ΠΕΡΙΕΧΟΜΕΝΑ:

Κωνσταντίνος Σ.

Χρυσός Χορηγός Μαθήματος

Ιωάννης Π.

Χρυσός Χορηγός Μαθήματος

ΜΑΘΗΜΑ 2: Μεταβλητές και Τύποι Δεδομένων

1. Μεταβλητές

python 3 psounis

- Κάθε πρόγραμμα έχει την εξής δομή:
  - αποθηκεύει δεδομένα,
  - τα επεξεργάζεται και
  - εξάγει κάποιο αποτέλεσμα.
- Οι **μεταβλητές** είναι χώροι αποθήκευσης στη μνήμη. Τις χρησιμοποιούμε για να αποθηκεύσουμε δεδομένα
- Χαρακτηριστικά μιας μεταβλητής:
  - Έχει ένα **όνομα** (από τα σύμβολα A-z, 0-9, ξεκινάει με γράμμα και είναι case sensitive):
    - Καλό θα είναι να είναι περιγραφικό (όχι ξερά x και y)
    - Αν αποτελείται από δύο λέξεις, αυτές να χωρίζονται με underscore '_'
  - Έχει έναν **τύπο δεδομένων:**
    - Επιλέγεται αυτόματα από τα δεδομένα που σχετίζονται με αυτήν
    - Δεν έχει δήλωση, δημιουργείται με την πρώτη καταχώρηση.
    - Μπορεί να αλλάζει στη διάρκεια του προγράμματος.
    - Οι βασικοί τύποι δεδομένων είναι:
      - **Συμβολοσειρά (string),**
      - **Ακέραιος (integer),**
      - **Πραγματικός (float),**
      - **Λογική μεταβλητή (boolean).**
  - Για να θέσουμε τιμή σε μια μεταβλητή, γράφουμε
    - **Μεταβλητή = Τιμή** (έκχώρηση, καταχώρηση, ανάθεση)

Παράδειγμα 1: string.py

```
message = "Hello World"
print(message)
```

Παράδειγμα 2: integer.py

```
number = 5
print(number)
```

Παράδειγμα 3: float.py

```
number1 = 1.05
number2 = 4.99
result = number1+number2
print(result)
```

Παράδειγμα 4: boolean.py

```
is_pretty = False
print(is_pretty)
```

- Μία ακέραια μεταβλητή αναπαριστά έναν ακέραιο αριθμό
  - Οι ακέραιοι είναι …, -2, -1, 0, 1, 2, ...
  - Επιδέχονται πράξεις με τους τελεστές:
    - +, -, *: Πρόσθεση, αφαίρεση, πολλαπλασιασμός
    - /: Διαίρεση (προσοχή, ότι το αποτέλεσμα μπορεί να είναι πραγματικός)
    - //: Ακέραια Διαίρεση
    - %: modulo
    - **: Ύψωση σε δύναμη

Παράδειγμα 5: seconds.py

```
seconds = 3700

hours = seconds//3600
print(hours)

seconds = seconds % 3600
minutes = seconds//60
print(minutes)

seconds = seconds % 60
print(seconds)
```

Παράδειγμα 6: integer_operations.py

```
x = 3+5
print(x)
y = 4-x
print(y)
z = x*y
print(z)
div1 = 5/3
print(div1)
div2 = 5//3
print(div2)
mod = 5%3
print(mod)
power = 5**25
print(power)
```

Παρατηρήσεις

- Δεν υπάρχει περιορισμός στο μέγεθος του ακεραίου
- Παρατηρήστε ότι στο PyCharm ένας ακέραιος έχει μπλε χρώμα.

