Μεταβλητές, Εκφράσεις και Εντολές

2

Βασικά στοιχεία python

Σταθερές / literal values

- Σταθερές τιμές όπως αριθμοί, γράμματα και συμβολοσειρές ονομάζονται «σταθερές» διότι η τιμή τους δεν αλλάζει.
- Οι αριθμητικές σταθερές είναι οι αναμενόμενες.
- Οι αλφαριθμητικές σταθερές περικλείονται σε μονά εισαγωγικά (‘) ή σε διπλά εισαγωγικά (")

```
>>> print(123)
123
>>> print(98.6)
98.6
>>> print('Hello world')
Hello world
```

Δεσμευμένες Λέξεις

Δεν μπορείτε να χρησιμοποιήσετε δεσμευμένες λέξεις ως ονόματα μεταβλητών / αναγνωριστικών

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

Μεταβλητές

- Μια μεταβλητή είναι ένα όνομα αντιστοιχισμένο στη μνήμη, όπου ο προγραμματιστής μπορεί να αποθηκεύσει δεδομένα και αργότερα να τα ανακτήσει με αναφορά του «ονόματος» της μεταβλητής.
- Οι προγραμματιστές επιλέγουν τα ονόματα των μεταβλητών
- Μπορείτε να αλλάξετε το περιεχόμενο μιας μεταβλητής σε επόμενη εντολή

```
x = 12.2
y = 14
```

x	12.2
y	14

Μεταβλητές

- Μια μεταβλητή είναι ένα όνομα αντιστοιχισμένο στη μνήμη, όπου ο προγραμματιστής μπορεί να αποθηκεύσει δεδομένα και αργότερα να τα ανακτήσει με αναφορά του «ονόματος» της μεταβλητής.

- Οι προγραμματιστές επιλέγουν τα ονόματα των μεταβλητών

- Μπορείτε να αλλάξετε το περιεχόμενο μιας μεταβλητής σε επόμενη εντολή

```
x = 12.2
y = 14
x = 100
```

x | ~~12.2~~ 100

y | 14

Κανόνες Ονοματολογίας Μεταβλητών στην Python

- Πρέπει να αρχίζουν με γράμμα ή κάτω παύλα _
- Πρέπει να αποτελούνται από γράμματα, ψηφία και/ή κάτω παύλες.
- Με διάκριση Πεζών – Κεφαλαίων.

```
Καλό:       spam     eggs    spam23      _speed
Κακό:        23spam       #sign  var.12
Διαφορετικό:      spam    Spam    SPAM
```

Μνημονικά Ονόματα Μεταβλητών

- Μιας και, εμείς οι προγραμματιστές, έχουμε τη δυνατότητα επιλογής των ονομάτων των μεταβλητών μας, υπάρχει η λεγόμενη “βέλτιστη πρακτική”
- Ονομάζουμε μεταβλητές με τέτοιο τρόπο ώστε να θυμόμαστε τι σκοπεύουμε να αποθηκεύσουμε σε αυτές ("μνημονικό" = "βοήθημα μνήμης")
- Αυτό μπορεί να μπερδέψει τους αρχάριους επειδή οι καλά ονομασμένες μεταβλητές συχνά «ακούγονται» τόσο καλά νομίζουν ότι είναι λέξεις-κλειδιά

```
x1q3z9ocd = 35.0
x1q3z9afd = 12.50
x1q3p9afd = x1q3z9ocd * x1q3z9afd
print(x1q3p9afd)
```

Τί κάνει αυτό το τμήμα κώδικα;

```python
x1q3z9ocd = 35.0
x1q3z9afd = 12.50
x1q3p9afd = x1q3z9ocd * x1q3z9afd
print(x1q3p9afd)
```

```python
a = 35.0
b = 12.50
c = a * b
print(c)
```

