8.3 python2 η Στοίβα

Παράδειγμα

Φτιάξτε πρόγραμμα που:

- Να θυμάται ιστοσελίδες (URL),
- με την σειρά που τις επισκεπτόμαστε
- Όταν πατάμε back, να μας φέρνει την προηγούμενη

το ιστορικό

http://www.iep.edu.gr
http://www.pythontutor.com
http://www.python.org

... αν πατήσω back ...

πάτησα back

http://www.pythontutor.com
http://www.python.org

... αν πάω στο minedu.gov.gr ...

πήγα στο minedu.gov.gr

http://www.minedu.gov.gr/
http://www.pythontutor.com
http://www.python.org

υλοποίηση με λίστες!

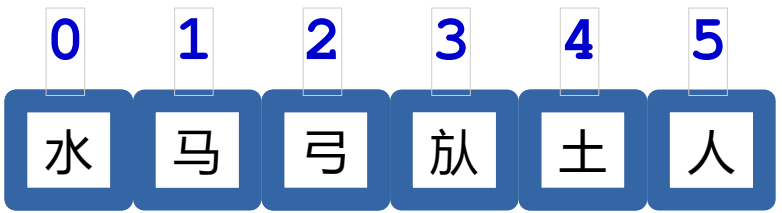
http://www.minedu.gov.gr/
http://www.pythontutor.com
http://www.python.org

άλλωστε με τις λίστες έχω τόσες πολλές **δυνατότητες**!

λίστα: αύξηση με string

ChineseLetters += “水马弓仈土人”

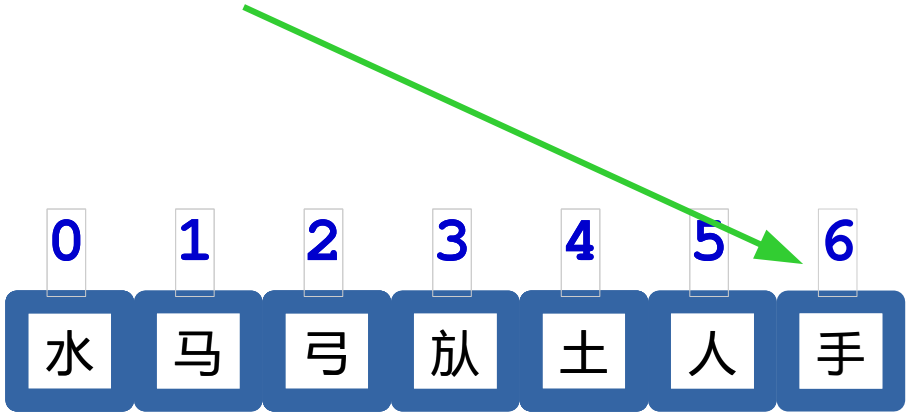
ChineseLetters :



https://en.wikipedia.org/wiki/Chinese_characters

λίστα: προσθήκη στο τέλος

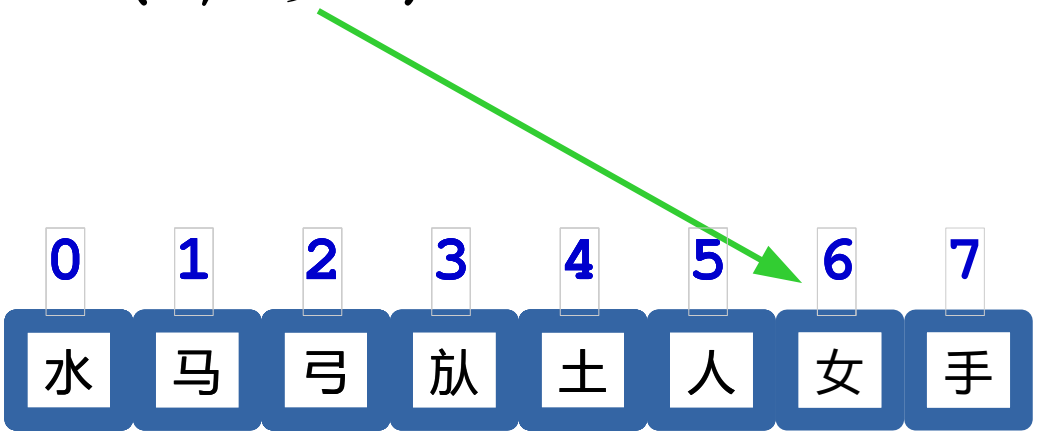
```
ChineseLetters.append(“手”)
```



https://en.wikipedia.org/wiki/Chinese_characters

λίστα: προσθήκη ό.θ.