Άσκηση 1: Το 3

Γράψτε ένα πρόγραμμα στο οποίο να γίνεται μία πράξη πρόσθεσης, αφαίρεσης, πολλαπλασιασμού, διαίρεσης, ύψωσης σε δύναμη και modulo και το αποτέλεσμα να είναι 3
Π.χ. για την πρόσθεση:

```
print(1+2)
```

(Για την άσκηση αυτή να μη χρησιμοποιηθεί καμία απολύτως μεταβλητή)

Άσκηση 2: Προτεραιότητα Τελεστών

Η προτεραιότητα των τελεστών είναι ακριβώς η ίδια με τα μαθηματικά, δηλαδή από τη μεγαλύτερη στη μικρότερη προτεραιότητα:

- Ύψωση σε δύναμη
- Πολλαπλασιασμός, Διαίρεση
- Πρόσθεση, αφαίρεση

Εκτός κι αν αυτή καθορίζεται διαφορετικά από παρενθέσεις.

Εκτελέστε το παρακάτω πρόγραμμα, αφού πρώτα έχετε προβλέψει ποιο θα είναι το αποτέλεσμα:

```
print(1+2*3)
print(3/2*7)
print(3**2*7/2)
print(3-2*7+2)
print(3-2*(7+2))
```

Σημείωση

- Αν δεν είμαστε σίγουροι, σε μία περίπλοκη παράσταση, με ποια σειρά θα γίνουν οι πράξεις, καλό είναι να χρησιμοποιούμε παρενθέσεις.

- Οι **πραγματικές μεταβλητές** αναπαριστούν αριθμούς με υποδιαστολή
  - Η υποδιαστολή (τελεία) είναι αυτή που προσδιορίζει ότι ο αριθμός είναι πραγματικός.
    - Έτσι το 2 είναι ακέραιος, ενώ το 2.0 είναι πραγματικός.
  - Επιδέχονται πράξεις με τους τελεστές:
    - + ,- ,*, /, %, ** με την ίδια σημασία όπως στους ακεραίους

- Οι τελεστές για τροποποίηση τιμής μεταβλητής
  - Ορίζονται και τελεστές οι οποίοι είναι συντομογραφίες για την τροποποίηση της τιμής μιας μεταβλητής
  - Ο τελεστής +=. Π.χ. η παράσταση:

```
x += 0.2
```

είναι συντομογραφία της παράστασης:

```
x = x + 0.2
```

  - Αντίστοιχα ορίζονται οι τελεστές -=, *=, /=, //=
- Οι τελεστές αυτοί, ισχύουν και για ακέραιους αριθμούς.
- Ενώ μόνο ο += μπορεί να εφαρμοστεί και σε συμβολοσειρές.

Παράδειγμα 7: float_operations.py

```
x = 3.0
print(x)
y = 6.0
print(y)
x = x ** y
print(x)
```

Παράδειγμα 8: increase.py

```
x = 3.1
print(x)
x += 0.2
print(x)
```

Παρατηρήσεις

- Οι πραγματικοί αριθμοί δεν αναπαρίστανται «τέλεια», πρέπει πάντα να περιμένουμε ότι υπάρχει (πολύ μικρό μεν αλλά υπαρκτό δε) σφάλμα στην αναπαράσταση.
- Παρατηρήστε ότι στο PyCharm ένας πραγματικός έχει μπλε χρώμα.

Άσκηση 3: Τα λεφτά μας πίσω

- Η τράπεζα μας, μας προσφέρει ετήσιο επιτόκιο 0.01%
- Αυτό σημαίνει ότι αν έχουμε καταθέσει ένα ποσό Χ, τότε η τράπεζα, μετά από ένα χρόνο θα μας αυξήσει το ποσό κατά 0.01% (νέο ποσό = X + X*0.01)

Κατασκευάστε ένα πρόγραμμα Python το οποίο:

- Θα αποθηκεύει σε μία μεταβλητή το αρχικό μας ποσό (έστω 100 ευρώ)
- Θα αποθηκεύει σε μία μεταβλητή το επιτόκιο (έστω 0.01%)
- Θα τυπώνει το ποσό που θα έχουμε μετά από ένα χρόνο

