ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ Η/Υ
σε Python

Διάλεξη 1: Εισαγωγή

Μάρκος Ζάμπογλου, mzampoglou@aegean.gr

Αξιολόγηση εαρινού εξαμήνου 2025

"Εργαστήριο Εισαγωγή στην Πληροφορική"

&

"Επιστημονικός Προγραμματισμός σε Γλώσσα Python"

Θεωρία

"Δεν ήταν πάντα σαφής στην ανάλυση της θεωρίας με αποτέλεσμα να υπάρχουν κενά"

"Ο διδάσκοντας προσπαθεί να βγάλει πολύ μεγάλο όγκο ύλης σε μια διάλεξη και με αυτόν τον τρόπο καθιστά το μάθημα βαρετό"

Υλικό

"Θα ήθελα η διαδικτυακή τάξη (e-class) να είχε περισσότερα παραδείγματα ασκήσεων."

Ασκήσεις μέσα στο αμφιθέατρο

"μου άρεσε το γεγονός ότι παρόλο που στις διαφάνειες είχε λυμένες τις ασκήσεις ο καθηγητής, πρώτα τις έλυνε μαζί με τους φοιτητές στον πίνακα και μετά τις εξηγούσε και από τις διαφάνειες"

"να βάζει ασκήσεις να τις λύνουμε εκείνη την ώρα"

Crowd control

“”Ισως η φασαρία που υπάρχει μερικές φορές και το γεγονός ότι το μισό τμήμα μετά την παρουσία εξαφανίζεται”

“να υπάρξει η κατάλληλη διαχείριση από τον διδάσκοντα σε περιπτώσεις που υπάρχει φασαρία”

“δεν βάζαμε τις παρουσίες σε συγκεκριμένο χρόνο και επικρατούσε μια οχλαγωγία στο αμφιθέατρο”

Feedback

“Ανεβάζει βαθμούς στις εργασίες και ξέρουμε πώς τα πάμε.”

“Τρομερά βοηθητικό το γεγονός πως διόρθωνε τις εργασίες μας και έστελνε αναλυτικά τις λύσεις και και τις επισημάνσεις του σε ατομικό επίπεδο.”

...

“Να ανέβουν τα ντεσιμπέλ της φωνής”

“Κάποιες φορές μιλάει λίγο γρήγορα και δεν καταλαβαίνεις τι λέει.”

Οργάνωση μαθήματος

Δομή μαθήματος

- 3 ωρες θεωρία
  - Δευτέρα 12:00-15:00, Αμφιθέατρο
- 3 ώρες εργαστήριο
  - Τμήμα 1: Τρίτη 09:00-12:00, ΥΚ3 **(Από Α- έως Μπα-)**
  - Τμήμα 2: Τρίτη 18:00-21:00, ΥΚ3 **(Από Μπε- έως Ω-)**
    - Ενδέχεται να ενσωματωθεί στο Τμήμα 1

Διδάσκων

- Μάρκος Ζάμπογλου
  - mzampoglou@aegean.gr
  - Κουντουριώτου 41
- Ώρες γραφείου (ενδέχεται να αλλάξουν, ρωτήστε)
  - Δευτέρα 15:30-17:30
  - Τετάρτη 09:30-11:30

Βαθμολογία

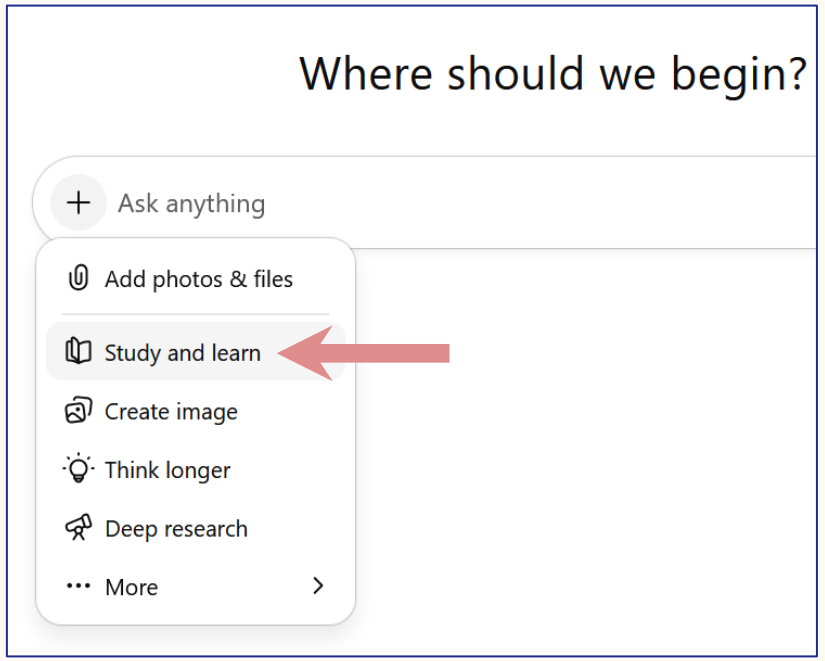
- Τελική εξέταση: **100%** του βαθμού
- Προαιρετικές εργαστηριακές ασκήσεις
  - Θα υποβάλλονται κατά τη διάρκεια του εργαστηρίου
  - **+1.5** μονάδα, αν ***βαθμός_εξετάσεων*** **> 5 ΚΑΙ** ***απουσίες_από_εργαστήριο*** **<= 2**
- Προαιρετικές ασκήσεις για το σπίτι
  - ...ακολουθεί εξήγηση

Homework στην εποχή των LLMs

- Μέχρι το 2022, η μαθησιακή αξία του homework συνίστατο στον κύκλο αποτυχία-επαναπροσπάθεια-διορθώσεις
- Τα LLMs σπάνε οριστικά αυτόν τον κύκλο σερβίροντας **έτοιμες απαντήσεις**
- Δεν πρόκειται να μάθουμε προγραμματισμό βλέποντας τον άψογο κώδικα άλλων, αλλά γράφοντας δικό μας *μέτριο* κώδικα, και αναστοχαζόμενες τα λάθη μας

Πρόταση: “Study mode”

- Αν είναι να χρησιμοποιείτε LLMs, τουλάχιστον κάντε το με τρόπο που να σας βοηθάει
  - Ιδανικά για τις ασκήσεις μην το χρησιμοποιείτε καθόλου
- **Study Mode**: Δε δίνει κατευθείαν τη λύση αλλά κάνει καθοδηγητικές ερωτήσεις
- Προσπαθεί να κατευθύνει τη μάθηση
- Σίγουρα προτιμότερο από το copy-paste
  - ChatGPT: “Study and learn”
  - Gemini: “Guided Learning”
  - …



Homework

Θα υποβάλετε προαιρετικά:

- Έναν αριθμό ασκήσεων από το σπίτι, όπως και στο ΕΕΠ

Θα λάβετε:

- Προσωποποιημένες διορθώσεις & σχολιασμό
- Προγραμματιστική εμπειρία

ΔΕΝ θα λάβετε:

- Βαθμολογικό όφελος

Το μάθημα στο e-Class

- Θεωρία:

203 - ΓΕ0107 - Προγραμματισμός Η/Υ

https://eclass.aegean.gr/courses/TMOD115/

- Εργαστήριο:

203- ΓΕ0107 Προγραμματισμός Η/Υ - Εργαστήρια

https://eclass.aegean.gr/courses/TMOD224/

Ενδέχεται να συμπτυχθούν. Θα υπάρξει ανακοίνωση στο eClass

Αναπληρώσεις

- Θα χρειαστεί να αναπληρώσουμε 2 θεωρίες και 4(!) εργαστήρια
- Θα προσπαθήσω να αποφύγω τις Κυριακές
  - Αλλά μάλλον με μέτρια επιτυχία

