ΚΡΥΠΤΟΓΡΑΦΙΑ | ΕΛΕΓΧΟΣ ΡΟΗΣ

ΕΛΕΓΧΟΣ ΡΟΗΣ

Πρακτικά Ζητήματα

Εισαγωγή: Στα προγράμματα που είδαμε μέχρι τώρα, υπήρξε πάντα μια σειρά εντολών, τις οποίες εκτελούσε πιστά η Python με την ίδια σειρά. Αν θέλουμε όμως να αλλάξουμε την ροή εκτέλεσης; Αν, για παράδειγμα, θέλουμε το πρόγραμμα να πάρει μερικές αποφάσεις και να κάνει διαφορετικά πράγματα υπό διαφορετικές προϋποθέσεις, όπως π.χ. να εκτυπώσει "καλημέρα" ή "καλησπέρα", ανάλογα με την ώρα;

Εντολή if: Οι όροι που χρησιμοποιούνται συχνά στην κρυπτογραφία εξηγούνται στη συνέχεια. Η εντολή if χρησιμοποιείται για να ελεγχθεί μια συνθήκη και εάν (if) η συνθήκη αυτή είναι αληθής, τότε εκτελείται ένα σύνολο ή πλοκάδα εντολών (που ονομάζεται if-block), διαφορετικά (else) γίνεται επεξεργασία ενός άλλου συνόλου εντολών (που ονομάζεται else-block). Η χρήση του όρου else είναι προαιρετική.

```
# Filename: if.py

number = 23
guess = int(input('Εισάγετε έναν ακέραιο
αριθμό: '))

if guess == number:
   print('Συγχαρητήρια, τον μαντέψατε.')
   print('(Αλλά δεν κερδίζετε και κανένα
   βραβείο!)')

elif guess < number:
   print('Όχι, είναι λίγο μεγαλύτερος.')

else:
   print('Όχι, είναι λίγο μικρότερος.')

print('Τέλος')
```

Εντολή while: Η εντολή while σας επιτρέπει να εκτελείτε επανειλημμένα μια πλοκάδα εντολών, όσο μια προϋπόθεση παραμένει αληθές. Η εντολή while είναι ένα παράδειγμα αυτού που αποκαλείται εντολή βρόχου (looping statement). Μια εντολή while μπορεί να έχει έναν προαιρετικό όρο else.

```
# Filename: while.py
number = 23
running = True
while running:
   guess = int(input('Εισάγετε έναν ακέραιο
   αριθμό: '))

if guess == number:
   print('Συγχαρητήρια, τον μαντέψατε.')
   running = False # αυτό κάνει τον βρόχο while
   να σταματήσει εδώ

elif guess < number:
   print('Όχι, είναι λίγο μεγαλύτερος.')

else:
   print('Όχι, είναι λίγο μικρότερος.')

else:
   print('Ο βρόχος while τερματίστηκε.')

# Μπορείτε να προσθέσετε ότι άλλο θέλετε εδώ
print('Τέλος')
```

Ο βρόχος for: Η εντολή for..in είναι άλλη μία εντολή βρόχου, η οποία επαναλαμβάνεται σε μια ακολουθία αντικειμένων, δηλαδή εκτελείται σε κάθε αντικείμενο σε μια ακολουθία. Θα δούμε περισσότερες λεπτομέρειες σχετικά με τις ακολουθίες στα επόμενα κεφάλαια. Αυτό που πρέπει να γνωρίζετε προς το παρόν είναι ότι μια ακολουθία είναι απλά μια ταξινομημένη συλλογή αντικειμένων.

```
# Filename: for.py

for i in range(1, 5):
   print(i)

else:
   print('Ο βρόχος loop τερματίστηκε')
```

ΠΑΡΑΔΟΤΕΟ 02 (if, while, for)

Εκτελέστε τις εντολές δημιουργώντας ένα αρχείο .py και **σχολιάστε σε κάθε γραμμή τι συντελείται**.

Υποχρεωτικά! Προσθέστε **δικές σας τιμές** σε κάθε περίπτωση. Ανεβάστε το αρχείο py.

ΚΡΥΠΤΟΓΡΑΦΙΑ | ΣΥΝΑΡΤΗΣΕΙΣ

Εισαγωγή: Οι συναρτήσεις είναι επαναχρησιμοποιήσιμα μέρη προγραμμάτων. Σας επιτρέπουν να δίνετε ένα όνομα σε ένα σύνολο εντολών και να τρέχετε εκείνο το σύνολο εντολών χρησιμοποιώντας το όνομά τους, οπουδήποτε στο πρόγραμμά σας και όσες φορές θέλετε. Αυτό είναι γνωστό σαν κλήση (calling) της συνάρτησης. Έχουμε ήδη χρησιμοποιήσει πολλές ενσωματωμένες συναρτήσεις όπως τη len και τη range.