Τί κάνει αυτό το
τμήμα κώδικα;

```
x1q3z9ocd = 35.0
x1q3z9afd = 12.50
x1q3p9afd = x1q3z9ocd * x1q3z9afd
print(x1q3p9afd)
```

```
a = 35.0
b = 12.50
c = a * b
print(c)
```

Τί κάνει αυτό το τμήμα κώδικα;

```
ώρες = 35.0
ωρομίσθιο = 12.50
μισθός = ώρες * ωρομίσθιο
print(μισθός)
```

Προτάσεις ή Γραμμές

```
x = 2
x = x + 2
print(x)
```

Εντολή Εκχώρησης Τιμής
Εκχώρηση με έκφραση
Εντολή Εκτύπωσης

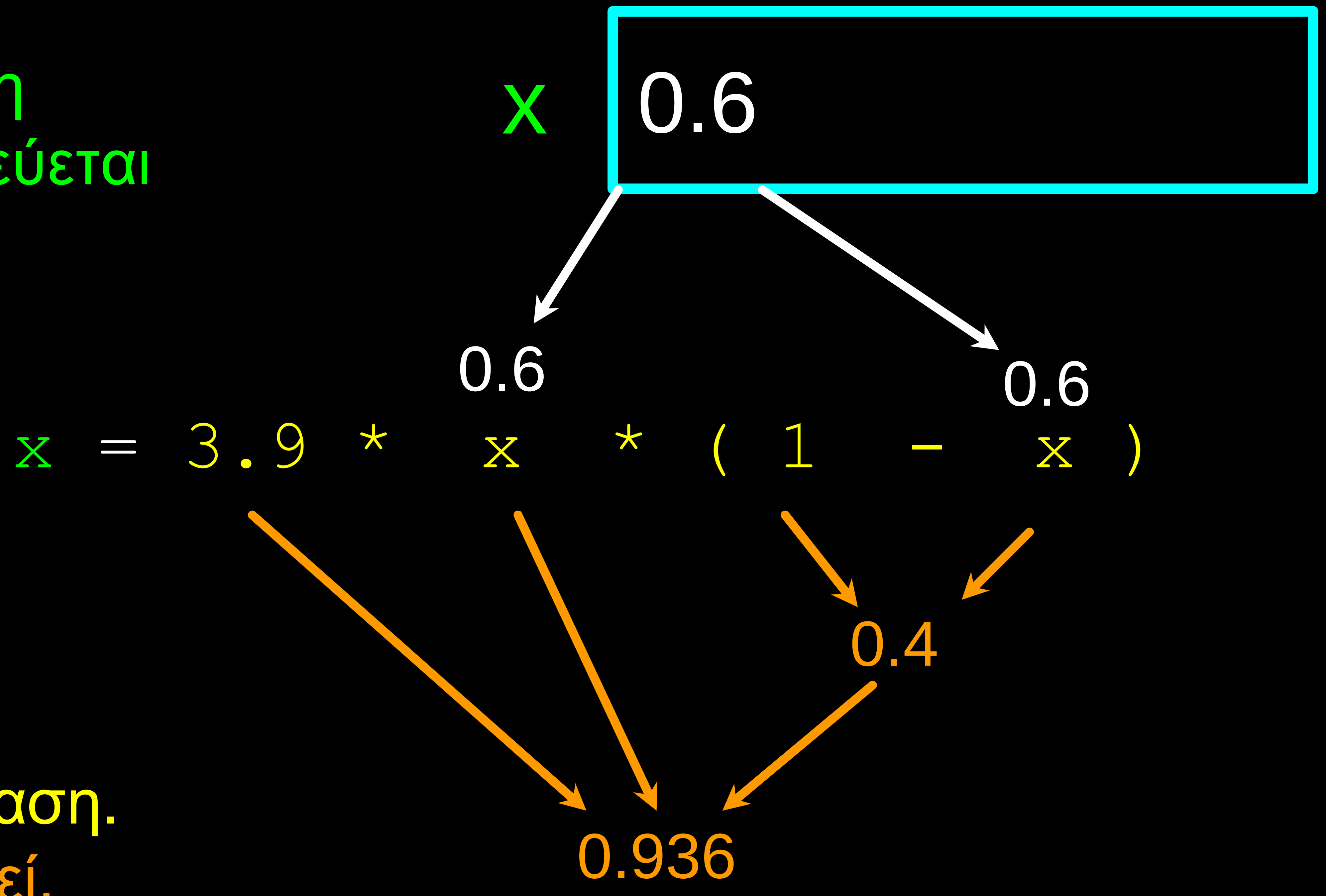
Μεταβλητή Τελεστής Σταθερά Συνάρτηση

Εντολές Εκχώρησης Τιμής

- Αναθέτουμε τιμή σε μια μεταβλητή χρησιμοποιώντας την εντολή εκχώρησης / ανάθεσης (=)
- Μια εντολή εκχώρησης αποτελείται από μια έκφραση στο δεξί μέλος και μια μεταβλητή στην οποία αποθηκεύεται το αποτέλεσμα