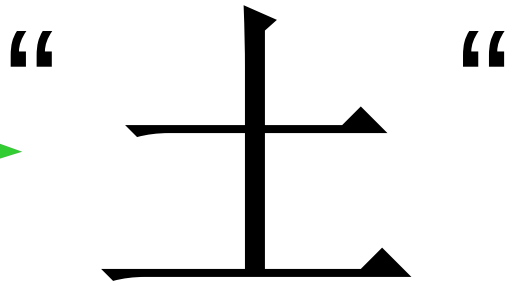
```
ChineseLetters.insert(6,“女”)
```



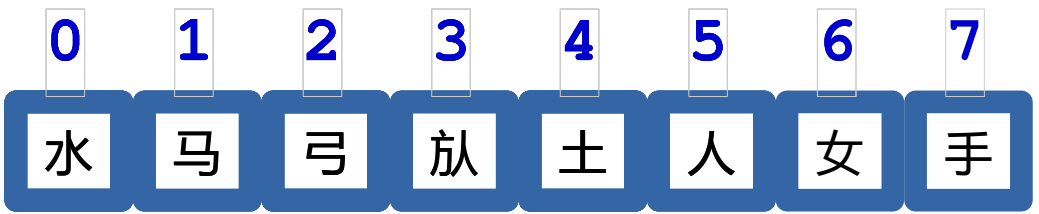
https://en.wikipedia.org/wiki/Chinese_characters

λίστα: παίρνω όποιο θέλω

ChineseLetters[4]



ChineseLetters :

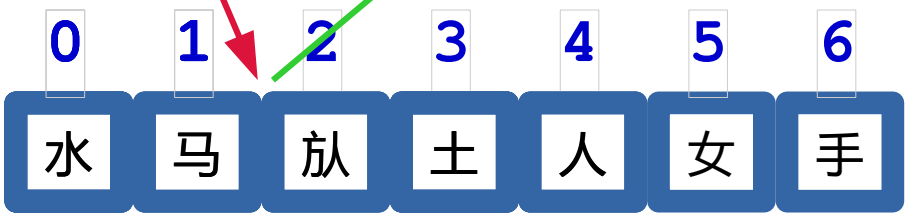


https://en.wikipedia.org/wiki/Chinese_characters

λίστα: βγάζω όποιο θέλω

```
ChineseLetters.pop(2)
```

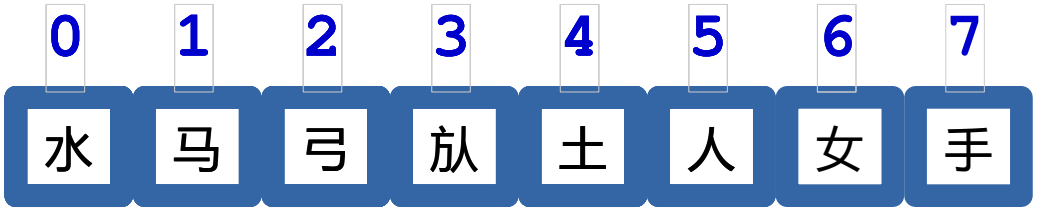
“ 弓 “



https://en.wikipedia.org/wiki/Chinese_characters

λίστα

ChineseLetters :