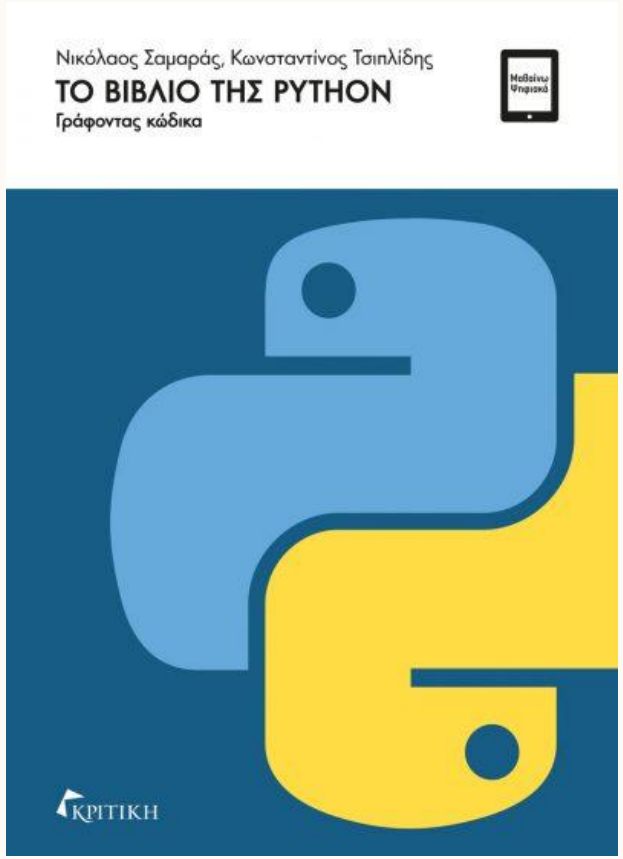
Σύγγραμμα μαθήματος

Νικόλαος Σαμαράς –
Κωνσταντίνος Τσιπλίδης

Το Βιβλίο της Python

Γράφοντας κώδικα

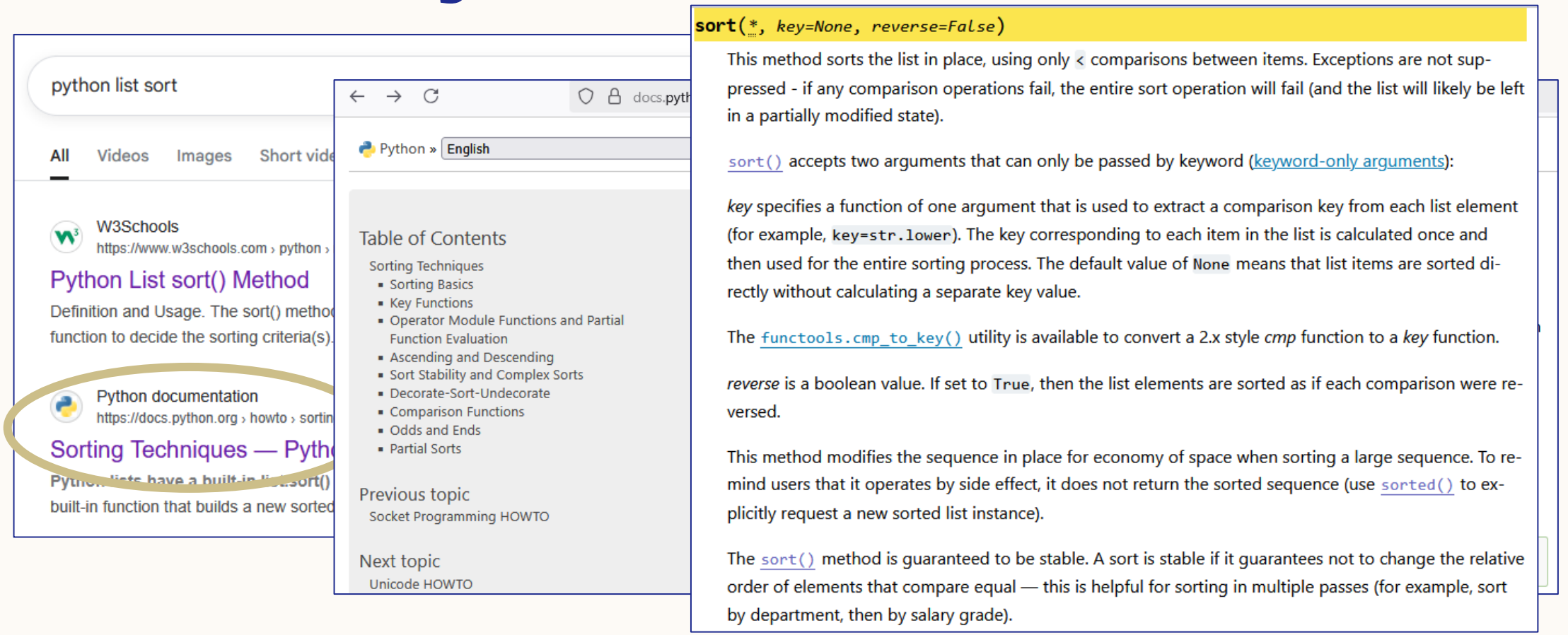
Εκδόσεις Κριτική



Online πηγές

- **W3Schools**
  - https://www.w3schools.com/python/
- Επίσημο Documentation της Python 3 (docs.python.org)
  - **Tutorial**: https://docs.python.org/3/tutorial/index.html
  - **Language reference**: καλύτερα μέσω μηχανής αναζήτησης