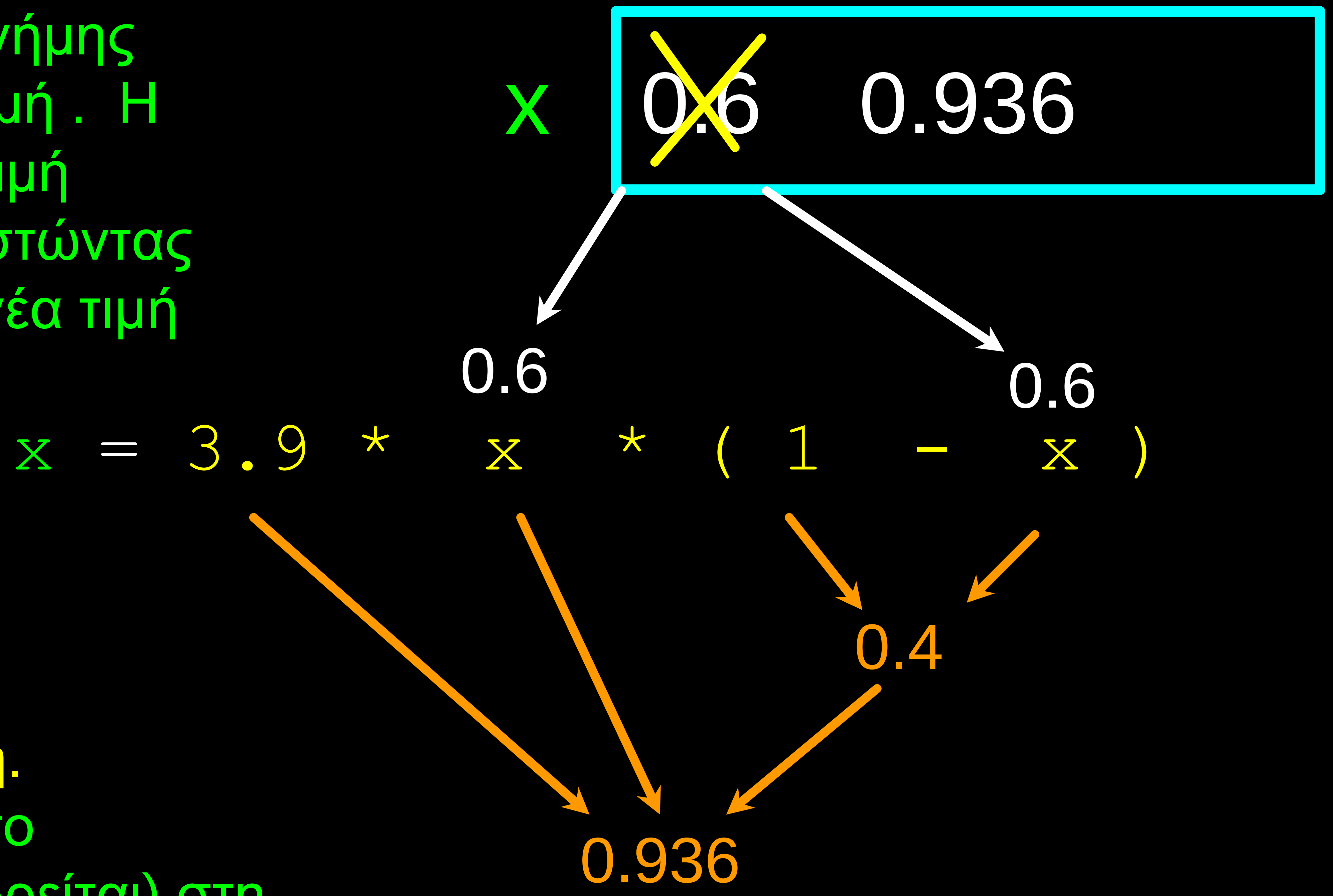
```
x = 3.9 * x * ( 1 - x )
```

Μια μεταβλητή είναι μια θέση μνήμης στην οποία αποθηκεύεται μια τιμή (0.6)



Το δεξί μέλος είναι μια έκφραση. Μόλις η έκφραση αξιολογηθεί, το αποτέλεσμα τοποθετείται (εκχωρείται) στο x.

Μια μεταβλητή είναι μια θέση μνήμης στην οποία αποθηκεύεται μια τιμή . Η αποθηκευμένη στη μεταβλητή τιμή μπορεί να ενημερωθεί αντικαθιστώντας την παλιά τιμή της (0.6) με μια νέα τιμή (0.936).



Το δεξί μέλος είναι μια έκφραση.
Μόλις η έκφραση αξιολογηθεί, το αποτέλεσμα τοποθετείται (εκχωρείται) στη μεταβλητή του αριστερού μέλους (π.χ., x).

Εκφράσεις…

Αριθμητικές Εκφράσεις

- Λόγω της έλλειψης των μαθηματικών συμβόλων στα πληκτρολόγια του υπολογιστή - χρησιμοποιούμε τα διαθέσιμα σύμβολαπλήκτρα για να εκφράσουμε τις κλασικές μαθηματικές πράξεις