η λίστα είναι μία δομή δεδομένων **πολύ ευέλικτη**

Πλήρη πρόσβαση: διαγραφή, ανάγνωση, εισαγωγή

άμεσα : σε όποιο στοιχείο θέλουμε

https://en.wikipedia.org/wiki/Chinese_characters

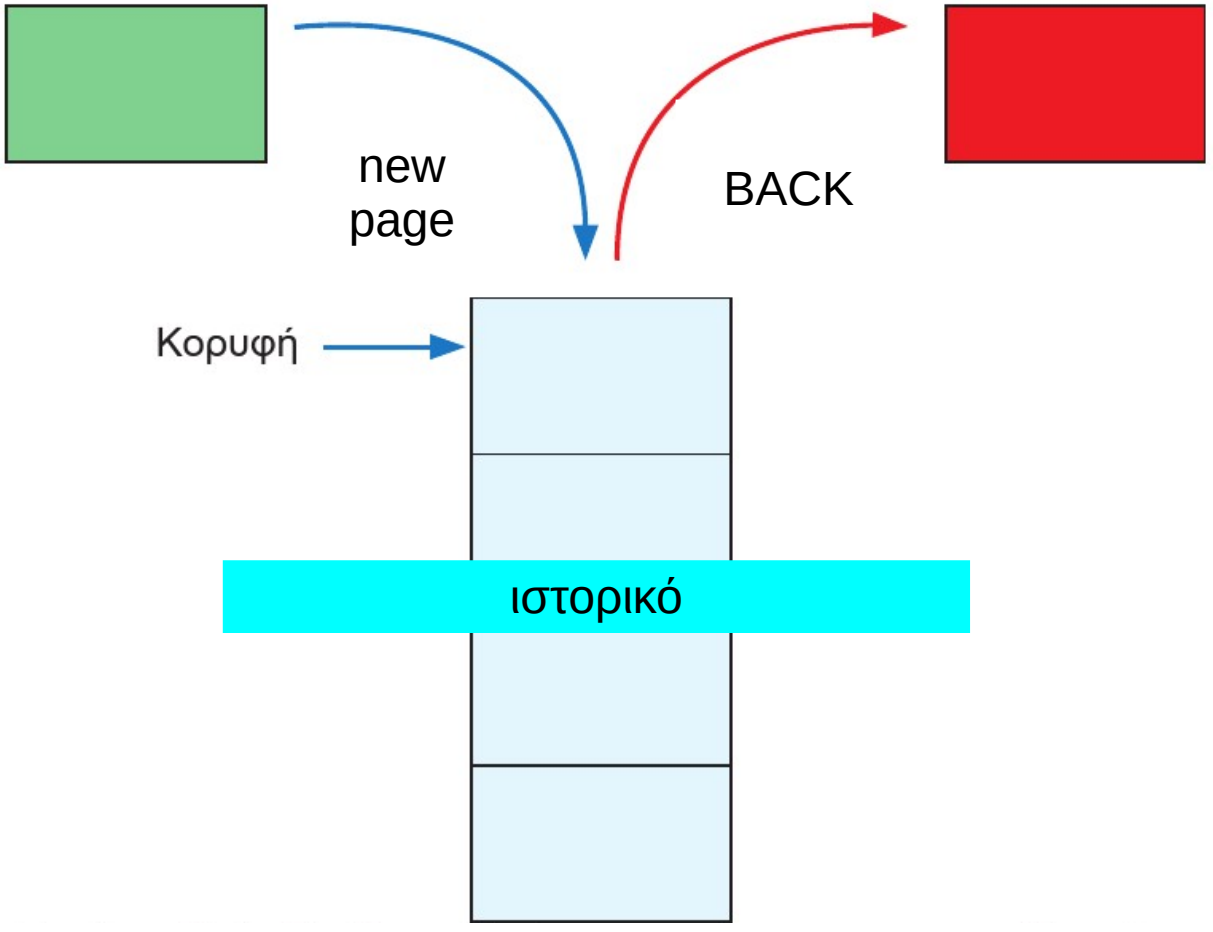
Όμως...

η ευελιξία ...

....μπορεί να φέρει **λάθη**

δεν χρειαζόμαστε πάντα όλες τις λειτουργίες

Τι θέλουμε;