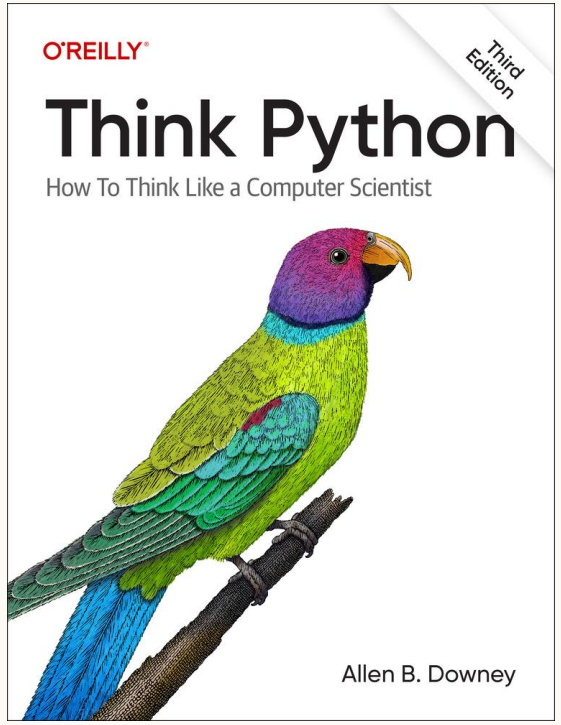
Python documentation



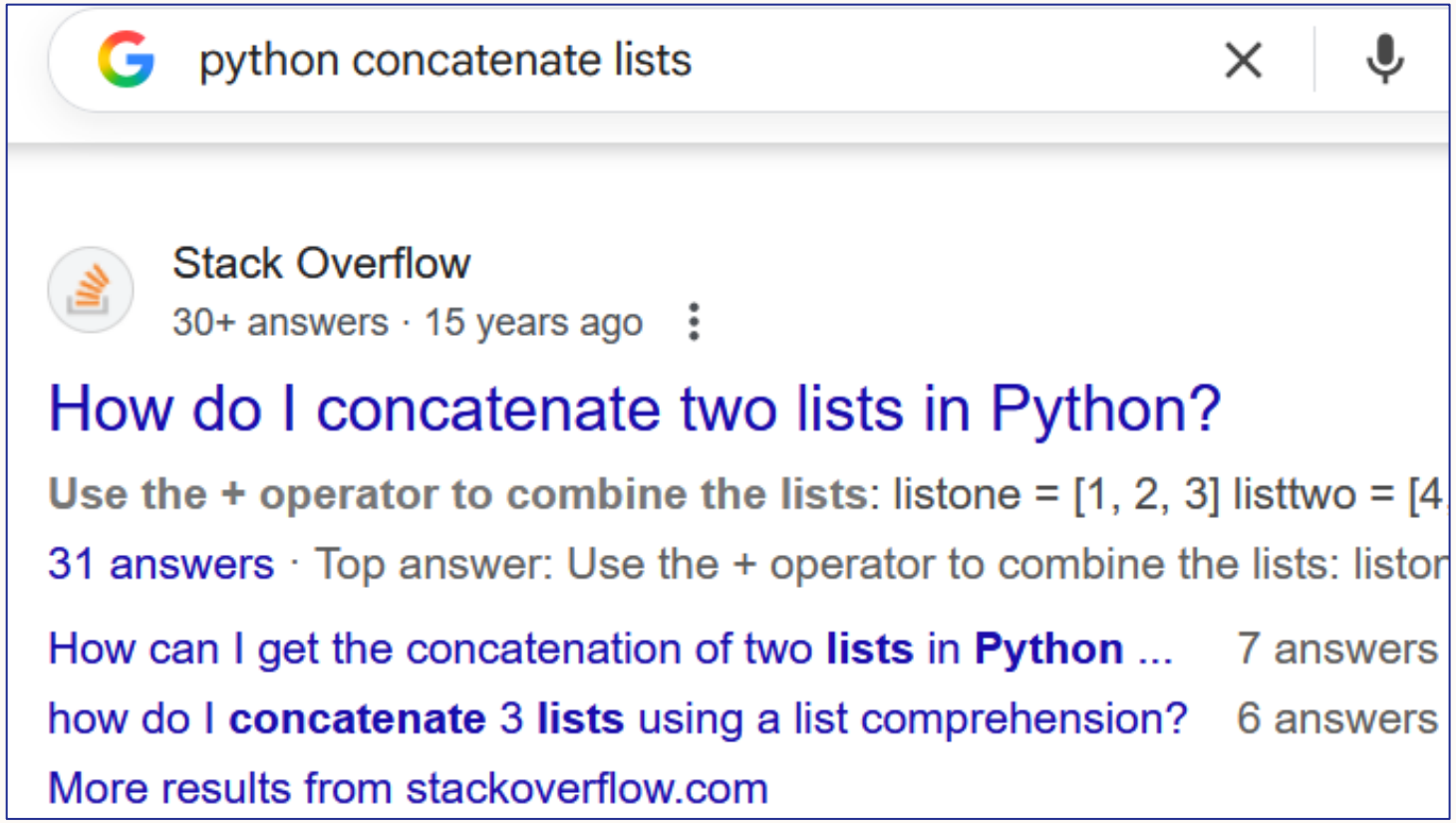
Βοηθητικό υλικό: Think Python

https://allendowney.github.io/ThinkPython/

Tutorials & κώδικας (σε Google Colab)



Πριν το ChatGPt: Stack Overflow



www.stackoverflow.com

Πριν το ChatGPt: Stack Overflow

How do I concatenate two lists in Python?

Asked 15 years, 9 months ago Modified 1 year, 7 months ago Viewed 4.8m times

3238

How do I concatenate two lists in Python?

Example:

```
listone = [1, 2, 3]
listtwo = [4, 5, 6]
```

Expected outcome:

```
>>> joinedlist
[1, 2, 3, 4, 5, 6]
```

python list concatenation

31 Answers

1 2 Next

5614

Use the `+` operator to combine the lists:

```
listone = [1, 2, 3]
listtwo = [4, 5, 6]

joinedlist = listone + listtwo
```

Output:

```
>>> joinedlist
[1, 2, 3, 4, 5, 6]
```

Ο προγραμματισμός είναι μια διαδικασία αυτο-μόρφωσης...

...που δεν τελειώνει ποτέ

Γενικά για την Python

Python: Ιστορικά

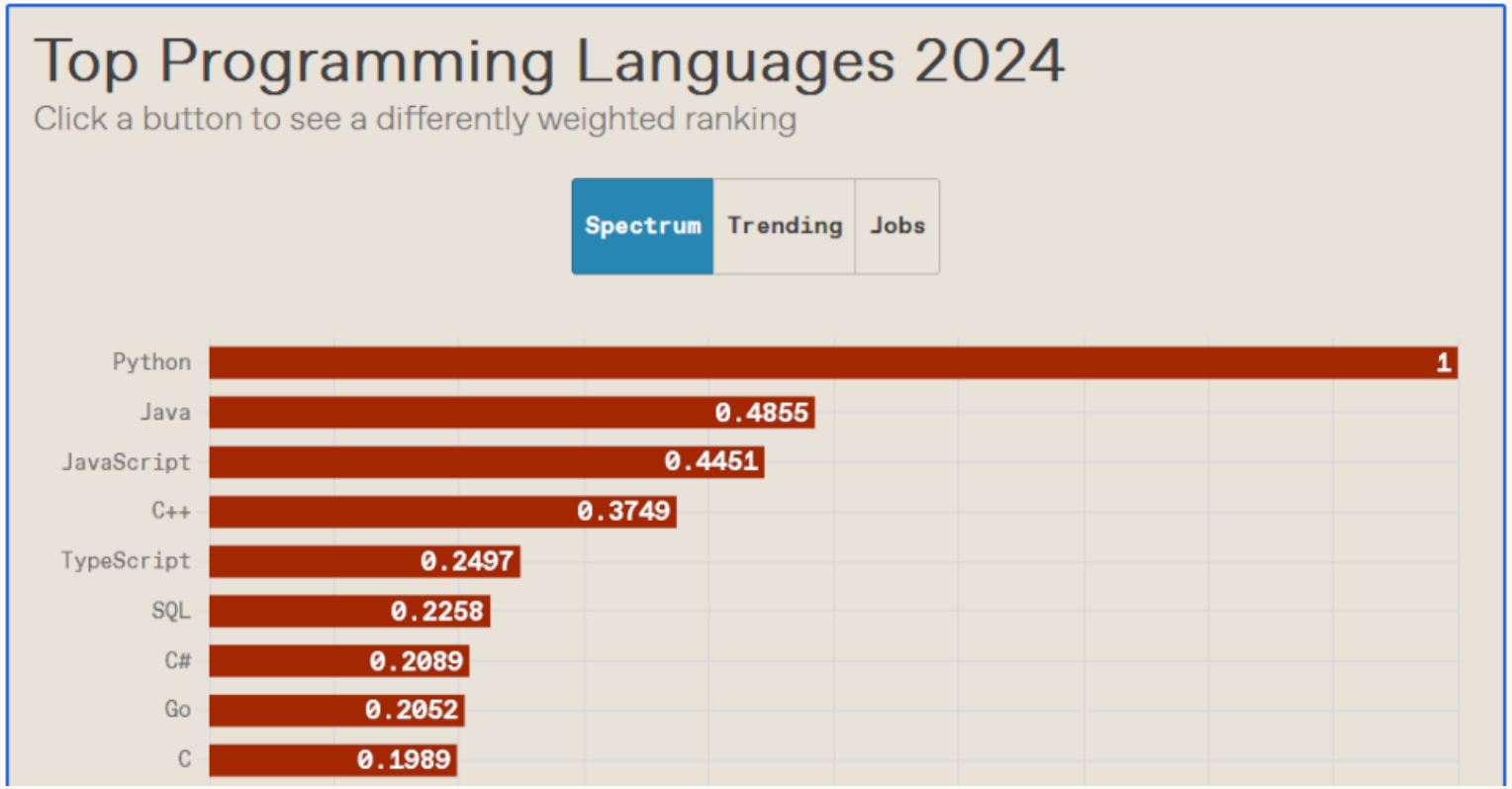
- Guido Van Rossum, 1989
  - Σχεδιασμένη για να είναι κοντά στην ανθρώπινη γλώσσα
  - Αλλά και ισχυρή (αντικειμενοστραφής, επεκτάσιμη)

- Python 2 (2000-2020) 🪦
  - 2.7. η τελευταία έκδοση
- Python 3 (2008-
  - 3.13.7 η τρέχουσα έκδοση

