πληροφορίας προς ένα πρόγραμμα από το περιβάλλον του ονομάζεται είσοδος. Η εμφάνιση μηνυμάτων και γενικότερα η ροή πληροφορίας από ένα πρόγραμμα προς το περιβάλλον του ονομάζεται έξοδος.

ΜΕΤΑΒΛΗΤΕΣ Μεταβλητή είναι το όνομα που δίνει ο προγραμματιστής σε μια τιμή ή ένα αντικείμενο. Χρειαζόμαστε τις μεταβλητές για τον ίδιο λόγο που χρειαζόμαστε γενικά τα ονόματα: για να μπορούμε να αναφερθούμε σε αυτές τις τιμές, ακόμα και όταν δεν γνωρίζουμε ή δεν έχει σημασία ποιες ακριβώς είναι αυτές. Η τιμή στην οποία αναφέρεται ένα όνομα, η τιμή της μεταβλητής, μπορεί να αλλάξει καθώς εκτελείται ένα πρόγραμμα. Είναι καλή πρακτική να επιλέγουμε περιγραφικά ονόματα για τις μεταβλητές μας, ονόματα που υποδηλώνουν το είδος των τιμών στις οποίες αντιστοιχούν.

ΔΟΜΗ ΕΠΙΛΟΓΗΣ Η δομή επιλογής δίνει τη δυνατότητα στα προγράμματά μας να ελέγχουν συνθήκες και να διαφοροποιούν την συμπεριφορά τους ανάλογα με αυτές. Χωρίς αυτή τη δυνατότητα, τα προγράμματά μας εκτελούν πάντα τις ίδιες εντολές, με την ίδια σειρά και αυτό τους στερεί την προσαρμοστικότητα ή την “εξυπνάδα” που είναι απαραίτητη ακόμα και σε πολύ απλά προβλήματα.

ΣΚΟΥΠΙΔΙΑ ΜΠΑΙΝΟΥΝ, ΣΚΟΥΠΙΔΙΑ ΒΓΑΙΝΟΥΝ Στο βιβλίο, ο *Deep Thought* προσφέρει μια εξήγηση για το παράδοξο της Απάντησης: “Για να είμαι ειλικρινής, πιστεύω ότι το πρόβλημα είναι ότι ποτέ δεν γνωρίζατε πραγματικά ποια είναι η ερώτηση. Αν μάθετε την ερώτηση, τότε θα καταλάβετε τι σημαίνει και η απάντηση.” Τα προγράμματα τα γράφουν προγραμματιστές. Άνθρωποι. Οι υπολογιστές απλά τα εκτελούν κι ελπίζουμε ότι η απάντηση που μας δίνουν αποτελεί τη λύση στο πρόβλημά μας. Για να γίνει όμως αυτό, θα πρέπει οι προγραμματιστές να έχουν κατανοήσει σωστά το πρόβλημα, οι οδηγίες που έχουν καθορίσει να είναι ορθές και τα δεδομένα που παρέχουν οι χρήστες να έχουν νόημα. Σε διαφορετική περίπτωση, η απάντηση θα μας είναι άχρηστη. Ο *Charles Babbage*, ο άνθρωπος που σχεδίασε έναν μηχανικό προγραμματιζόμενο υπολογιστή εκατό χρόνια πριν κατασκευαστούν οι πρώτοι ηλεκτρονικοί υπολογιστές, είχε γράψει: “Σε δύο περιπτώσεις έχω ερωτηθεί: Πείτε μας κύριε *Babbage*, αν εισάγετε λάθος νούμερα στη μηχανή, θα βγουν οι σωστές απαντήσεις; ... Δεν μπορώ να συλλάβω τι είδους σύγχυση ιδεών θα προκαλούσε μια τέτοια ερώτηση.”

ΤΑ ΤΕΣΣΕΡΑ ΚΑΠΕΛΑ Συνήθως όλα ξεκινούν από ένα πρόβλημα. Κάποιος σκέφτεται “δεν θα ήταν ωραία αν είχαμε ένα πρόγραμμα που να μας λύνει αυτό το πρόβλημα;” Μέσα σ’ αυτό το βιβλίο υπάρχουν αρκετά έτοιμα προβλήματα, αλλά σύντομα θα θελήσετε κι εσείς να φορέσετε το καπέλο του Προβληματοθέτη, γιατί είναι αλήθεια ότι τα πιο ενδιαφέροντα προβλήματα είναι αυτά που έχουν σημασία για εσάς. Μετά, υπάρχει κάποιος

που φοράει το καπέλο του Προγραμματιστή. Διατυπώνει, σε μια γλώσσα προγραμματισμού, τις εντολές που πρέπει να εκτελεστούν για να λυθεί το πρόβλημα. Εσείς πιθανότατα διαβάζετε αυτό το βιβλίο ακριβώς για να μάθετε πως μπορείτε να παίξετε αυτό το ρόλο. Τα προγράμματα τα εκτελούν οι υπολογιστές, οι οποίοι είναι φτιαγμένοι ειδικά γι' αυτόν το σκοπό, όμως πολύ συχνά θα φορέσετε κι εσείς το καπέλο του Εκτελεστή και θα εκτελέσετε εσείς οι ίδιοι τα προγράμματά σας, για να καταλάβετε πως λειτουργούν οι εντολές που γράψατε και, πιθανώς, γιατί δεν λειτουργούν όπως θα θέλατε. Τέλος, υπάρχουν εκείνοι που χρησιμοποιούν το πρόγραμμα, αλληλεπιδρούν μαζί του και αυτό λύνει για λογαριασμό τους το πρόβλημα από το οποίο ξεκίνησαν όλα. Θα φοράτε κι εσείς το καπέλο του Χρήστη όταν ελέγχετε τα προγράμματά σας ή απλά διασκεδάζετε με αυτά.
