Τμήμα Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών

Μάθημα: **Τεχνικές Προγραμματισμού Υπολογιστών. (Εργαστηριακό μάθημα)**
Καθηγητής : Πεφάνης Ευάγγελος

1) Εργαστηριακές σημειώσεις στην γλώσσα προγραμματισμού Python.

Μεταβλητές

Τύποι μεταβλητών

1. `int` (ακέραιος)
2. `float` (κινητής υποδιαστολής)
3. `bool` (λογική μεταβλητή: παίρνει τις τιμές `True` ή `False`)
4. `NoneType` (παίρνει την τιμή `None`, δηλώνει απουσία τιμής)

Παράδειγμα Θα δώσουμε τιμές σε τρεις μεταβλητές (κινητής υποδιαστολής) με ονόματα: `pi, a, area`.

```
pi = 3.14
a = 10.0
area = pi*a**2
```

Βρήκαμε την επιφάνεια δίσκου με ακτίνα ίση με 10. Χρειάστηκε να υψώσουμε στο τετράγωνο (`a**2`) και να κάνουμε πολλαπλασιασμό (με το σύμβολο `*`). Τελικά εκχωρήσαμε το αποτέλεσμα του π * a^2 στη νέα μεταβλητή `area`.

Παράδειγμα Δίνουμε τιμές σε μεταβλητές:

```
>>> a = 3
>>> b = 3.14
>>> c = True
>>> d = None
```

Με την εντολή `type` μπορούμε να ελέγξουμε τον τύπο των μεταβλητών οι οποίες ήδη έχουν ορισθεί (έχουν τιμές).

```
>>> type(a)
<class 'int'>
>>> type(b)
<class 'float'>
>>> type(c)
<class 'bool'>
```

```
>>> type(d)
<class 'NoneType'>
```

Ονόματα μεταβλητών

- Μπορούν να περιέχουν γράμματα και αριθμούς (αλφαριθμητικούς χαρακτήρες),
- αλλά πρέπει να ξεκινούν με ένα γράμμα.
- Τα πεζά διακρίνονται από τα κεφαλαία.
- Μπορεί να περιέχεται το underscore (_).

Παραδείγματα

```
>>> natural_number = 3
>>> to_pi = 3.14
>>> greeting = 'hallo there!'
>>> protasi = True
>>> bathmos = None
```

Μη αποδεκτά ονόματα μεταβλητών:

- 1number - αρχίζει με αριθμό.
- number! - περιέχει το !
- class - είναι λέξη-κλειδί.

Λέξεις κλειδιά

Είναι δεσμευμένες λέξεις με ειδική σημασία. Για να τις δούμε όλες:

```
>>> import keyword
>>> keyword.kwlist
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue',
'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if',
'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return',
'try', 'while', 'with', 'yield']
```

Εντολές

Έχουμε ήδη δει τις μερικές εντολές της python: `print, type, int, float, str`.

Παραδείγματα:

```
>>> print(a)
3
>>> print(c)
hallo there!
>>> print(d)
True
>>> print(a,b)
3 3.14
>>> type(c)
```

```
<class 'str'>
>>> int(b)
3
>>> float(a)
3.0
>>> int("19")
19
>>> str(3.14159)
'3.14159'
```

'Αλλα παραδείγματα:

```
>>> print(a,b,c)
3 3.14 hallo there!
>>> print(c,'Nick')
hallo there! Nick
```

Πακέτα

Έχουμε δει επίσης την εντολή `import` η οποία μας επιτρέπει πρόσβαση σε προκαθορισμένες μεταβλητές, συναρτήσεις κλπ. Το πακέτο `math` περιέχει χρήσιμες μαθηματικές σταθερές και μαθηματικές συναρτήσεις.

```
>>> import math
>>> print(math.pi)
3.141592653589793
>>> math.sqrt(2.0)
1.4142135623730951
>>> math.sqrt(2)
1.4142135623730951
>>> math.exp(1)
2.718281828459045
```

Σφάλματα

Ας δούμε τα εξής

```
>>> print c
SyntaxError: Missing parentheses in call to 'print'
>>> print(c,Nick)
Traceback (most recent call last):
  File "", line 1, in
    print(c,Nick)
NameError: name 'Nick' is not defined
```

Η πρώτη εντολή έχει **συντακτικό** λάθος (syntax error). Η δεύτερη εντολή έχει ένα λάθος το οποίο η Python ονομάζει NameError.

Ας δούμε επίσης

```
>>> print(3,2)
3 2
```

Τυπώσαμε δύο αριθμούς. Εάν όμως ο σκοπός μας ήταν να τυπώσουμε τον αριθμό 3.2 τότε έχουμε κάνει ένα **λάθος σημαντικής** (semantic error). Δηλαδή, η εντολή δίνει μεν ένα αποτέλεσμα, αλλά δεν είναι αυτό που θέλαμε.