Τρέχουσα έκδοση

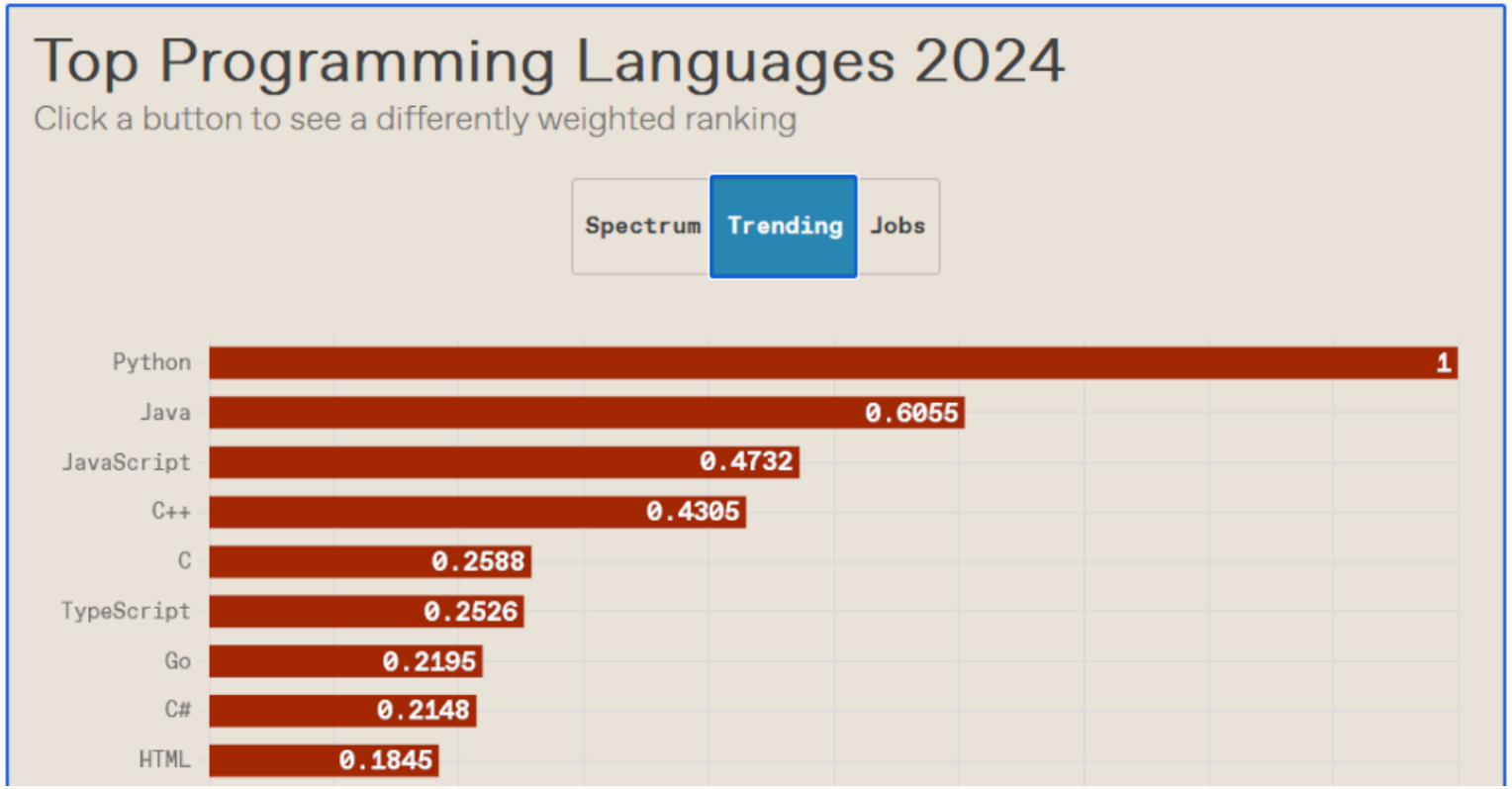
- **Python 3.13**
  - https://www.python.org/downloads/
    - Αν και είναι προτιμότερο να κατεβάσουμε κάποια διανομή όπως Anaconda
- Ακόμα μπορεί να βρεθεί κώδικας γραμμένος σε Python 2.7
  - Όχι 100% συμβατός με Python 3
  - Η υποστήριξή της έληξε το 2020

Γιατί Python;



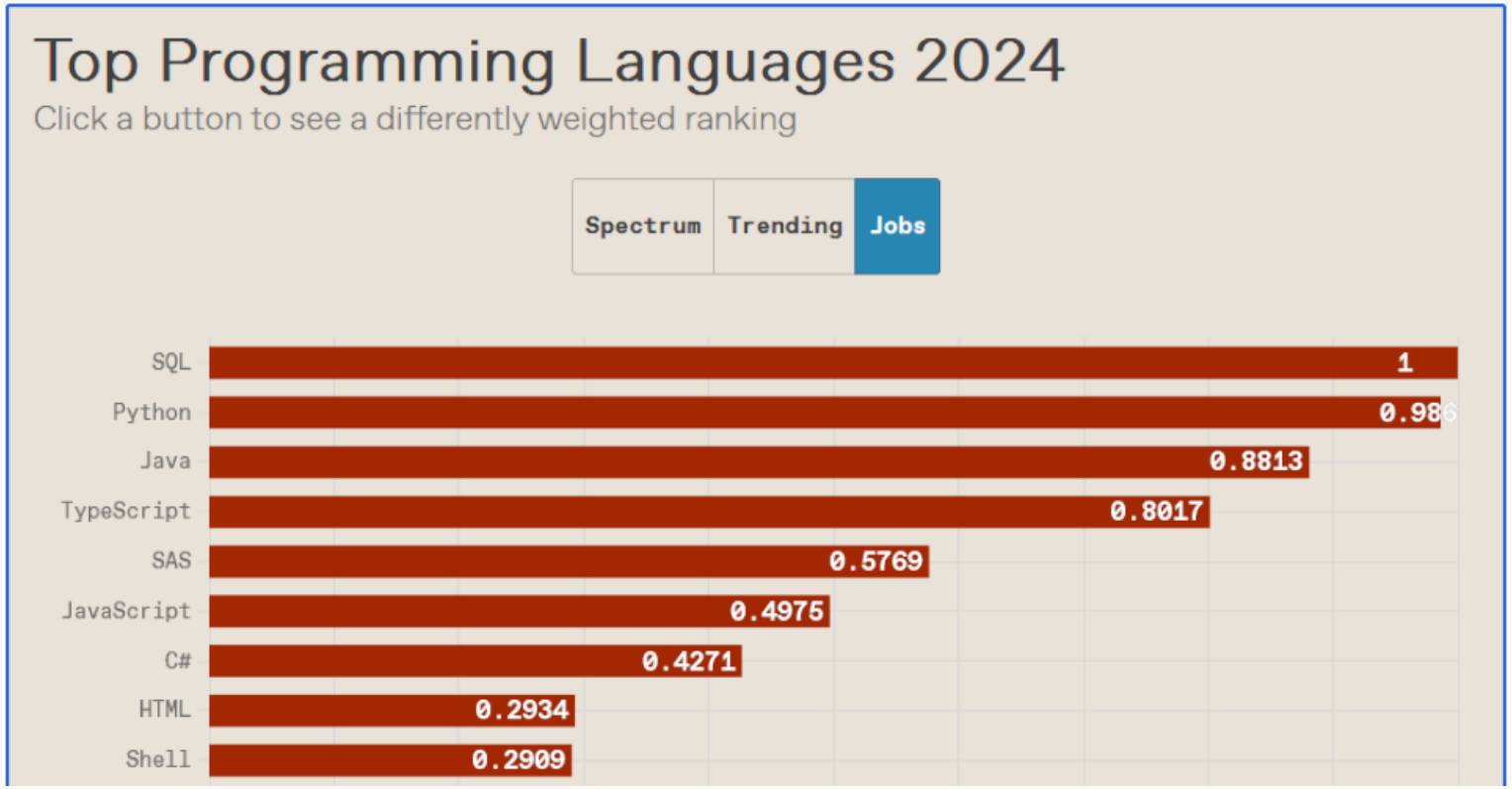
https://spectrum.ieee.org/top-programming-languages-2024

Γιατί Python;



https://spectrum.ieee.org/top-programming-languages-2024

Γιατί Python;



https://spectrum.ieee.org/top-programming-languages-2024

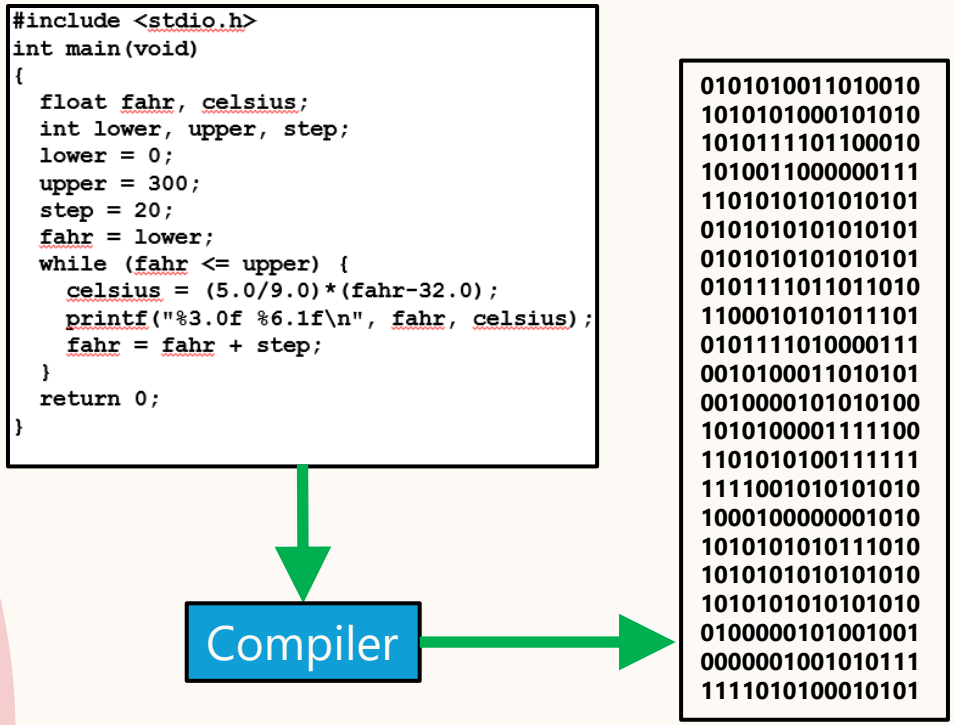
Γιατί Python;

