Python – Τα Βασικά

Ένας απλός κώδικας

```
x = 34 - 23                    # A comment.
y = “Hello”                    # Another one.
z = 3.45
if z == 3.45 or y == “Hello”:
    x = x + 1
    y = y + “ World”    # String concat.
print x
print y
```

Τα απαραίτητα για την κατανόηση του κώδικα

- **Η ανάθεση τιμής χρησιμοποιεί το = ενώ οι συγκρίσεις με το ==**
- **Οι αριθμητικές πράξεις γίνονται με τα γνωστά σύμβολα + - * / %**
  - Το **+** χρησιμοποιείται και για συνένωση αλφαριθμητικών.
  - Το **%** χρησιμοποιείται και για μορφοποίηση (όπως και στην printf της C).
  - Το ****** χρησιμοποιείται για την ύψωση σε δύναμη.
- **Οι λογικοί τελεστές είναι λέξεις (`and, or, not`) και όχι σύμβολα**
- **Η κύρια εντολή εκτύπωσης είναι η `print`.**
- **Η κύρια εντολή εισαγωγής δεδομένων από το πληκτρολόγιο είναι η `input`.**
- **Η πρώτη ανάθεση τιμής σε μεταβλητή την δημιουργεί αυτόματα.**
  - Δεν υπάρχουν δηλώσεις τύπων για μεταβλητές.
  - Η Python υπολογίζει τους τύπους των μεταβλητών από μόνη της.

Βασική είσοδος και έξοδος

```
>>> name=input('Ποιο είναι τ\' όνομά σου;')
Ποιο είναι τ' όνομά σου;Κώστας
>>> name
'Κώστας'
```

```
>>> print('Hello ', name)
Hello  Κώστας
>>> help(input)
Help on built-in function input in module builtins:

input(prompt=None, /)
    Read a string from standard input.  The trailing newline is stripped.

    The prompt string, if given, is printed to standard output without a
    trailing newline before reading input.

    If the user hits EOF (*nix: Ctrl-D, Windows: Ctrl-Z+Return), raise EOFError.
    On *nix systems, readline is used if available.

>>>
```

Ln: 39 Col: 4

Βασικοί Τύποι

- **Ακέραιος (Int) - προεπιλογή για αριθμητικά δεδομένα**

```
z = 5 / 2# 2, ακέραια διαίρεση
```

  - Για να εκφραστεί ένας αριθμός στο δυαδικό σύστημα βάζουμε το πρόθεμα 0b, για το οκταδικό 0o και για το δεκαεξαδικό 0x.
  - Μετατροπές από το δεκαδικό σε δυαδικό, οκταδικό και δεκαεξαδικό σύστημα γίνονται με τις συναρτήσεις bin(), oct() και hex().
- **Κινητής υποδιαστολής (float)**

```
x = 3.456
```

- **Αλφαριθμητικά (str)**
  - Η χρήση των “” και ‘’ είναι η ίδια π.χ.“abc” ή ‘abc’
  - Μπορούν να συνυπάρχουν μέσα σε ένα αλφαριθμητικό, όπως “matt’s”
  - Χρήση τριπλών “” για αλφαριθμητικά πολλαπλών γραμμών ή για αλφαριθμητικά που περιέχουν μέσα τους “ και ’ π.χ.“““a‘b“c”””

Βασικοί Τύποι

- **Λογικός Τύπος (bool)**
  - Η τιμή True αντιστοιχεί στο 1, ενώ η False στο 0.