Strings (συμβολοσειρές)

Μπορούμε να ορίσουμε ακολουθίες χαρακτήρων, τα όρια των οποίων καθορίζονται από απλές (') ή διπλές (") αποστρόφους.

Παραδείγματα. Ορίζουμε μεταβλητές οι οποίες περιέχουν συμβολοσειρές. Επίσης, μπορούμε μα εφαρμόσουμε σε αυτές τους τελεστές πρόσθεσης (+) και πολλπλασιασμού (*).

```
>>> a = 'Nick'
>>> b = 'Papadakis'
>>> onoma = a + b
>>> print(onoma)
NickPapadakis
>>> onoma = a + " " + b
>>> print(onoma)
Nick Papadakis
>>> len(onoma)
14
>>> 2*a
'NickNick'

>>> year_in = 2016
>>> print(onoma,"came to Crete in",year_in)
Nick Papadakis came to Crete in 2016
```

Κάθε χαρακτήρας σε ένα string έχει αριθμό θέσης (δείκτη, index). Η αρίθμηση ξεκινάει από το 0 και ο τελευταίος χαρακτήρας ενός string με μήκος n έχει δείκτη n-1.

```
>>> len(onoma)
14
>>> onoma[0]
'N'
>>> onoma[13]
's'
>>> onoma[1]
'i'
```

Slicing: επιλέγουμε ένα τμήμα της συμβολοσειράς. Ο συμβολισμός onoma[start:end] αναφέρεται σε ένα τμήμα του αρχικού string, ειδικότερα στους χαρακτήρες του string onoma που βρίσκονται στις θέσεις start, start+1,..., end-1.

```
>>> onoma[0:4]
'Nick'
>>> onoma[1:4]
'ick'
```

Η θέση κάθε χαρακτήρα θέσης μπορεί επίσης να καθορισθεί μετρώντας από το τέλος της συμβολοσειράς (string). Αν το μήκος του string είναι `n` τότε η αρίθμηση ξεκινάει από το -n. Το συνολικό string μπορεί να αναφερθεί ως s[-n:].

```
>>> s = 'Nick'
>>> s[-4:-1]
'Nic'
>>> s[-4:]
'Nick'
```

Μπορούμε να γράψουμε [start:end:step] για να αναφερθούμε στις θέσεις μεταξύ start και end-1 με βήμα step.

```
>>> s1 = 'Take'
>>> s2 = 'Make'
>>> s3 = 'Fake'
>>> s = s1 + s2 + s3
>>> s
'TakeMakeFake'
>>> len(s)
12
>>> s[0:12:4]
'TMF'
```

Είσοδος δεδομένων.

Η είσοδος δεδομένων (από το πληκτρολόγιο) γίνεται με τη συνάρτηση `input`. Το (προαιρετικό) όρισμα της συνάρτησης τυπώνεται στην οθόνη κατά την εκτέλεση του προγράμματος ως μια προτροπή (prompt) στο χρήστη να εισαγάγει δεδομένα. Οτιδήποτε εισάγει ο χρήστης αποθηκεύεται ως ακολουθία χαρακτήρων και εκχωρείται στη μεταβλητή αριστερά του ίσον (=).

```
>>> name = input('What is your name? ')
What is your name? Barack Obama
>>> print('Hello,', name)
Hello, Barack Obama
>>> age = input('How old are you? ')
How old are you? 42
>>> print(age)
42
>>> type(age)
<class 'str'>
```

Παρατηρήστε ότι ο τύπος της μεταβλητής *name* αλλά και της *age* είναι `str`, δηλαδή ακολουθία χαρακτήρων (και όχι `int` που ίσως να περίμενε κάποιος). Μπορούμε όμως να μετατρέψουμε τη μεταβλητή *age* σε ακέραιο γράφοντας `int(age)` και με αυτό τον τρόπο αυτή μπορεί να χρησιμοποιηθεί σε αριθμητικές εκφράσεις:

```
>>> age = input('How old are you? ')
How old are you? 42
>>> my_age = 25
>>> print('You are', int(age)-my_age, 'years older than I am!')
You are 17 years older than I am!
```

Μπορούμε επίσης να περάσουμε το αποτέλεσμα της `input` απ' ευθείας σε μία άλλη συνάρτηση, π.χ., στην `float`.

```
>>> price = float(input('How much for the apples? '))
How much for the apples? 1.79
>>> type(price)
<class 'float'>
>>> kilos = 4.5
>>> cost = price * kilos
```

Εντολές ελέγχου

Λογικές εκφράσεις και μεταβλητές

Παραδείγματα

```
>>> x = 5     # give a value to the variable x
>>> y = 7     # give a value to the variable y
>>> x == y    # check if x is equal to y
False
>>> x < y     # check if x is smaller than y, etc
True
>>> x <= y
True
>>> x > y
False
>>> x >= y
False
>>> x != y    # check if x is not equal to y
True
```

Λογικές μεταβλητές

```
>>> cond = (x == y)
>>> cond
False
>>> type(cond)
>class 'bool'>
```

Παράδειγμα. Είναι το φετινό έτος δίσεκτο;

```
>>> year = 2016
>>> isleap = year%4 == 0
>>> print(isleap)
```

Σημείωση: η παραπάνω συνθήκη χρειάζεται βελτίωση (θα το δούμε αργότερα).