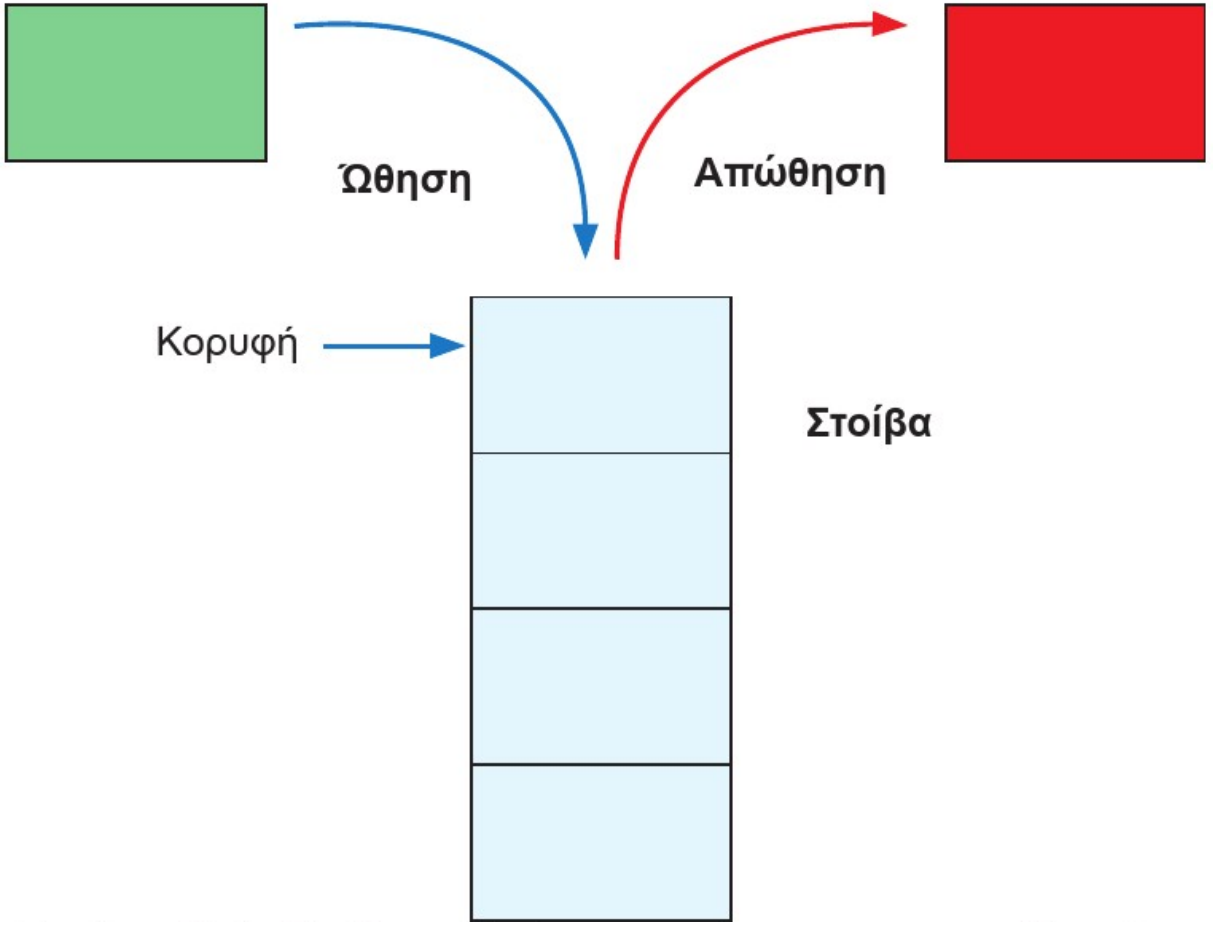
η στοίβα

stack

η στοίβα: μόνο δύο λειτουργίες

- ώθησε (push)
  - `στοίβα.push(“http://minedu.gov.gr”)`
- απώθησε (pop)
  - `print στοίβα.pop()`

Στοίβα: Last In First Out (LIFO)



η στοίβα: μόνο δύο λειτουργίες

- Απλούστερη δομή,
- Ευκολότερη χρήση,
- Λιγότερα λάθη

Παράδειγμα 1ο

Φτιάξτε πρόγραμμα που:

- Να θυμάται ιστοσελίδες (URL),
- με την σειρά που τις επισκεπτόμαστε
- Όταν πατάμε back, να μας φέρνει την προηγούμενη

text browser

```
from stack import *

print "Type . to end"
print "Type BACK to go back"
print "Type a URL to go there"

twra = "home page"
epomeno = raw_input("URL:")

#η συνέχεια στην επόμενη διαφάνεια
```

(Ιδανικό) Παράδειγμα χρήσης

```
ηστοίβαμου = MyStack()

while epomeno!=".":
    if epomeno == "BACK" and ηστοίβαμου.notEmpty():
        print "Going to "
        twra = ηστοίβαμου.pop()
        print twra
    elif epomeno == ".":
        print "this is the end"
    else:
        print "Going to ", epomeno
        ηστοίβαμου.push(twra)
        twra = epomeno
        ηστοίβαμου.emfanise()
    epomeno = raw_input("URL:")
```

επειδή δεν ξέρουμε απο κλάσεις

θα υλοποιήσουμε την στοίβα μόνο με **συναρτήσεις**

υλοποιήση με συναρτήσεις

θα χρειαστούμε

- createStack(): δημιουργεί μία στοίβα
  - Στην πραγματικότητα δημιουργεί μία λίστα
- push(s, u) : **ωθεί** στην στοίβα s, το URL u
  - Στην πραγματικότητα το βάζει σε μία λίστα
- pop(s) : **απωθεί** το τελευταίο στοιχείο στοίβας s
  - Στην πραγματικότητα βγάζει το κατάλληλο στοιχείο από μία λίστα
- isEmpty(s): επιστρέφει True αν η στοίβα είναι άδεια
  - Στην πραγματικότητα αν η λίστα είναι άδεια...