  >>> True + 4

  5

  >>> False – 4

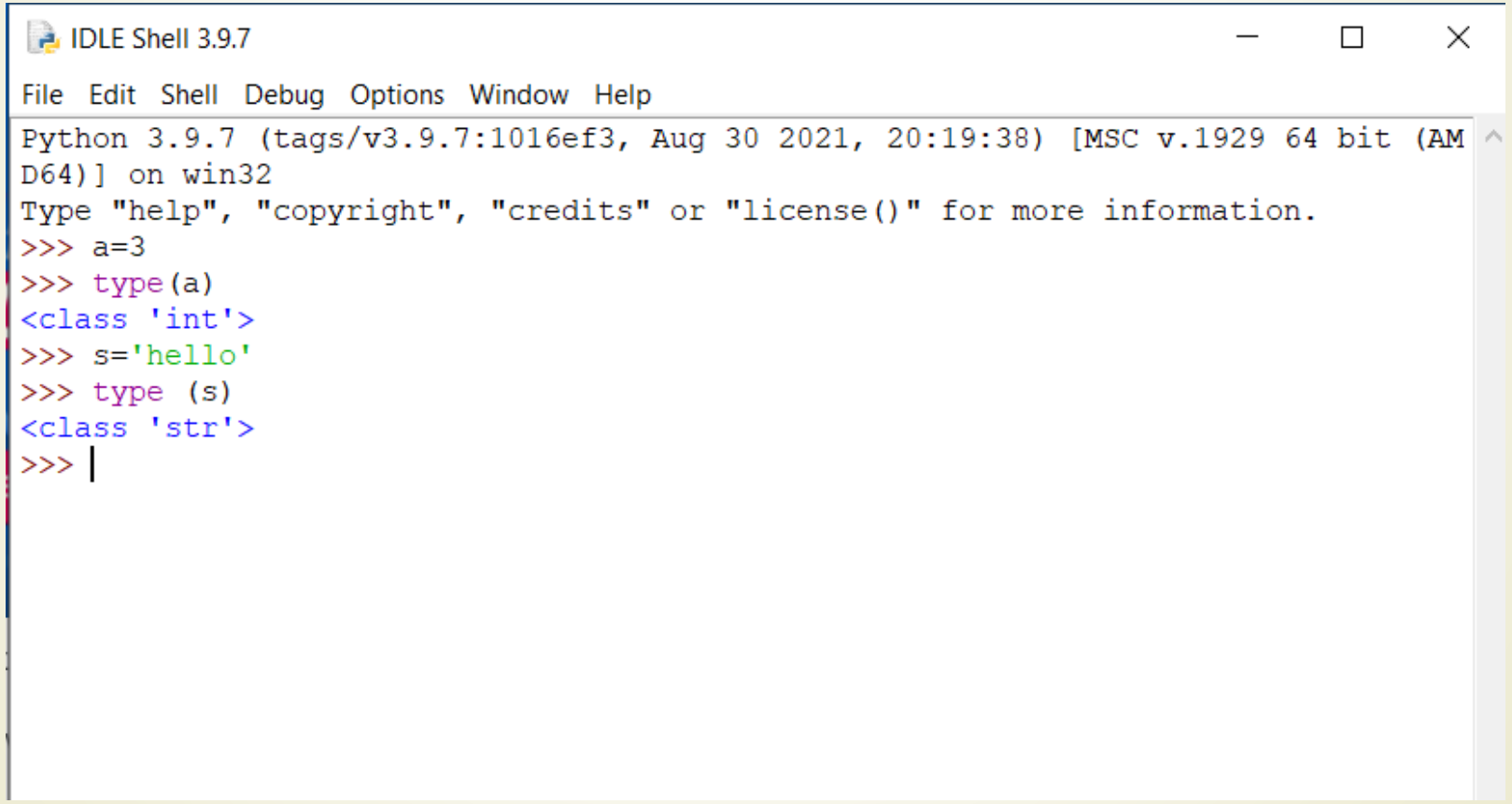
  -4

  - Υπάρχει ο τύπος NoneType με τιμή None.

Οι τύποι συγκεντρωτικά

Όνομα	Περιγραφή	Μεταβλητότητα
int	Προσημασμένος ακέραιος	Αμετάβλητος
float	Κινητής υποδιαστολής	Αμετάβλητος
bool	Λογική τιμή	Αμετάβλητος
str	Συμβολοσειρά	Αμετάβλητος
list	Λίστα	Μεταβαλλόμενος
tuple	Πλειάδα	Αμετάβλητος
set	Σύνολο	Μεταβαλλόμενος
frozenset	Αμετάβλητο σύνολο	Αμετάβλητος
dict	Λεξικό	Μεταβαλλόμενος

Έλεγχος τύπου



Συναρτήσεις μετατροπής τύπου

- int()
- float()
- bool()
- str()
- list()
- tuple()
- set()
- frozenset()
- dict()

Λευκός χώρος

- Αγνοούνται οι μη ορατοί χαρακτήρες.
- Στην αρχή μιας γραμμής όμως ο λευκός χώρος έχει συντακτική σημασία.
- Χρησιμοποιούμε το πλήκτρο “enter” για να οριοθετήσουμε το τέλος μιας γραμμής κώδικα.
- Χρησιμοποιούμε τον χαρακτήρα ‘\’ για να συνεχίσουμε τη συγγραφή μιας εντολής στην επόμενη γραμμή.
- Δεν χρησιμοποιούνται τα σύμβολα { } για να ορίσουν ένα τμήμα εντολών.
- Χρησιμοποιούνται εσοχές (λευκός χώρος) για να οριοθετήσουν εντολές που ανήκουν στο ίδιο τμήμα εντολών.
- Η πρώτη εντολή με εσοχή μικρότερη από τις παραπάνω δεν ανήκει στο τμήμα αυτό των εντολών.
- Η πρώτη γραμμή εντολής με μεγαλύτερη εσοχή ξεκινά ένα εμφωλευμένο τμήμα κώδικα.
- Συχνά εμφανίζεται το σύμβολο ‘:’ στην αρχή ενός τμήματος κώδικα (π.χ. στον ορισμό μιας συνάρτησης ή ενός αντικειμένου).