Άσκηση 4: Δευτερόλεπτα

Θέλουμε να κατασκευάσουμε ένα πρόγραμμα που να υπολογίζει το πλήθος των δευτερολέπτων κάποιας χρονικής διάρκειας.
Π.χ. 3 λεπτά και 15 δευτερόλεπτα: είναι συνολικά 195 δευτερόλεπτα.
Το πρόγραμμα:

- Θα αποθηκεύει σε μια μεταβλητή πλήθος ωρών
- Θα αποθηκεύει σε μια μεταβλητή πλήθος λεπτών
- Θα αποθηκεύει σε μια μεταβλητή πλήθος δευτερολέπτων
- Θα υπολογίζει σε μία νέα μεταβλητή το συνολικό πλήθος δευτερολέπτων και έπειτα θα τυπώνει την τιμή αυτής της μεταβλητής.

1.3. Συμβολοσειρές

- Μία **συμβολοσειρά** είναι μία σειρά χαρακτήρων (συμβόλων)
  - Η συμβολοσειρά ορίζεται από τα διπλά εισαγωγικά στα οποία περικλείονται οι χαρακτήρες.
  - Τα εισαγωγικά μπορεί να είναι είτε διπλά είτε μονά
    - Έτσι οι ακόλουθες δύο παραστάσεις συμβολοσειρών είναι ισοδύναμες:

```
"Hello World!"
```

```
'Hello World!'
```

  - Αν χρειαστούμε μία συμβολοσειρά που περιέχει διπλά εισαγωγικά, πρέπει να την ενσωματώσουμε σε μονά εισαγωγικά, π.χ.:

```
‘He is “the” worst’
```

    - Ισχύει και το αντίθετο (μονά μέσα σε διπλά εισαγωγικά)
- **Συνένωση Συμβολοσειρών:**
  - Με το + μπορούμε να ενώσουμε δύο συμβολοσειρές σε μία καινούργια.

```
new_string = string1 + string2
```

Παράδειγμα 9: string_concatenation.py

```
name = "Alan"
surname = "Turing"
full_name = name + surname
print(full_name)
```

Παράδειγμα 10: string_concatenation2.py

```
full_name = "Alan" + " Turing"
print(full_name)
```

Παράδειγμα 11: string_concatenation3.py

```
name = "Alan"
surname = "Turing“
print(name + " " + surname)
```

Παρατηρήσεις

- Οι συμβολοσειρές έχουν πολλά ακόμη χαρακτηριστικά, που θα μελετήσουμε σε επόμενο μάθημα
- Παρατηρήστε ότι στο PyCharm μία συμβολοσειρά έχει πράσινο χρώμα.

Άσκηση 5: Ένα γνωστό γνωμικό

Ο Βασίλης Λεβέντης κάποτε είπε: «Πηγαίνετε να βοσκήσετε, προβατάκια του ΠΑΣΟΚ και της ΝΔ»

Κατασκευάστε συμβολοσειρά που να ενσωματώνει ολόκληρη την παραπάνω φράση και τυπώστε την (το γνωμικό πρέπει να είναι σε εισαγωγικά)

Άσκηση 6: Το νόημα μιας μεταβλητής

- Κατασκευάστε μια μεταβλητή με όνομα full_name
- Αρχικοποιήστε την με το όνομα σας
- Το πρόγραμμα να τυπώνει Hello, έπειτα το όνομά σας και τελικά ένα θαυμαστικό

Σχήμα του προγράμματος:

```
full_name = …
message = …
print(message)
```

Παράδειγμα εκτέλεσης:

```
Hello Dimitris Psounis!
```

Επαναλάβετε ακριβώς το ίδιο πρόγραμμα, με κάποιο άλλο όνομα.