1. Εύκολη στη μάθηση
2. Μεταφέρσιμη (Windows, Mac, Linux…)
3. **Πάρα** πολλές βιβλιοθήκες & επεκτάσεις
4. Γρήγορη ανάπτυξη κώδικα
5. Ευκολοδιάβαστος κώδικας
6. Επεκτάσιμη (με C/C++)
7. *Κατάλληλη για το είδος δουλειάς που αναλαμβάνουν οι απόφοιτοι του ΤΜΟΔ*

Compilers και Interpreters

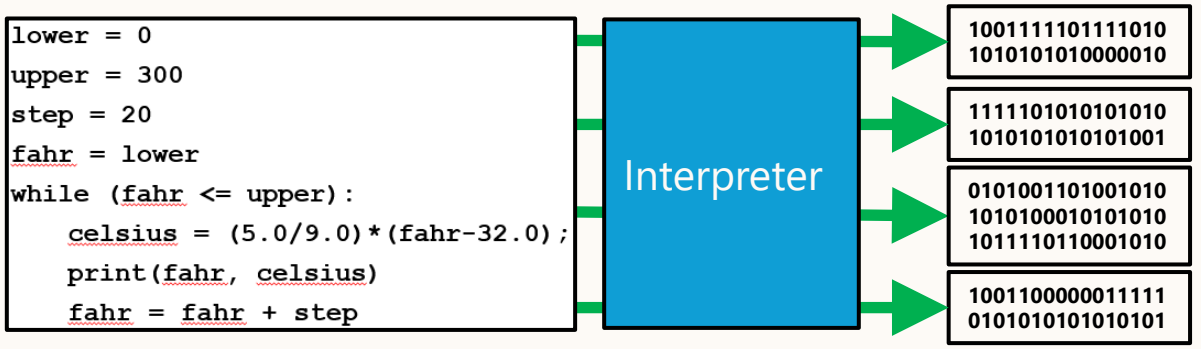
Μεταγλωττιστής (Compiler)

- Ο κώδικας μετατρέπεται σε ένα ενιαίο εκτελέσιμο αρχείο γλώσσας μηχανής



Διερμηνέας (Interpreter)

- Κάθε εντολή μεταφράζεται και εκτελείται χωριστά



Python Interpreter

Σε σχέση με τις compiled γλώσσες (π.χ. C, C++, C#), οι interpreted/script γλώσσες έχουν κάποιες ιδιαιτερότητες:

- Ο συντακτικός έλεγχος γίνεται γραμμή-γραμμή
  - Άρα δεν έχουμε εξαρχής έλεγχο όλου του προγράμματος
- Η εκτέλεση κάθε μιας γραμμής απαιτεί περισσότερο χρόνο
  - Γι' αυτό στην Python προσπαθούμε να αποφεύγουμε τις δομές επανάληψης
    - Θα επανέλθουμε σε αυτό
- Από την άλλη, το ότι είναι interpreted μας δίνει σοβαρά πλεονεκτήματα,
  - π.χ. να διακόπτουμε την εκτέλεση του προγράμματος για να δούμε την κατάσταση στη μνήμη

Οι πρώτες μου εντολές Python

Μπαίνω στο https://colab.research.google.com και φορτώνω (Upload) τον κώδικα "Διάλεξη 1"

- Εμφάνιση/Έξοδος πληροφορίας με την εντολή **`print()`**
- Σχόλια
- Ανάθεση τιμής σε μια μεταβλητή με το **`=`**
- Διάβασμα/Είσοδος πληροφορίας με την εντολή **`input()`**
- Πράξεις;
  - Σφάλματα…

Δομικά στοιχεία της Python

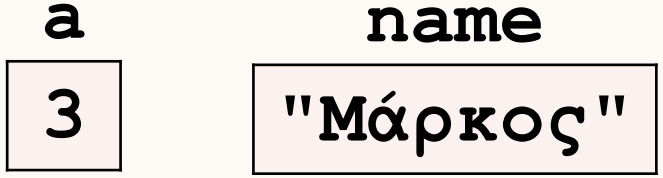
- Μεταβλητές
- Τελεστές
- Συναρτήσεις
- Δηλώσεις
- Εκφράσεις

Μεταβλητή (Variable)

Μια θέση μνήμης με

- Όνομα
- Τύπο
- Τιμή

Μπορούμε να φανταστούμε μια μεταβλητή ως ένα ονοματισμένο «κουτί» στη μνήμη, που χωράει μέσα της μια τιμή.

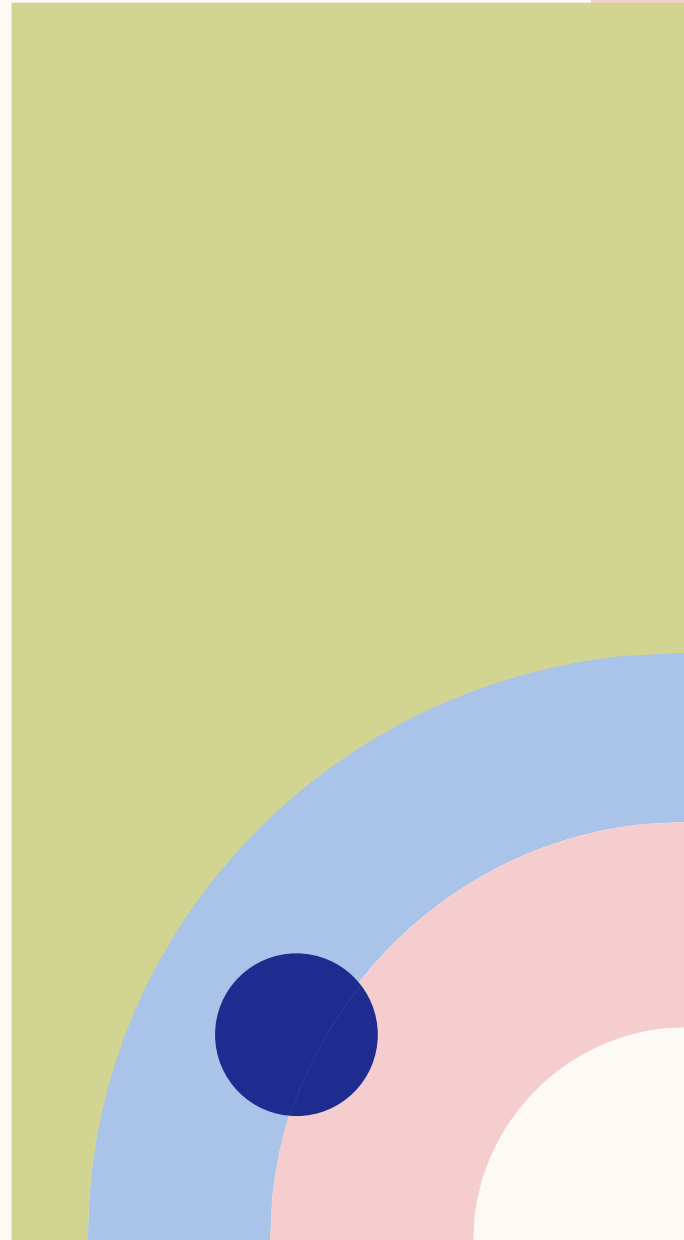


Την τιμή αυτή μπορούμε να την αναθέτουμε ή να την ανακαλούμε για να τη χρησιμοποιήσουμε, χρησιμοποιώντας το όνομά της

Μεταβλητή (Variable)

Μια θέση μνήμης με

- **Όνομα**
- Τύπο
- Τιμή



Ονοματοδοσία μεταβλητών

- Μια ακολουθία γραμμάτων και αριθμών (πεζών ή κεφαλαίων) και αριθμών
  - Αλλά πρέπει να ξεκινάει με γράμμα ή κάτω παύλα (**_**)
  - Case-sensitive: Η **`number1`** είναι διαφορετική από τη **`Number1`**
- Από σύμβολα, επιτρέπεται μόνο η κάτω παύλα (**_**)
- Δεν επιτρέπεται να έχουν όνομα **δεσμευμένης λέξης** της Python (π.χ. **`if`**, **`and`**)

Δεσμευμένες λέξεις της Python

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

Προσοχή!