Η έννοια των συναρτήσεων είναι πιθανόν το πιο σπουδαίο δομικό στοιχείο οποιουδήποτε μη στοιχειώδους προγράμματος (σε όλες τις γλώσσες προγραμματισμού), γι' αυτό θα διερευνήσουμε διάφορες πτυχές των συναρτήσεων σε αυτό το κεφάλαιο. Οι συναρτήσεις ορίζονται χρησιμοποιώντας τη λέξη κλειδί def, μετά την οποία ακολουθεί ένα όνομα που ταυτοποιεί την εκάστοτε συνάρτηση και κατόπιν ακολουθεί ένα ζευγάρι παρενθέσεων που μπορούν να περικλείουν μερικά ονόματα μεταβλητών, και η γραμμή τελειώνει με διπλή τελεία (:).

```
# Filename: function1.py

def sayHello():
    print('Hello World!') # σύνολο εντολών που
    ανήκουν στη συνάρτηση

# Τέλος της συνάρτησης
sayHello() # κλήση της συνάρτησης
sayHello() # κλήση της συνάρτησης ξανά
```

Παράμετροι Συναρτήσεων: Μια συνάρτηση μπορεί να δεχθεί παραμέτρους, οι οποίες είναι τιμές που δίνετε στη συνάρτηση, έτσι ώστε αυτή να μπορεί να κάνει κάτι αξιοποιώντας αυτές τις τιμές. Αυτές οι παράμετροι μοιάζουν με τις μεταβλητές, διαφέροντας ως προς το ότι οι τιμές αυτών των μεταβλητών ορίζονται όταν καλούμε τη συνάρτηση και τους έχουν ήδη εκχωρηθεί τιμές όταν τρέχει η συνάρτηση. Οι παράμετροι καθορίζονται μέσα στο ζευγάρι των παρενθέσεων στον ορισμό της συνάρτησης και διαχωρίζονται με κόμμα. Όταν καλούμε τη συνάρτηση δίνουμε και τις τιμές με τον ίδιο τρόπο. Σημείωση για την ορολογία που χρησιμοποιείται: οι ονομασίες που δίνετε στον ορισμό της συνάρτησης ονομάζονται παράμετροι ενώ οι τιμές που δίνετε όταν καλείτε τη συνάρτηση ονομάζονται ορίσματα.

```
# Filename: func_param.py
def printMax(a, b):
    if a > b:
        print(a, 'είναι το μέγιστο')
        elif a == b:
            print(a, 'είναι ίσο με το', b)
        else:
            print(b, 'είναι το μέγιστο')

printMax(3, 4)
y = 7

printMax(x, y)
```

ΠΑΡΑΔΟΤΕΟ 03 (function1, param, sys, mymodule, demo)

Εκτελέστε τις εντολές δημιουργώντας ένα αρχείο .py και **σχολιάστε σε κάθε γραμμή τι συντελείται**.

Υποχρεωτικά! Προσθέστε **δικές σας τιμές** σε κάθε περίπτωση. Ανεβάστε το αρχείο py.

Αρθρώματα: Ένα άρθρωμα μπορεί να εισαχθεί από ένα άλλο πρόγραμμα για να κάνετε χρήση της λειτουργικότητάς του. Έτσι μπορείτε να χρησιμοποιήσετε επίσης την πρότυπη βιβλιοθήκη της Python. Αρχικά θα δείτε πως να χρησιμοποιείτε τα αρθρώματα της πρότυπης βιβλιοθήκης.

```
# Filename: using_sys.py

import sys
print('The command line arguments are:')
for i in sys.argv:
    print(i)
print('\n\nThe PYTHONPATH is', sys.path, '\n')
```

Custom Αρθρώματα (libraries): Η δημιουργία των δικών σας αρθρωμάτων είναι εύκολη, το έχετε κάνει ήδη, κι αυτό διότι κάθε πρόγραμμα στην Python είναι επίσης κι ένα άρθρωμα. Για το μόνο που πρέπει να είστε σίγουροι είναι να έχει μια επέκταση .py.

```
# Filename: mymodule.py
def sayhi():
    print('Hi, this is mymodule speaking.')

__version__ = '0.1'
# End of mymodule.py
```