- Ο αστερίσκος είναι ο πολλαπλασιασμός
- Ο εκθέτης (ύψωση σε δύναμη) φαίνεται διαφορετικά από ότι στα μαθηματικά

Τελεστής	Πράξη
+	Πρόσθεση
-	Αφαίρεση
*	Πολλαπλασιασμός
/	Διαίρεση
**	Δύναμη
%	Υπόλοιπο

Αριθμητικές Εκφράσεις

```
>>> xx = 2
>>> xx = xx + 2
>>> print(xx)
4
>>> yy = 440 * 12
>>> print(yy)
5280
>>> zz = yy / 1000
>>> print(zz)
5.28
```

```
>>> jj = 23
>>> kk = jj % 5
>>> print(kk)
3
>>> print(4 ** 3)
64
```

```
23 | 5
20 | 4
---
 3
```

Τελεστής	Πράξη
+	Πρόσθεση
-	Αφαίρεση
*	Πολλαπλασιασμός
/	Διαίρεση
**	Δύναμη
%	Υπόλοιπο

Προτεραιότητα Εκτέλεσης

- Όταν συνδέουμε τελεστές στην ίδια έκφραση - η Python πρέπει να ξέρει ποια πράξη να κάνει πρώτα
- Αυτό ονομάζεται «προτεραιότητα τελεστή»
- Ποιος τελεστής «προηγείται» έναντι των άλλων;

```
x = 1 + 2 * 3 - 4 / 5 ** 6
```