- Τα ονόματα των συναρτήσεων της Python **δεν** αποτελούν δεσμευμένες λέξεις
  - Μπορούμε να δημιουργήσουμε μια μεταβλητή που θα λέγεται print

```
print=5
```

    - Όμως από εκεί και πέρα όποτε γράφουμε **`print`** η Python θα θεωρεί ότι εννοούμε τη μεταβλητή μας
    - Δε θα έχουμε πια τη δυνατότητα να τυπώσουμε κάτι στην οθόνη!
  - Τι είναι συναρτήσεις, πέρα από την **`print`** και την **`input`**;
    - Θα το δούμε στη συνέχεια

Ονοματοδοσία μεταβλητών

Ποια από τα παρακάτω είναι επιτρεπτά ονόματα;

- number2 ✓
- 2number ✗
- sum_store ✓
- True ✗
- true ✓
- mult$ ✗
- άθροισμα ✓
- _sum ✓

Μεταβλητή (Variable)

Μια θέση μνήμης με

- Όνομα
- **Τύπο**
- Τιμή

Κάθε δεδομένο που υπάρχει στην Python, είτε είναι μεταβλητή, είτε άλλο αντικείμενο, είτε απλά το αποτέλεσμα μιας εντολής/τελεστή, έχει πάντα έναν **τύπο**.

Τύποι δεδομένων στην Python

- **Numeric (Αριθμητικός)**
  - **Integer (Ακέραιος):** 0, 4, 1819, -40
  - **Float (Πραγματικός):** 6.3, 5.0, 519.32, -11.61
  - **Complex (Μιγαδικός):** 2+4j, 3.2-2.1j
- **Boolean (Λογικός):** True, False
- **String (Συμβολοσειρά):** "Hello World", "Καλημέρα", "4+4=8"

Μία τιμή, "*βαθμωτές*" ("*scalar*") μεταβλητές

- **List (Λίστα):** [2, 3.14, 'b', [33, 49], False]
- **Set (Σύνολο):** {2,3,4, 'Γιώργος', 3.14}
- **Tuple (Πλειάδα):** (3, 4, 'c')
- **Dictionary (Λεξικό):** {'Name': 'Markos', 'Height': 1.94}

Πολλές τιμές, "*σύνθετα*" ("*composite*") αντικείμενα από διάφορους άλλους τύπους

...και ένας ακόμα, το «τίποτα»

```
a = None
print(type(a))
<class 'NoneType'>
```

...**βλέπε κώδικα** για τη χρήση της συνάρτησης `type()`

...στην πραγματικότητα, πολλοί ακόμα

- Η Python είναι μια **αντικειμενοστραφής** γλώσσα
- Δεν έχουμε να κάνουμε με μεταβλητές, αλλά με **αντικείμενα**
- Μπορούμε να εισαγάγουμε επιπλέον τύπους αντικειμένων μέσα από **βιβλιοθήκες**

  Αντικείμενα: σύνθετες δομές δεδομένων, που ενσωματώνουν μαζί και το δικό τους κώδικα

- Μπορούμε επίσης να **δημιουργούμε** εμείς δικούς μας τύπους αντικειμένων
  - ...μελλοντικές διαλέξεις

Τελεστής (Operator)

- Ένα σύμβολο που επιτελεί μια πράξη ή ενέργεια. Π.χ.:
  - **+** σημαίνει «πρόσθεση»
  - **=** σημαίνει «ανάθεση τιμής»

Ο τελεστής ανάθεσης =

- Σημαίνει:

α) *Υπολόγισε την έκφραση στο δεξί μέρος*
β) Ανάθεσε το αποτέλεσμα στη μεταβλητή που αναφέρεται στο αριστερό μέρος

  - Παράδειγμα: `b=4+2`
- Στο δεξί μέρος μπορώ να έχω ό,τι θέλω
- Στο αριστερό μέρος **πρέπει να έχω το όνομα μιας μεταβλητής/αντικειμένου**
  - Ή περισσότερων, θα το δούμε αργότερα

= δε σημαίνει "ίσον"

- Το = στην Python δε σημαίνει "ίσον", σημαίνει "ανάθεση"
- Για αυτό επιτρέπεται να γράψω π.χ **a=a+2**
  1. *Υπολόγισε την έκφραση στα δεξιά*: πάρε την τιμή της **a** από τη μνήμη και πρόσθεσέ της **2**
  2. *Ανάθεσε το αποτέλεσμα* στην **a**
- Άρα τελικά: "αύξησε την τιμή της **a** κατά **2**"

Σειριακή και πολλαπλή ανάθεση

Σε αντίθεση με άλλες γλώσσες, η Python επιτρέπει σειριακή ανάθεση

- `a=b=c=12` Όλες οι μεταβλητές παίρνουν τιμή 12

Επιτρέπει επίσης πολλαπλή ανάθεση

- `a,b = 5,"Name"` Η a γίνεται 5 και η b γίνεται "Name"

Python vs άλλες γλώσσες

Έστω ότι έχω δυο αριθμούς αποθηκευμένους σε δυο μεταβλητές

```
a = 6
b = 3
```

και θέλω να τους αντιμεταθέσω έτσι ώστε η τιμή της a να πάει στη b και αντίστροφα

Τι θα γίνει αν τρέξω

```
a = b
b = a
```

```
a = 6
b = 3
a = b
b = a
```

```
a = 6
b = 3
a = b
b = a
```

Μνήμη

a ⟶ 6

```
a = 6
b = 3
a = b
b = a
```

Μνήμη

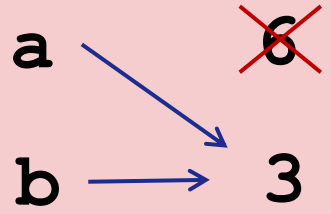
a ⟶ 6

b ⟶ 3

```
a = 6
b = 3
a = b
b = a
```

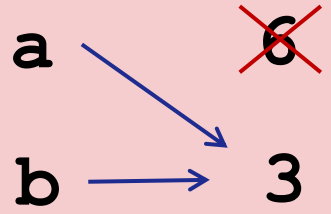
άρα 3

Μνήμη



```
a = 6
b = 3
a = b   άρα 3
b = a   άρα 3
```

Μνήμη



Παραδοσιακά σωστή λύση

```
a = 6
b = 3
c = b
```

→ χρησιμοποιώ μια προσωρινή, βοηθητική μεταβλητή για να μη χαθεί η παλιά τιμή της **b**

```
b = a
a = c
```

→ παίρνω την παλιά τιμή της **b** που την είχα κρατήσει στη **c**

Λύση με Python

```
a = 6
b = 3
a, b = b, a
```

→ η ανάθεση γίνεται και στις δυο μεταβλητές ταυτόχρονα, οπότε καμία δεν χάνεται πριν διαβαστεί

Αριθμητικοί τελεστές

Πράξεις μεταξύ αριθμών που επιστρέφουν ως αποτέλεσμα έναν αριθμό

+, **-** , ***** , **/** : πρόσθεση, αφαίρεση, πολλαπλασιασμός, διαίρεση

//: ακέραια διαίρεση, **%**: υπόλοιπο ακέραιας διαίρεσης, ******: δύναμη

- `5//2` επιστρέφει `2`
- `5%2` επιστρέφει `1`
- `5**2` επιστρέφει `25`

Συγκριτικοί τελεστές

Πράξεις μεταξύ αριθμών που επιστρέφουν boolean/λογικές τιμές

== , **<** , **>** , **<=** , **>=** , **!=** («όχι ίσο»)

Π.χ. **5<3** επιστρέφει **False**

ΠΡΟΣΟΧΗ!
Το == είναι ο τελεστής που σημαίνει 'σύγκριση για ισότητα'
Το = είναι ο τελεστής που σημαίνει 'απόδοση τιμής'

Λογικοί τελεστές

Πράξεις μεταξύ λογικών τιμών που επιστρέφουν λογικές τιμές

`and`, **`or`**, **`not`**

Π.χ. **`5>3 and 7<10`** υπολογίζεται ως **`True and False`** που επιστρέφει **`False`**

Οι πράξεις με λογικούς τελεστές μας δίνουν τη λογική άλγεβρα που θα τη δούμε σε επόμενες διαλέξεις σε μεγαλύτερο βάθος