ΚΡΥΠΤΟΓΡΑΦΙΑ | MODULES/REVERSE

Θυμηθείτε ότι το άρθρωμα πρέπει να τοποθετείται στον ίδιο κατάλογο με το πρόγραμμα που το εισάγουμε, ή το άρθρωμα πρέπει να βρίσκεται σε έναν από τους καταλόγους που είναι στη λίστα του sys.path.

```
# Filename: mymodule_demo.py

import mymodule # Προαιρετικά import *
mymodule.sayhi()
print ('Version', mymodule.__version__)
```

Λίστες: Μια λίστα είναι μια δομή δεδομένων που συγκρατεί μια διατεταγμένη συλλογή στοιχείων, δηλαδή μπορείτε να αποθηκεύσετε μια ακολουθία (sequence) αντικειμένων στη λίστα. Η λίστα των στοιχείων πρέπει να κλείνεται σε αγκύλες (δηλαδή [και]) έτσι ώστε να καταλαβαίνει η Python ότι καθορίζετε μια λίστα.

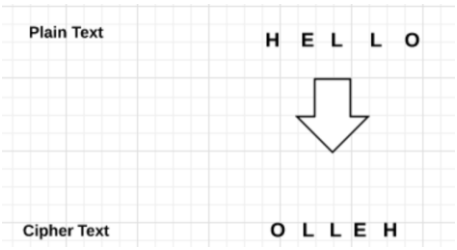
```
# Filename: using_list.py
# Αυτή είναι η λίστα αγορών μου

shoplist = ['μήλο', 'μάνγκο', 'καρότο',
'μπανάνα']
print('Πρέπει ν\' αγοράσω', len(shoplist),
'πράγματα.')
print('Τα πράγματα αυτά είναι:', end=' ')
for item in shoplist:
   print(item, end=' ')

print('\nΠρέπει επίσης ν\' αγοράσω ρύζι.')
shoplist.append('ρύζι')
print('Η λίστα αγορών μου τώρα είναι:',
shoplist)
print('Θα ταξινομήσω τη λίστα μου τώρα')
shoplist.sort()
print('Η ταξινομημένη λίστα μου είναι',
shoplist)
print('Το πρώτο πράγμα που θ\' αγοράσω είναι',
shoplist[0])
olditem = shoplist[0]
del shoplist[0]
print('Αγόρασα το', olditem)
print('Η λίστα αγορών μου τώρα είναι',
shoplist)
```

Αφού έχετε δημιουργήσει μια λίστα μια φορά μπορείτε να προσθέσετε, να μετακινήσετε ή να ψάξετε για στοιχεία σ' αυτή τη λίστα. Από τη στιγμή που μπορούμε να προσθέσουμε και να μετακινήσουμε στοιχεία, λέμε ότι η λίστα είναι ένας μεταβλητός τύπος δεδομένων (mutable data type), δηλαδή αυτός ο τύπος μπορεί να αλλαχθεί.

Αντίστροφη Κρυπτογράφηση: Ο αλγόριθμος της αντίστροφης κρυπτογράφησης έχει τα ακόλουθα χαρακτηριστικά.

i. Το Reverse Cipher χρησιμοποιεί ένα μοτίβο αντιστροφής της συμβολοσειράς απλού κειμένου για μετατροπή στο κρυπτογραφημένο κείμενο.
ii. Η διαδικασία κρυπτογράφησης και αποκρυπτογράφησης είναι η ίδια.
iii. Για να αποκρυπτογραφήσει το κρυπτογραφημένο κείμενο, ο χρήστης χρειάζεται απλώς να αντιστρέψει το κρυπτογραφημένο κείμενο για να το λάβει απλό κείμενο.

Το κύριο μειονέκτημα του αντίστροφου κρυπτογράφησης είναι ότι είναι πολύ αδύναμος. Το αποτέλεσμα είναι να μπορεί εύκολα να σπάσει το κρυπτογραφημένο κείμενο για να λάβετε το αρχικό μήνυμα. Ως εκ τούτου, η αντίστροφη κρυπτογράφηση δεν θεωρείται ως καλή επιλογή για τη διατήρηση ασφαλούς καναλιού επικοινωνίας.



Σχήμα 1. Αντιστροφή

```
# Filename: reverse.py
message = 'This is program to explain reverse
cipher.'
translated = '' #cipher text is stored in this
variable
i = len(message) - 1

while i >= 0:
   translated = translated + message[i]
   i = i - 1

print("The cipher text is : ", translated)
```

ΠΑΡΑΔΟΤΕΟ 04 (using_list, reverse)

Εκτελέστε τις εντολές δημιουργώντας ένα αρχείο .py και **σχολιάστε σε κάθε γραμμή τι συντελείται**.

Υποχρεωτικά! Προσθέστε **δικές σας τιμές** σε κάθε περίπτωση. Ανεβάστε το αρχείο py.