Εκτέλεση υπό συνθήκη (εντολή if)

Παράδειγμα. `if`

```
>>> x = 5
>>> if x>0:
        print("x is positive")
        print("x =",x)
```

Παρατήρηση: δείτε τη σημασία της εσοχής στην python.

Παράδειγμα. `if - else`

```
>>> x = 5
>>> if x%2==0:
        print(x,"is even")
else:
        print(x,"is odd")
```

Παράδειγμα. `if - elif - else`

```
>>> x = 5
>>> y = 7
>>> if x>y:
        print(x,"is greater than",y)
elif(x<y):
        print(x,"is less than",y)
else:
        print(x,"and",y,"are equal")
```

Παράδειγμα. Είναι το φετινό έτος δίσεκτο;

```
>>> year = 2016
>>> isleap = year%4 == 0
>>> if isleap:
       print("Year",year,"is leap year")
else:
       print("Year",year,"is not leap year")
```

Η συνθήκη που χρησιμοποιήσαμε στην παραπάνω άσκηση χρειάζεται βελτίωση:

```
>>> year = 2016
>>> isleap = (year%4 == 0) and (year%100 != 0)
>>> print(isleap)
```

και επιπλέον βελτίωση

```
>>> year = 2016
>>> isleap = (year%4 == 0) and (year%400 != 0) or (year%400 == 0)
>>> print(isleap)
```

Τελεστές για λογικές μεταβλητές `b, c` (κατά σειρά χαμηλότερης προτεραιότητας)

- `b or c`
- `b and c`
- `not b`

Φωλιασμένες εντολές if

Εντολές ελέγχου `if-elif-else` μπορούν να είναι μέρος εντολών οι οποίες εκτελούνται υπό μία συνθήκη.

Παράδειγμα. Οι παρακάτω εντολές επιλέγουν τον μέγιστο των *x, y, z* :

```
if x >= y:
    if x >= z:
        print('x is largest')
    else:
        print('z is largest')
else:
    if y >= z:
        print('y is largest')
    else:
        print('z is largest')
```

Μια κομψότερη, ίσως, λύση (η οποία μπορεί εύκολα να γενικευθεί σε μεγαλύτερο πλήθος αριθμών) είναι η ακόλουθη:

```
maxnum = x
if y > maxnum:
    maxnum = y
if z > maxnum:
    maxnum = z
print('largest number is', maxnum)
```

Αλγόριθμοι

Παράδειγμα: Εύρεση κυβικής ρίζας τέλειου κύβου. Η κυβική ρίζα κάποιων αριθμών είναι ακέραιος. Μπορούμε να βρούμε την κυβική ρίζα του $a=216$a=216 (και άλλων τελείων κύβων) χρησιμοποιώντας το παρακάτω πρόγραμμα. Θα πρέπει να το τρέξουμε πολλές φορές δοκιμάζοντας διαδοχικούς ακεραίους. Η μέθοδος ονομάζεται *εξαντλητική απαρίθμηση*.

```
# find the cubic root of a perfect cube

perfect_cube = 216   # this is a perfect cube

print("Try to guess the cubic root of",perfect_cube)
x = input("Give an integer: ")
x = int(x)

if x**3 == perfect_cube:
    print("Yes, indeed!", x,"is the cubic root of",perfect_cube)
elif x**3 < perfect_cube:
     print("You need a larger number")
elif x**3 > perfect_cube:
     print("You need a smaller number")
```

Ένας **αλγόριθμος** αποτελείται από βήματα υπολογισμών τα οποία εφαρμόζονται σε κάποια αρχικά δεδομένα και, αφού εξελιχθούν μέσω διαφόρων καταστάσεων, δίνουν κάποια τελικά δεδομένα (δηλαδή, ένα αποτέλεσμα).

Για να εφαρμόσουμε έναν αλγόριθμο χρειαζόμαστε μία **γλώσσα προγραμματισμού**. Μία γλώσσα προγραμματισμού δίνει εντολές στον υπολογιστή ώστε να πραγματοποιηθούν οι πράξεις, οι οποίες αποτελούν τα βήματα οποιουδήποτε αλγορίθμου.

Προγράμματα σε μία γλώσσα *υψηλού επιπέδου* (π.χ., python) πρέπει να υποστούν επεξεργασία ώστε να μετατραπούν σε προγράμματα γλώσσας χαμηλού επιπέδου (γλώσσα μηχανής) τα οποία μπορεί να τρέξει ο υπολογιστής. Η επεξεργασία γίνεται από διερμηνευτές (interpreters) και μεταγλωττιστές (compilers).