Κανόνες Προτεραιότητας Τελεστών

Υψηλότερη προτεραιότητα προς Χαμηλότερη προτεραιότητα:

- Οι παρενθέσεις προηγούνται πάντα
- Δύναμη (ύψωση σε δύναμη)
- Πολλαπλασιασμός, Διαίρεση και Υπόλοιπο
- Πρόσθεση και Αφαίρεση
- Από αριστερά προς τα δεξιά

Παρενθέσεις
Δύναμη
Πολλαπλασιασμός
Πρόσθεση
Αριστερά προς Δεξιά

```
>>> x = 1 + 2 ** 3 / 4 * 5
>>> print(x)
11.0
>>>
```

Παρενθέσεις
Δύναμη
Πολλαπλασιασμός
Πρόσθεση
Αριστερά προς Δεξιά

```
1 + 2 ** 3 / 4 * 5
1 + 8 / 4 * 5
1 + 2 * 5
1 + 10
11
```

Προτεραιότητα Τελεστών

Parenthesis
Power
Multiplication
Addition
Left to Right

- Θυμηθείτε τους κανόνες από πάνω προς τα κάτω
- Όταν γράφετε κώδικα - χρησιμοποιήστε παρενθέσεις
- Όταν γράφετε κώδικα - κρατήστε τις μαθηματικές εκφράσεις αρκετά απλές ώστε να είναι εύκολα κατανοητές
- Σπάστε μακροσκελείς μαθηματικές πράξεις για να γίνουν πιο σαφείς

Τι σημαίνει «Τύπος δεδομένων»;

- Στην Python μεταβλητές, είσοδος και σταθερές έχουν ένα «τύπο».
- Η Python γνωρίζει τη διαφορά ανάμεσα σε έναν ακέραιο και μια συμβολοσειρά.
- Για παράδειγμα το «+» σημαίνει «πρόσθεση» αν πρόκειται για αριθμούς και «συνένωση» αν αναφέρεται σε συμβολοσειρά.

```
>>> ddd = 1 + 4
>>> print(ddd)
5
>>> eee = 'hello ' + 'there'
>>> print(eee)
hello there
```

συνένωση = τοποθέτηση μαζί

Ο Τύπος έχει Σημασία

- Η Python γνωρίζει τι «τύπου» είναι το κάθε τι.
- Κάποιες πράξεις απαγορεύονται
- Δεν μπορείς να «προσθέσεις 1» σε μια συμβολοσειρά
- Μπορείς να ρωτήσεις την Python τι τύπου είναι κάτι χρησιμοποιώντας τη συνάρτηση type()

```
>>> eee = 'hello ' + 'there'
>>> eee = eee + 1
Traceback (most recent call last):
File "<stdin>", line 1, in
<module>TypeError: Can't convert
'int' object to str implicitly
>>> type(eee)
<class'str'>
>>> type('hello')
<class'str'>
>>> type(1)
<class'int'>
>>>
```

Διαφορετικοί Τύποι Αριθμών

- Οι αριθμοί έχουν δύο κύριους τύπους
  - Ακέραιοι - Integers είναι αριθμοί χωρίς δεκαδικά ψηφία: -14, -2, 0, 1, 100, 401233
  - Αριθμοί Κινητής Υποδιαστολής - Floating Point Numbers έχουν δεκαδικό μέρος: -2.5 , 0.0, 98.6, 14.0
- Υπάρχουν κι άλλοι αριθμητικοί τύποι- είναι παραλλαγές των float και integer

```
>>> xx = 1
>>> type (xx)
<class 'int'>
>>> temp = 98.6
>>> type(temp)
<class'float'>
>>> type(1)
<class 'int'>
>>> type(1.0)
<class'float'>
>>>
```