Σχόλια

- Ξεκινούν με το σύμβολο # - το υπόλοιπο της γραμμής αγνοείται.
- Μπορούν να συμπεριλάβουν ένα "αλφαριθμητικό τεκμηρίωσης" σαν πρώτη γραμμή κάθε νέας συνάρτησης ή κλάσης που ορίζεται.
- Ένα περιβάλλον ανάπτυξης κώδικα, ο διορθωτής λαθών, και άλλα εργαλεία το χρησιμοποιούν:

```
def my_function(x, y):
    """This is the docstring. This
    function does blah blah blah."""
    # The code would go here...
```

Τελεστές επαυξημένης εκχώρησης

- **+=** πρόσθεση και εκχώρηση.
- **-=** αφαίρεση και εκχώρηση.
- ***=** πολλαπλασιασμός και εκχώρηση.
- **/=** διαίρεση και εκχώρηση.
- **//=** διαίρεση και εκχώρηση.
- **/=** διαίρεση και εκχώρηση.
- **%=** υπόλοιπο και εκχώρηση.
- ****=** ύψωση σε δύναμη και εκχώρηση.

Υπερφόρτωση τελεστών

```
>>> 5+5
10
>>> “Γλώσσα “ + “Python”
‘Γλώσσα Python’
>>> a_list=[1,2,3]
[1,2,3]
>>> a_list=[1,2,3]+[4,5]
>>> a_list
[1,2,3,4,5]
```

Ανάθεση τιμής σε μεταβλητές

- Δέσμευση μιας μεταβλητής στην Python σημαίνει να θέτεις ένα όνομα για να διατηρήσεις μια αναφορά σε ένα αντικείμενο.
  - Η ανάθεση τιμής δημιουργεί αναφορές, όχι αντίγραφα.
- Τα ονόματα στην Python δεν έχουν εγγενώς τύπο, τα αντικείμενα έχουν τύπους.
- Η Python καθορίζει τον τύπο της αναφοράς αυτόματα, με βάση με το αντικείμενο δεδομένων που της ανατίθεται.
- Η δημιουργία του ονόματος γίνεται την πρώτη φορά που εμφανίζεται στην αριστερή πλευρά μιας έκφρασης ανάθεσης τιμής.
- Μια αναφορά διαγράφεται μέσω του μηχανισμού συλλογής σκουπιδιών (garbage collection) όταν όλα τα ονόματα που έχουν δεσμευτεί γι' αυτήν έχουν τεθεί εκτός εμβέλειας.

Προσπέλαση μη υπαρχόντων ονομάτων

- Μια προσπάθεια πρόσβασης σε όνομα πριν τον κατάλληλο ορισμό του (με την τοποθέτησή του στην αριστερή πλευρά μια εντολής ανάθεσης τιμής) οδηγεί σε σφάλμα:

```
>>> y


Traceback (most recent call last):
   File "<pyshell#16>", line 1, in -
   toplevel-  y
NameError: name 'y' is not defined
>>> y = 3
>>> y
3
```

Πολλαπλή ανάθεση τιμής

- Μπορεί να γίνει ανάθεση τιμής σε πολλαπλά ονόματα την ίδια στιγμή.

```
>>>   x, y = 2, 3
>>>   x
2
>>>   y
3
```

Κανόνες ονομάτων

- Υπάρχει διαχωρισμός κεφαλαίων και πεζών γραμμάτων.
- Δεν μπορούν να αρχίζουν με αριθμό.
- Μπορούν να περιέχουν γράμματα, αριθμούς και κάτω παύλες '_'
- Δεσμευμένες λέξεις (δεν μπορούν να χρησιμοποιηθούν ως ονόματα):

```
and, assert, break, class, continue, def, del, elif,  else,
except, exec, finally, for, from, global, if,  import, in, is,
lambda, not, or, pass, print, raise,  return, try, while
```