Άλλοι τελεστές ανάθεσης

- **+=** πρόσθεση και μετά ανάθεση
  - Π.χ. **a+=3** σημαίνει "πρόσθεσε **3** στο **a**, και ανάθεσέ το στο **a**"
  - ...δηλαδή το ίδιο με το **a=a+3**
- αντίστοιχα **-=, *= , /= κλπ**

και πολλοί ακόμα...

https://www.w3schools.com/python/python_operators.asp

...βλέπε κώδικα για παραδείγματα

Συνάρτηση (Function)

- Ένα τμήμα κώδικα με **όνομα** και **παραμέτρους** που εκτελείται και **επιστρέφει** μια τιμή π.χ.
- **`abs(x)`** επιστρέφει την απόλυτη τιμή της μεταβλητής **`x`**
- **`print(x)`** εμφανίζει την τιμή της **`x`** στην οθόνη και επιστρέφει **`None`**

Ενσωματωμένες συναρτήσεις (Built-in Functions)

- **`print(x)`**, εμφανίζει την τιμή του **`x`**, επιστρέφει **`None`**
- **`input(x)`**, εμφανίζει την τιμή του **`x`**, περιμένει μια τιμή από το χρήστη, και επιστρέφει την τιμή που έδωσε ο χρήστης
- **`max(x,y,z,a…)`**, επιστρέφει το μέγιστο των **`x, y`** κλπ
- **`min(x,y,z,a…)`**, επιστρέφει τον ελάχιστο των **`x`** και **`y`** κλπ
- **`pow(x,y)`**, επιστρέφει τη δύναμη του **`x`** στην **`y`** (μοιάζει με τον τελεστή **`**`**)

Ενσωματωμένες συναρτήσεις (Built-in Functions)

- **`int(x)`**, μετατροπή τύπου. Επιστρέφει μια μεταβλητή τύπου **`int`** με το ακέραιο μέρος του **`x`**
- **`abs(x)`**, επιστρέφει την απόλυτη τιμή του **`x`**
- **`round(x)`**, επιστρέφει μια μεταβλητή τύπου int που περιέχει το αποτέλεσμα της στρογγυλοποίησης του **`x`**

...βλέπε κώδικα

https://docs.python.org/3/library/functions.html

Μετατροπή τύπου

- Όλες οι συναρτήσεις επιστρέφουν τιμές που ανήκουν σε ένα συγκεκριμένο τύπο δεδομένων
  - Π.χ. η συνάρτηση **`input()`** διαβάζει μια τιμή από το χρήστη και επιστρέφει ένα string
  - Η print() επιστρέφει **`None`**
- Για να μετατρέψουμε έναν τύπο δεδομένου σε έναν άλλον, χρησιμοποιούμε κάποιες ειδικές συναρτήσεις
  - Έχουν γενικά το ίδιο όνομα με τον τύπο
  - **`int()`, `float()`, `complex()`, `str()`, `bool()`, `list()`** κλπ
    - Δέχονται δεδομένα οποιουδήποτε τύπου, και επιστρέφουν δεδομένα ίδιου τύπου με το όνομά τους

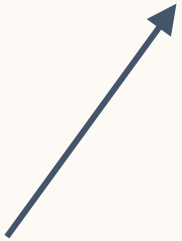
...βλέπε κώδικα

Η συνάρτηση `input()`

```
<έξοδος>=input(<μήνυμα σε string>)
```

π.χ.

```
a = input("Δώσε το όνομά σου: ")
```



Άνω και κάτω τελεία ακολουθούμενη από κενό οριοθετεί καλύτερα τις εισόδους οπτικά

Η συνάρτηση `print()`

- Με μία έκφραση μέσα στην παρένθεση:

```
a = 5
print(a*2)
```

- Με περισσότερες από μία εκφράσεις χωρισμένες με κόμματα:

```
a = 5
print("Το a είναι", a, "και το διπλάσιό του είναι", a*2)
```

Είσοδοι και έξοδοι στην πράξη

- Σε πραγματικές εφαρμογές, σπάνια χρησιμοποιούμε την **`input()`**
  - Μπορούμε να πούμε σχεδόν ποτέ
- Η **`print()`** από την άλλη, είναι χρήσιμη για να εμφανίζουμε δεδομένα ή ενδιάμεσα βήματα

- «Είσοδος» μπορεί να θεωρηθεί και ένα γραφικό interface (Graphical User Interface, GUI) που δέχεται εντολές από το χρήστη
  - Η Python παρέχει κάποιες δυνατότητες για ανάπτυξη εφαρμογών με GUI
  - ...περιορισμένης δημοφιλίας

Είσοδοι και έξοδοι στην πράξη

- Η Python χρησιμοποιείται πολύ συχνά για «επιστημονικές εφαρμογές»
  - Ανάλυση δεδομένων
  - Μηχανική μάθηση
  - Προσομοιώσεις/μοντελοποίηση (φυσική, μηχανική)
  - Επεξεργασία σήματος

- Σε τέτοιες εφαρμογές, συνήθως δεν έχει νόημα η είσοδος δεδομένων με το χέρι
- Οπότε πιο συχνά οι είσοδοι του προγράμματος θα είναι:
  - Αρχεία στο δίσκο
  - Καταχωρήσεις σε βάσεις δεδομένων
  - Online πηγές
    - Δηλ. πιθανότατα βάσεις δεδομένων ή αρχεία, αλλά απομακρυσμένα

Είσοδοι και έξοδοι στην πράξη

- Παρόλα αυτά οι **`input()`** και **`print()`** είναι πολύ χρήσιμες για να δοκιμάζουμε μικρά κομμάτια κώδικα και να σχεδιάζουμε εκπαιδευτικό υλικό
- Οι περισσότερες από τις ασκήσεις μας θα χρησιμοποιούν **`input()`** για είσοδο και **`print()`** για έξοδο

Δήλωση (Statement)

Μια δεσμευμένη λέξη της Python με συγκεκριμένη λειτουργία. Π.χ.:

- **`if`**: ελέγχει μια λογική τιμή, και αν είναι αληθής εκτελεί ένα μπλοκ κώδικα
- **`del`**: διαγράφει μια μεταβλητή από τη μνήμη

Η εντολή `del`

Διαγράφει ένα αντικείμενο από τη μνήμη

```
a=5
print(a)
del a
print(a)
```

Θα εμφανίσει:

```
5
Error
```

(επειδή η a δεν υπάρχει πια στη μνήμη)

...βλέπε κώδικα

Εκφράσεις (Expressions)

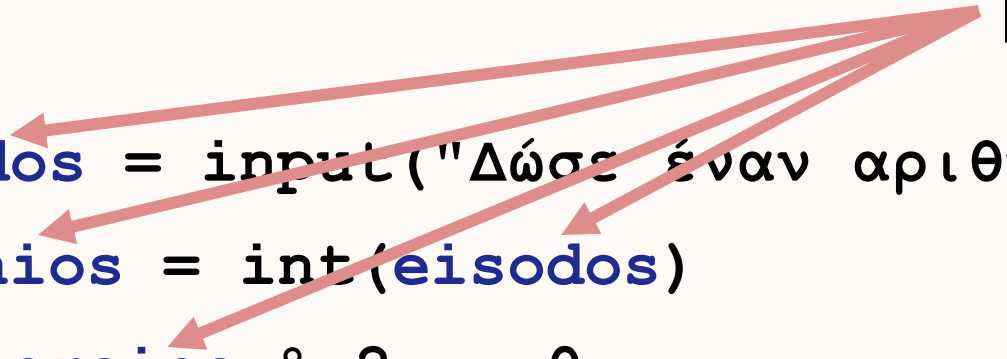
- Ένας συνδυασμός από **τιμές**, **μεταβλητές**, **τελεστές**, και κλήσεις σε **συναρτήσεις**
- Αφού υπολογιστεί, επιστρέφει μία τιμή
  - Αυτή η τιμή προφανώς έχει και τύπο
    - (Ο οποίος μπορεί να είναι π.χ. αριθμός αλλά μπορεί και λίστα)
- Π.χ. στην εντολή `a = 5 + abs(-3.2)`, το `5 + abs(-3.2)` είναι μια έκφραση που επιστρέφει `8.2` ενώ το `abs(-3.2)` είναι μια έκφραση που επιστρέφει `3.2`.

