8. Η Γλώσσα Προγραμματισμού Python

Αρχές Γλωσσών Προγραμματισμού και Μεταφραστών

Γιάννης Γαροφαλάκης, Σπύρος Σιούτας, Παναγιώτης Χατζηδούκας, Γιάννης Βασιλόπουλος

Διαδικαστικά μαθήματος (1)

- Εισαγωγή στη Γλώσσα Προγραμματισμού **Python**
  - Κατανόηση των βασικών εννοιών της Python: Μεταβλητές, τύποι δεδομένων και δομές ελέγχου.
  - Εισαγωγή στη συνάρτηση και τη δομή του κώδικα: Σύνταξη, αναγνώριση σφαλμάτων και debugging.
- Προχωρημένα Θέματα στη Γλώσσα Προγραμματισμού **Python**
  - Χρήση προηγμένων βιβλιοθηκών, για τη διαχείριση δεδομένων.
  - Κατανόηση της έννοιας της κληρονομικότητας και των αντικειμενοστραφών προγραμματιστικών αρχών στην python.
- Εισαγωγή στην Επιστήμη των Δεδομένων με **Python** (φροντιστήριο)
  - Εισαγωγή στη NumPy
  - Εισαγωγή στη χρήση Pandas
  - Εισαγωγή στη Matplotlib

Η Γλώσσα Προγραμματισμού Python

Το 1989 ο Guido Van Rossum δημιούργησε τη γλώσσα Python.

Η ονομασία της δεν προέρχεται από το φίδι πύθωνα αλλά από το γεγονός ότι ο Guido λάτρευε το Monty Pythons Flying Circus.

- Το 1994 παρουσιάστηκε η έκδοση 1.0
- Το 2000 παρουσιάστηκε η έκδοση 2.0
- Το 2008 παρουσιάστηκε η έκδοση 3.0
- Σήμερα χρησιμοποιούμε την έκδοση 3.13.2

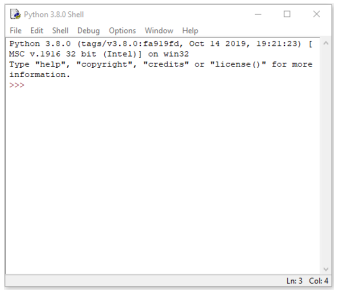
Χαρακτηριστικά της γλώσσας Python

- Η Python είναι μια γλώσσα με απλή και καθαρή σύνταξη, εύκολη στην ανάγνωση και τη συγγραφή, ιδανική για αρχάριους και προχωρημένους.
- Ο κώδικάς της είναι ανοικτού λογισμικού και διαθέτει μια μεγάλη, ενεργή παγκόσμια κοινότητα υποστήριξης.
- Υποστηρίζει πολλαπλά προγραμματιστικά πρότυπα, όπως:
  - Διαδικαστικός προγραμματισμός
  - Αντικειμενοστρεφής προγραμματισμός
- Είναι cross-platform: λειτουργεί σε Windows, macOS και Linux.

Εργαλεία συγγραφής κώδικα Python

Python IDLE

- Το προεγκατεστημένο περιβάλλον που έρχεται μαζί με την εγκατάσταση της Python.
- Πολύ ελαφρύ, χωρίς περίπλοκα μενού και ρυθμίσεις.



- Ιδανικό για την άμεση εκτέλεση μεμονωμένων εντολών και γρήγορους μαθηματικούς ελέγχους.
- Ένας απλός κειμενογράφος για τη συγγραφή βασικών αρχείων κώδικα (.py).