Μετατροπές Τύπου

- Όταν συνδυάζετε σε μια έκφραση ακεραίους και κινητής υποδιαστολής, ο ακέραιος σιωπηρά μετατρέπεται σε κινητής υποδιαστολής
- Ελεγχόμενα η μετατροπή γίνεται με τις ενσωματωμένες συναρτήσεις **int() και float()**

```
>>> print(float(99) + 100)
199.0
>>> i = 42
>>> type(i)
<class'int'>
>>> f = float(i)
>>> print(f)
42.0
>>> type(f)
<class'float'>
>>>
```

Διαίρεση Ακεραίων

Η Διαίρεση Ακεραίων παράγει αποτέλεσμα κινητής υποδιαστολής

```
>>> print(10 / 2)
5.0
>>> print(9 / 2)
4.5
>>> print(99 / 100)
0.99
>>> print(10.0 / 2.0)
5.0
>>> print(99.0 / 100.0)
0.99
```

Αυτό λειτουργούσε διαφορετικά στην Python 2.x

Μετατροπές Συμβολοσειρών

- Μπορείτε επίσης να χρησιμοποιήσετε τις int() και float() σε μετατροπές μεταξύ συμβολοσειρών και ακεραίων
- Θα προκύψει error / λάθος αν η συμβολοσειρά δεν περιέχει αριθμητικούς χαρακτήρες

```
>>> sval = '123'
>>> type(sval)
<class 'str'>
>>> print(sval + 1)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: Can't convert 'int' object
to str implicitly
>>> ival = int(sval)
>>> type(ival)
<class 'int'>
>>> print(ival + 1)
124
>>> nsv = 'hello bob'
>>> niv = int(nsv)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
ValueError: invalid literal for int()
with base 10: 'x'
```

Είσοδος από το Χρήστη

- Μπορούμε να δώσουμε εντολή στην Python να σταματήσει και να διαβάσει δεδομένα από το χρήστη χρησιμοποιώντας τη συνάρτηση input()
- Η συνάρτηση input() επιστέφει μια συμβολοσειρά

```
όνομα = input('Ποιος είσαι; ')
print('Καλωσήρθες', όνομα)
```

Ποιος είσαι; Chuck
Καλωσήρθες Chuck

Μετατροπή της Εισόδου από το Χρήστη

- Αν επιθυμούμε να διαβάσουμε έναν αριθμό από το χρήστη, πρέπει να τον μετατρέψουμε από συμβολοσειρά σε αριθμό χρησιμοποιώντας μια συνάρτηση μετατροπής τύπου
- Στη συνέχεια θα ασχοληθούμε με την περίπτωση εισαγωγής μη έγκυρων δεδομένων

```
inp = input('Όροφος στην Ευρώπη;')
usf = int(inp) + 1
print('Όροφος στις ΗΠΑ', usf)
```

Όροφος στην Ευρώπη; 0
Όροφος στις ΗΠΑ 1

Σχόλια στην Python

- Οτιδήποτε μετά από το # αγνοείται από την Python
- Γιατί Σχόλια;
  - Περιγράφουν τι πρόκειται να συμβεί σε με μια ακολουθία κώδικα
  - Τεκμηρίωση αυτού που έγραψε τον κώδικα ή άλλες βοηθητικές πληροφορίες
  - Ακυρώνει μια γραμμή κώδικα – ίσως προσωρινά

Σύνοψη

- Τύπος
- Δεσμευμένες λέξεις
- Μεταβλητές (μνημονικά)
- Τελεστές
- Προτεραιότητα τελεστών
- Διαίρεση ακεραίων
- Μετατροπές μεταξύ τύπων
- Είσοδος από το χρήστη
- Σχόλια (#)

Άσκηση

Γράψτε ένα πρόγραμμα που θα ζητά από το χρήστη ώρες και ποσό ανά ώρα, για τον υπολογισμό του ακαθάριστου μισθού.

```
Δώστε Ώρες: 35
Δώστε Ποσό/Ώρα: 2.75

Μισθός: 96.25
```