Ένα πρόγραμμα σε Python

```
eisodos = input("Δώσε έναν αριθμό: ")
akeraios = int(eisodos)
if akeraios % 2 == 0:
  print("Ο αριθμός είναι άρτιος")
else:
  print("Ο αριθμός είναι περιττός")
```

Ένα πρόγραμμα σε Python

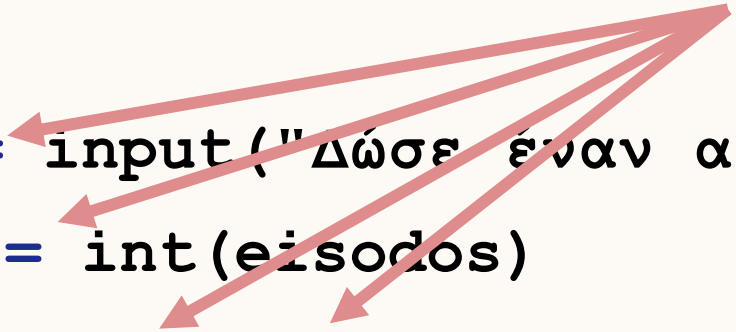
Μεταβλητές



```
eisodos = input("Δώσε έναν αριθμό: ")
akeraios = int(eisodos)
if akeraios % 2 == 0:
  print("Ο αριθμός είναι άρτιος")
else:
  print("Ο αριθμός είναι περιττός")
```

Ένα πρόγραμμα σε Python

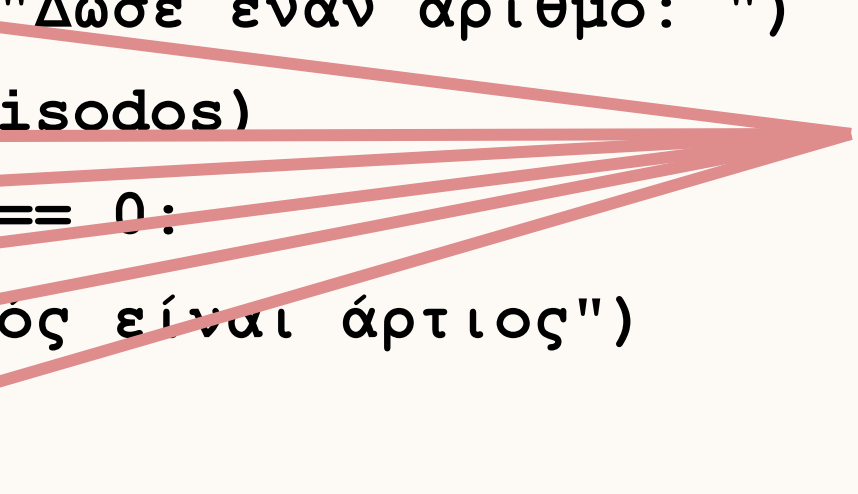
Τελεστές



```
eisodos = input("Δώσε έναν αριθμό: ")
akeraios = int(eisodos)
if akeraios % 2 == 0:
  print("Ο αριθμός είναι άρτιος")
else:
  print("Ο αριθμός είναι περιττός")
```

Ένα πρόγραμμα σε Python

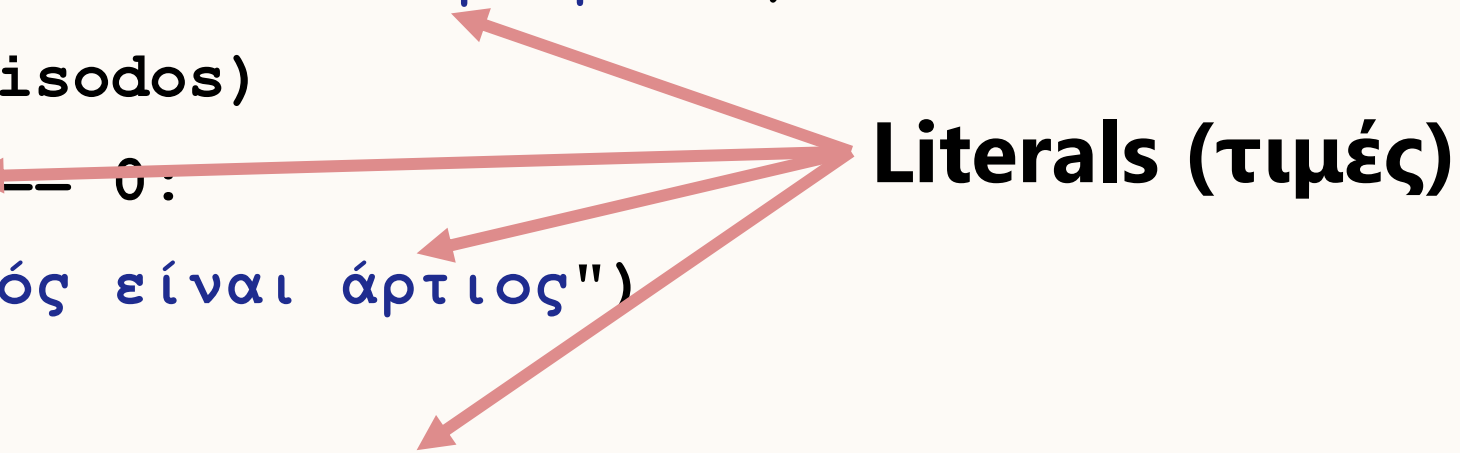
```
eisodos = input("Δώσε έναν αριθμό: ")
akeraios = int(eisodos)
if akeraios % 2 == 0:
  print("Ο αριθμός είναι άρτιος")
else:
  print("Ο αριθμός είναι περιττός")
```



Συναρτήσεις και Statements

Ένα πρόγραμμα σε Python

```
eisodos = input("Δώσε έναν αριθμό: ")
akeraios = int(eisodos)
if akeraios % 2 == 0:
  print("Ο αριθμός είναι άρτιος")
else
  print("Ο αριθμός είναι περιττός")
```

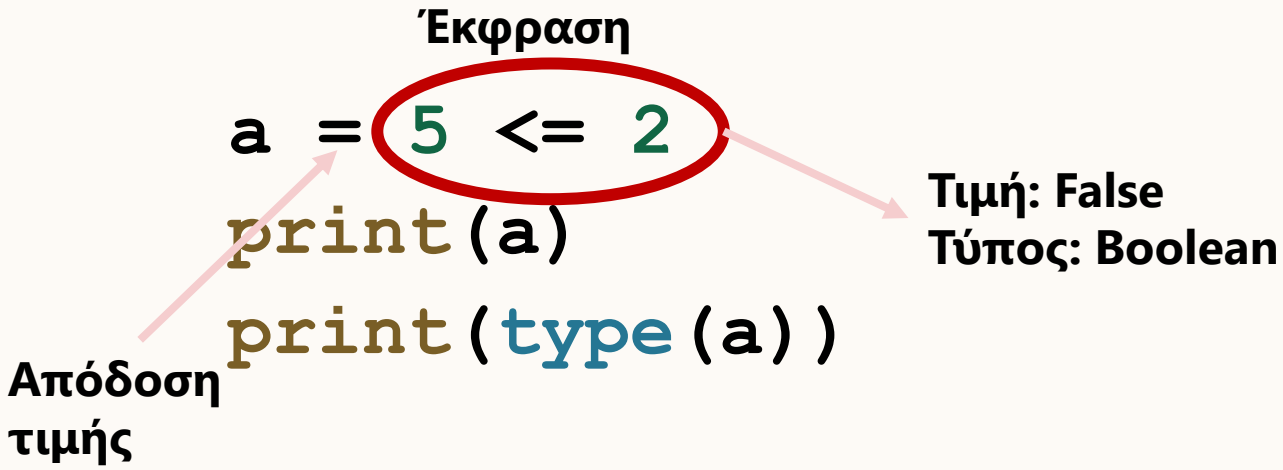


Εκτελώντας κώδικα στο χαρτί

```
a = 5 <= 2
print(a)
print(type(a))
```

τι θα εμφανίσει;

Εκτελώντας κώδικα στο χαρτί



Άρα θα εμφανίσει:

```
False
Boolean (για την ακρίβεια, <class 'bool'>)
```

Θεωρία: Dynamic Typing

- Στην Python ο τύπος δεδομένων των μεταβλητών προκύπτει δυναμικά κατά την ανάθεση τιμής
  - Δεν ορίζουμε τις μεταβλητές εκ των προτέρων
  - Οι μεταβλητές αλλάζουν τύπο όταν τους αναθέτουμε διαφορετικές τιμές
  - Ο τύπος των μεταβλητών λοιπόν δεν είναι *στατικός* (static), είναι *δυναμικός* (dynamic)