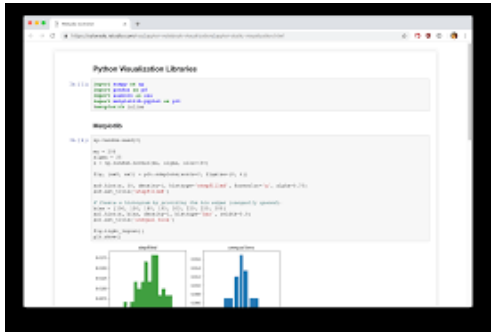
Google Colab (Cloud)

- Αρχιτεκτονική: Βασίζεται στο οικοσύστημα των Jupyter Notebooks, φιλοξενούμενο σε υποδομές cloud της Google.
- Πόροι Συστήματος: Παροχή virtual μηχανών με προ-εγκατεστημένα data science runtimes και πρόσβαση σε επιταχυντές υλικού (Hardware Accelerators) όπως GPUs/TPUs, διευκολύνοντας την εκτέλεση υπολογιστικά απαιτητικών μοντέλων.

- Backend Integration: Άμεση διασύνδεση με το Google Drive για persistency των δεδομένων και το GitHub για version control.
- Περιορισμοί: εξάρτηση από τη σταθερότητα της δικτυακής σύνδεσης.

Jupyter Notebooks

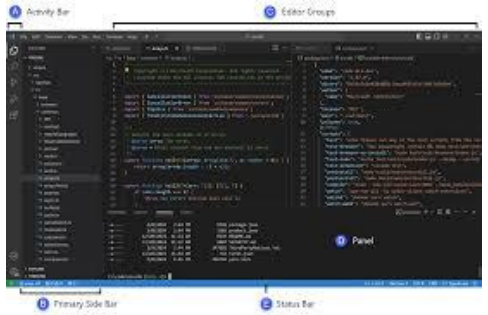
- Συνδυασμός κώδικα, στατιστικών αποτελεσμάτων και επεξηγηματικού κειμένου σε ένα ενιαίο αρχείο (.ipynb).
- **Τμηματική Εκτέλεση (Cells):** Ο κώδικας χωρίζεται σε ανεξάρτητα τμήματα. Μπορούμε να τρέξουμε μόνο έναν υπολογισμό χωρίς να ξεκινάμε την ανάλυση από την αρχή



- **Διατήρηση Κατάστασης (State):** Οι μεταβλητές και τα δεδομένα παραμένουν στη μνήμη, επιτρέποντας τον άμεσο πειραματισμό.
- **Άμεση Οπτικοποίηση:** Τα γραφήματα και οι πίνακες εμφανίζονται αμέσως κάτω από τις εντολές, διευκολύνοντας τον έλεγχο των δεδομένων.

Visual Studio Code

- Ένα κεντρικό λογισμικό που οργανώνει όλη τη ροή εργασίας (αρχεία, κώδικας, αποτελέσματα).
- **Variable Explorer:** Ένα περιβάλλον που επιτρέπει να βλέπουμε ανά πάσα στιγμή όλους τους πίνακες και τις τιμές μας

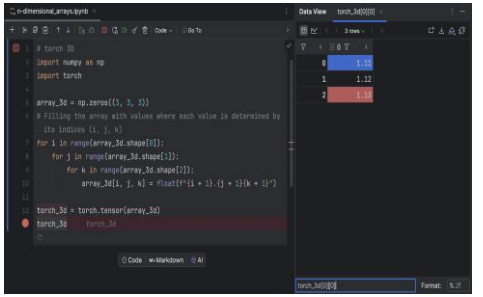


- **Έξυπνη Υποστήριξη:** Αυτόματη συμπλήρωση εντολών και εντοπισμός λαθών κατά τη συγγραφή.
- **Ενοποίηση:** Επιτρέπει τη χρήση των Jupyter Notebooks μέσα σε ένα πιο ισχυρό και οργανωμένο περιβάλλον διαχείρισης αρχείων.

PyCharm Community Edition

- Η δωρεάν, Open Source έκδοση του PyCharm, σχεδιασμένη αποκλειστικά για ανάπτυξη σε Python.

- **Intelligent Editor:** Κορυφαία αυτόματη συμπλήρωση κώδικα και πλοήγηση σε μεγάλα projects.



- Εξελιγμένο εργαλείο για τον εντοπισμό λαθών στη λογική των υπολογισμών βήμα-βήμα.