Παράδειγμα χρήσης

```
stack = createStack()

while epomeno!=".":
    if epomeno == "BACK" and not isEmpty(stack):
        print "Going to "
        twra = pop( stack )
        print twra
    elif epomeno == ".":
        print "this is the end"
    else:
        print "Going to ", epomeno
        push(stack, twra)
        twra = epomeno
        emfanise(stack)
    epomeno = raw_input("URL:")
```

Τι κερδίζω;

- Απλούστερος χειρισμός
  - Μονο δύο πράγματα μπορώ να κάνω:
    - ώθησε push
    - απώθησε pop
  - Δεν μπλέκω καθόλου με δείκτες...

- Μικρότερη πιθανότητα να κάνω κάποιος λάθος

Παράδειγμα 2ο

- Ο χρήστης δίνει αριθμούς
  - μέχρι να δώσει το 0
- Το πρόγραμμα εμφανίζει τους αριθμούς
  - με την ανάποδη σειρά

Παράδειγμα 2ο

- Ο χρήστης δίνει :
  - 1
  - 2
  - 3
  - 0
- Το πρόγραμμα εμφανίζει :
  - 3
  - 2
  - 1

Λύση 1/3

```
stack = createStack()
```

Λύση 2/3

```
number = int( raw_input( ) )
while number != 0 :
    push(stack, number)
    number = int( raw_input( ) )
```

Λύση 3/3

```
while not isEmpty( stack ) :
    number = pop(stack)
    print number
```

Πως υλοποιούνται οι συναρτήσεις

- createStack()
- push(stack, item)
- pop(stack)
- isEmpty(stack)

1η υλοποίηση

```
def push(stack, item) :
    stack.append( item )
def pop(stack) :
    return stack.pop( )
def isEmpty(stack) :
    return len(stack) == 0
def createStack( ) :
    return [ ]
```

2η υλοποίηση

```
def push(stack, item) :
    stack.insert(0, item)
def pop(stack) :
    return stack.pop( 0 )
def isEmpty(stack) :
    return len(stack) == 0
def createStack( ) :
    return [ ]
```

Ποιά υλοποίηση κάναμε εδώ;

```
stack = createStack()

while epomeno!=".":
    if epomeno == "BACK" and not isEmpty(stack):
        print "Going to "
        twra = pop( stack )
        print twra
    elif epomeno == ".":
        print "this is the end"
    else:
        print "Going to ", epomeno
        push(stack, twra)
        twra = epomeno
        emfanise(stack)
    epomeno = raw_input("URL:")
```

Ποιά υλοποίηση κάναμε εδώ;

```
stack = createStack()
number = int( raw_input( ) )
while number != 0 :
    push(stack, number)
    number = int( raw_input( ) )
while not isEmpty( stack ) :
    number = pop(stack)
    print number
```

Δεν ξέρουμε!

- Απόκρυψη υλοποίησης
- Χρησιμοποιούμε μόνο συγκεκριμένες συναρτήσεις
  - Διεπαφή της δομής (interface)

Διαχωρισμός διεπαφής - υλοποίησης

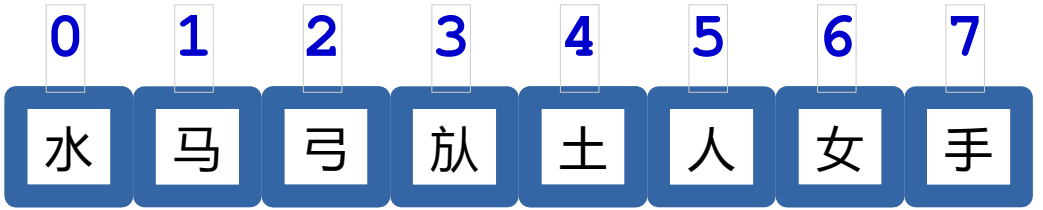
Διεπαφή της δομής δεδομένων Στοίβα

```
def push(stack, item)
def pop(stack)
def isEmpty(stack)
def createStack( )
```

<u>Διεπαφή</u>
Ποιές λειτουργίες έχει η δομή μου

κινέζικα

```
ChineseLetters =[
```




```
]
```

水（氵）water
马 / 馬 horse
弓 bow (bend)
旅 flag
土 earth
人（亻）person
女 female
手（扌）hand

https://en.wikipedia.org/wiki/Chinese_characters