- Οι **λογικές μεταβλητές (Boolean variables: τύπος Bool)** αναπαριστούν τις λογικές τιμές: Αλήθεια / Ψέματα
  - Αλήθεια: τιμή True
  - Ψέμα: τιμή False

- Οι λογικές μεταβλητές χρησιμοποιούνται για:
  - Να ελέγξουμε τη ροή ένός προγράμματος
  - Να αποθηκεύσουμε το αποτέλεσμα κάποιου ελέγχου
  - κ.α.
- Αυτά τα στοιχεία θα τα μελετήσουμε σε επόμενα μαθήματα.

Παράδειγμα 12: Bool.py

```
x = True
print(x)
x = False
print(x)
```

Παράδειγμα 13: comparison.py

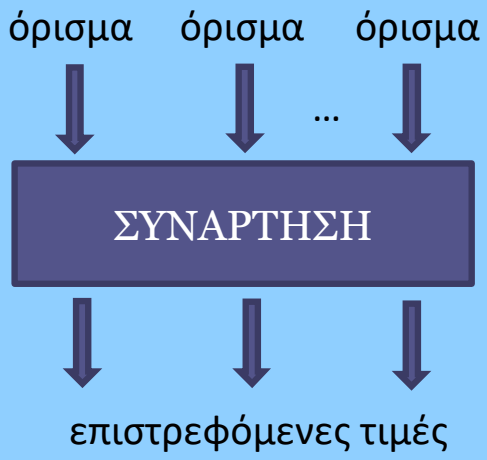
```
x = 3 < 5
print(x)

x = 1 < 0
print(x)
```

Παρατηρήσεις

- Οι τέσσερις τύποι δεδομένων που είδαμε είναι οι απλούστεροι που υπάρχουν και θα τους χρησιμοποιούμε κατά κόρον.
- Υπάρχουν όμως και άλλοι, πιο σύνθετοι τύποι δεδομένων, τους οποίους θα μελετήσουμε σε επόμενα μαθήματα, όπως:
  - οι λίστες (lists),
  - οι πλειάδες (tuples),
  - τα λεξικά (dictionaries),
  - τα σύνολα (sets) κ.α.

- Αν και θα αφιερώσουμε ξεχωριστό μάθημα για τις συναρτήσεις, θα πούμε λίγα πράγματα και εδώ:
  - Μία **συνάρτηση**:
    - δέχεται κάποια **ορίσματα** (0,1,2,…)
    - **επιστρέφει** κάποιο αποτέλεσμα (0,1,2,…)
  - Προσοχή, ότι με βάση τον ορισμό μία συνάρτηση μπορεί να μη δέχεται ορίσματα ή να μην παράγει επιστρεφόμενες τιμές.



- Το συντακτικό στην Python είναι:

```
ΣΥΝΑΡΤΗΣΗ(όρισμα, όρισμα, ..., όρισμα)
```

- Λέμε ότι «**καλούμε**» μια συνάρτηση «**διοχετεύοντας**» σε αυτήν τα ορίσματα.
- Όταν τελειώσει η κλήση, το αποτέλεσμα που επιστρέφεται αντικαθιστά την κλήση στον κώδικά μας.

Παρατηρήσεις

- Η print είναι μια συνάρτηση η οποία:
  - Δέχεται ως όρισμα αυτό που θέλουμε να τυπώσουμε
  - Έχει ως αποτέλεσμα να τυπωθεί αυτό στην κονσόλα
  - Προσοχή, όμως ότι δεν έχει επιστρεφόμενη τιμή.
- Πολύ σημαντικό:
  - Η εκτύπωση στην κονσόλα δεν είναι επιστρεφόμενη τιμή.
- Τι είναι η επιστρεφόμενη τιμή;
  - Θα την δούμε σε άλλες συναρτήσεις.
  - Π.χ. θα ορίσουμε σε επόμενο μάθημα τη συνάρτηση
    - sum(x,y)
  - Η οποία παίρνει δύο ορίσματα, τα προσθέτει, επιστρέφει το αποτέλεσμα (χωρίς να το τυπώνει)
  - Με χρήση αυτής της συνάρτησης μπορούμε π.χ. να γράψουμε την εντολή:

```
result = sum(3,5)
```