- Ειδοποιεί τον χρήστη αν ο κώδικας δεν ακολουθεί τα διεθνή πρότυπα (PEP 8), διασφαλίζοντας την αναγνωσιμότητα.

Anaconda – Ολοκληρωμένη Διανομή για Python & Data Science

- Περιλαμβάνει:
  - Τον Python interpreter
  - Το conda (διαχειριστής πακέτων & περιβαλλόντων)
  - Πλήθος έτοιμων βιβλιοθηκών (π.χ. numpy, pandas, matplotlib)
  - Εργαλεία όπως Jupyter Notebook, Spyder
- Πλεονεκτήματα:
  - Ευκολία στην εγκατάσταση όλων των εργαλείων μαζί
  - Ιδανικό για επιστημονικό υπολογισμό, στατιστική, machine learning
  - Κατάλληλο για εκπαιδευτικά εργαστήρια και αρχάριους

Εναλλακτικά & Συμπληρωματικά Εργαλεία

- **Miniconda**
  - Ελαφριά έκδοση του Anaconda – εγκαθιστάς μόνο τα πακέτα που χρειάζεσαι
- **JupyterLab**
  - Νέο, πιο ισχυρό interface για notebooks
  - Υποστηρίζει αρχεία, terminals, live code
- **Google Colab**
  - Online περιβάλλον Jupyter notebook
  - Υποστήριξη GPU/TPU χωρίς εγκατάσταση
  - Ιδανικό για συνεργασία και machine learning
- **Spyder IDE**
  - Περιβάλλον τύπου MATLAB, κατάλληλο για επιστημονικό υπολογισμό

Google Colab / Jupyter Notebook

- Εστίαση στην Ανάλυση (Insights): Στην ανάλυση δεδομένων, μας ενδιαφέρει το αποτέλεσμα και η ερμηνεία του. Τα Notebooks επιτρέπουν να βλέπουμε τα δεδομένα μας "ζωντανά" σε κάθε βήμα.

- Αποφυγή Τεχνικών Εμποδίων: Το Google Colab μας επιτρέπει να ξεκινήσουμε αμέσως, διασφαλίζοντας ότι όλοι οι συμμετέχοντες έχουν ακριβώς το ίδιο περιβάλλον και τις ίδιες βιβλιοθήκες, χωρίς προβλήματα συμβατότητας.

- Η δυνατότητα να έχουμε κείμενο, κώδικα και γραφήματα στην ίδια σελίδα καθιστά την ανάλυση εύκολα κατανοητή από τρίτους.

- Μεταφερσιμότητα: Οι γνώσεις από τη χρήση Notebooks μεταφέρονται αυτούσιες σε επαγγελματικά περιβάλλοντα όπως το VS Code, όταν οι ανάγκες γίνουν πιο σύνθετες.

Εγκατάσταση Python

Τοπική Εγκατάσταση (Windows, macOS, Linux)

Windows:

Λήψη του Installer από το python.org.Προσοχή: Πρέπει να επιλεγεί το κουτάκι "Add Python to PATH" κατά την έναρξη της εγκατάστασης (κρίσιμο για να αναγνωρίζει ο υπολογιστής τις εντολές).

macOS:

Το macOS διαθέτει μια έκδοση Python προεγκατεστημένη, αλλά προτείνεται η λήψη της τελευταίας σταθερής έκδοσης από το python.org ή μέσω του Homebrew (brew install python).

Ubuntu / Linux:

Συνήθως είναι ήδη εγκατεστημένη. Η ενημέρωση ή εγκατάσταση γίνεται μέσω του τερματικού με την εντολή: sudo apt update && sudo apt install python3.

Εγκατάσταση μέσω Docker (Containerization)

Το Container περιλαμβάνει ήδη την Python και όλες τις απαραίτητες βιβλιοθήκες ανάλυσης δεδομένων (Pandas, NumPy, κλπ.)

-Εγκατάσταση Docker Desktop: Λήψη και εγκατάσταση από το docker.com για Windows ή macOS. Στη συνέχεια κάνω είσοδο στο πρόγραμμα.