  - Προσοχή! Η συνάρτηση θα προσθέσει το 3 και το 5, και το αποτέλεσμα (8) θα αντικαταστήσει την έκφραση sum(3,5) άρα θα γίνει

```
result = 8
```

3. Μετατροπές Τύπων

- Όταν κάνουμε πράξεις μεταξύ μεταβλητών διαφορετικών τύπων δεδομένων
  - είτε θα έχουμε μια προβλέψιμη συμπεριφορά (π.χ. int + float)
  - είτε θα προκαλείται σφάλμα (π.χ. int + string)
- (βλ. παράδειγμα 14)

- Η Python μας επιτρέπει να κάνουμε μετατροπές τύπων χρησιμοποιώντας κατάλληλες συναρτήσεις
  - **int():** Μετατρέπει το όρισμα σε ακέραιο αριθμό, π.χ.:
    - ```
      int(3.84)
      ```
      αποκόπτει το δεκαδικό μέρος
  - **float()**: Μετατρέπει το όρισμα σε πραγματικό αριθμό
    - Γενικά θα ξέρουμε ότι πράξεις μεταξύ ακεραίων και πραγματικών επιστρέφουν πραγματικό.
  - **str()**: Μετατρέπει το όρισμα σε συμβολοσειρά
    - π.χ.
      ```
      str(5.19)
      ```
  - **bool()**: Μετατρέπει το όρισμα σε λογική μεταβλητή

- Με αυτές μπορούμε να κάνουμε μετατροπές ώστε να μην προκαλούνται σφάλματα:
  - π.χ. να συνενώσουμε μία συμβολοσειρά με την μετατροπή

Παράδειγμα 14: bugs_data_types.py

```
an_int = 5
a_float = 3.14
print(an_int + a_float)
a_string = "Hello!"
print(a_string + an_int)
```

Παράδειγμα 15: conversions.py

```
a_float = 3.84
to_int = int(a_float)
print(to_int)
to_str = str(a_float)
print(to_str)
```

Παράδειγμα 16: smart_message.py

```
age = 19
name = "John Doe"
message = name + " is " + str(age) + " years old."
print(message)
```
```

Άσκηση 7: Πειραματισμός με τις μετατροπές

Δοκιμάστε όλες τις πράξεις μεταξύ των βασικών τύπων και ερμηνεύστε το αποτέλεσμα.
Δηλαδή, δοκιμάστε να κάνετε πράξεις με τους εξής δυνατούς συνδυασμούς:

- int + float
- int + string
- int + bool
- float + string
- float + bool
- bool + string

4. Ο τελεστής εκχώρησης

- Λίγα λόγια για τον τελεστή εκχώρησης (ή καταχώρησης ή ανάθεσης):
  - Το συντακτικό του είναι:

```
ΜΕΤΑΒΛΗΤΗ = ΠΑΡΑΣΤΑΣΗ
```

  - Δηλαδή:
    - Αριστερά του τελεστή =, πρέπει να υπάρχει ακριβώς μία μεταβλητή
    - Δεξιά του τελεστή = , είναι μία παράσταση που μπορεί να είναι:
      - Μία τιμή, π.χ.:

```
x = 5
```

      - Μία άλλη μεταβλητή, π.χ.:

```
x = y
```

      - Το αποτέλεσμα πράξεων, π.χ.:

```
x = 8+y
```

      - Η επιστροφή μιας συνάρτησης, π.χ.:

```
x = int(my_str)
```

      - καθώς και οποιοσδήποτε συνδυασμός τους, π.χ.:

```
x = int(my_str) + 2.4
```

Παρατηρήσεις

- Γενικά με τον όρο παράσταση (έκφραση, expression), θα εννοούμε κάτι που υπολογίζεται.
  - Έτσι άτυπα όταν χρησιμοποιούμε τον τελεστή ανάθεσης είναι σαν λέμε:
    - **«Κάνε ότι χρειάζεται για να υπολογίσεις την Παράσταση, και έπειτα, το αποτέλεσμα, αποθήκευσε το στην Μεταβλητή»**
- Πολύ σημαντικό:
  - Τα ορίσματα μιας συνάρτησης είναι και αυτά παραστάσεις!
  - έτσι μπορούμε να βάζουμε μια υπολογιζόμενη παράσταση και ξέρουμε ότι πρώτα αυτή θα υπολογιστεί και έπειτα το όρισμα θα έχει την τιμή αυτής της παράστασης.

Παράδειγμα 17: brain_damage.py

```
print(str(int(5.12)+4)+"!")
```

Συντακτικές Παραλλαγές:

- Ανάθεση τιμών σε πολλές μεταβλητές σε μία γραμμή:

```
x , y = 3 , 5
```

- Ανάθεση ίδιας τιμές σε πολλές μεταβλητές:

```
x  = y = 3
```

- Η συνάρτηση input()
  - Παίρνει ένα όρισμα (που είναι μήνυμα προτροπής)
  - ο χρήστης πληκτρολογεί στην κονσόλα μία συμβολοσειρά
  - Επιστρέφει την συμβολοσειρά.
  - π.χ. για να πληκτρολογήσει ο χρήστης το όνομά του:

```
name = input("Type your name: ")
```

- Η συνάρτηση type()
  - Παίρνει ένα όρισμα (που είναι μία παράσταση)
  - Επιστρέφει τον τύπο δεδομένων της παράστασης, όπως αυτός αναπαρίσταται εσωτερικά στην Python
  - π.χ.

```
print(type(3))
```

    - Θα τυπώσει <class 'int'>
    - (που είναι η εσωτερική αναπαράσταση του ακεραίου)
    - Σε επόμενο μάθημα θα μάθουμε τι σημαίνει το class!

Παράδειγμα 18: magic_pill.py

```
name = input("Type your name: ")
surname = input("Type your surname: ")
age = int(input("Type your age: "))
magic_pill = 10
age -= magic_pill
message_name = "Hello " + name + " " + surname
message_age = ". You are " + str(age) + " years old!"
message = message_name + message_age
print(message)
```

Άσκηση 8:

- Κατασκευάστε ένα πρόγραμμα για να δείτε τις ονομασίες της εσωτερικής αναπαράστασης των 4 βασικών τύπων δεδομένων (ακέραιος, πραγματικός, συμβολοσειρά και λογική μεταβλητή)

Άσκηση 9: Περίμετρος και Εμβαδόν κύκλου

Κατασκευάστε ένα πρόγραμμα το οποίο θα δέχεται από την είσοδο την ακτίνα R ενός κύκλου και θα υπολογίζει και θα τυπώνει:

- Την περίμετρό του (από τον τύπο $2\pi R$)
- Το εμβαδόν του (από τον τύπο πR^2)
- Για τον υπολογισμό,θεωρήστε ότι π=3.14

Άσκηση 10: Μορφοποίηση ώρας

Κατασκευάστε ένα πρόγραμμα το οποίο θα δέχεται από την είσοδο πλήθος ωρών, πλήθος λεπτών και πλήθος δευτερολέπτων

- και τυπώνει στην οθόνη την ώρα στη μορφή: ΩΩ:ΛΛ:ΔΔ

Σημείωση:

- Αν και είναι αντιαισθητικό, η ώρα θα τυπώνεται π.χ. στην μορφή 1:1:8 αντί για το πιο κομψό και σύνηθες 01:01:08
- Αυτό θα το διορθώσουμε στο επόμενο μάθημα.

"Beautiful is better than ugly."

Zen of Python #1