-Λήψη του Image (Pull): Ανοίγουμε το τερματικό και πληκτρολογούμε:
docker pull jupyter/datascience-notebook

-Εκτέλεση του Container (Run):
docker run -p 8888:8888 jupyter/datascience-notebook

-Πρόσβαση: Ανοίγουμε τον browser στη διεύθυνση localhost:8888 για να δουλέψουμε σε περιβάλλον Jupyter

Εγκατάσταση Εργαλείων & Βιβλιοθηκών (PIP)

- pip; Είναι ο επίσημος διαχειριστής πακέτων της Python.

Μας επιτρέπει να εγκαθιστούμε, να ενημερώνουμε και να αφαιρούμε βιβλιοθήκες κώδικα με απλές εντολές.

- Εγκατάσταση του Jupyter: Για να έχουμε το περιβάλλον των Notebooks τοπικά, δίνουμε την εντολή:

pip install jupyter

- Με μια γραμμή μπορούμε να εγκαταστήσουμε όλα τα εργαλεία:

pip install pandas numpy matplotlib

- Βασικές Εντολές:

pip list: Εμφανίζει όλες τις εγκατεστημένες βιβλιοθήκες.

pip install --upgrade [όνομα]: Ενημερώνει μια βιβλιοθήκη στην τελευταία έκδοση

Εκκίνηση του Jupyter Notebook

- Μόλις ολοκληρωθεί η εγκατάσταση μέσω pip, η διαδικασία για να ξεκινήσουμε την εργασία μας είναι η ακόλουθη:

- Η Εντολή Εκκίνησης: Ανοίγουμε το Τερματικό (ή το PowerShell στα Windows) και πληκτρολογούμε:
- jupyter notebook

- Τι συμβαίνει στο παρασκήνιο:
  1. Ο υπολογιστής ξεκινά έναν τοπικό διακομιστή (Local Server).
- 2. Ανοίγει αυτόματα μια νέα καρτέλα στον browser.

- Θα εμφανιστεί μια λίστα με τους φακέλους του υπολογιστή. Από εκεί επιλέγουμε:
- New -> Notebook: Για να δημιουργήσετε ένα νέο κενό αρχείο ανάλυσης Python 3.

- **Προσοχή**: Δεν κλείνουμε το παράθυρο του Τερματικού όσο εργαζόμαστε.

Εκκίνηση στο Google Colab

- Βρίσκεται στη διεύθυνση colab.research.google.com
- Συνδεόμαστε με τον Google λογαριασμό μας.
- Νέο Σημειωματάριο: Επιλέγουμε "New Notebook in Drive" (κάτω δεξιά στο αναδυόμενο παράθυρο).
- Σύνδεση με το Runtime: Πατάμε το κουμπί "Connect" στην πάνω δεξιά γωνία.
- Μόλις εμφανιστεί η ένδειξη RAM/Disk, ο υπολογιστής στο cloud είναι έτοιμος.

Εφαρμογές της Python και διαθέσιμες βιβλιοθήκες

- Διαθέτει χιλιάδες βιβλιοθήκες για διάφορους τομείς, όπως:
  - Ανάλυση δεδομένων (π.χ. pandas, numpy)
  - Μηχανική μάθηση και τεχνητή νοημοσύνη (π.χ. scikit-learn, tensorflow)
  - Ανάπτυξη ιστοσελίδων (π.χ. Django, Flask)
  - Κατασκευή παιχνιδιών (π.χ. pygame)
- Χρησιμοποιείται ευρέως:
  - Στην έρευνα και την εκπαίδευση
  - Στην επιχειρηματική ανάλυση
  - Στην αυτοματοποίηση διαδικασιών (scripts, bots)
  - Στην ανάπτυξη λογισμικού και εφαρμογών

Χρήσιμες Ιστοσελίδες

- Επίσημη ιστοσελίδα της Python
- Κεντρική βιβλιογραφία της Python
- Πληροφορίες για τη βασική βιβλιοθήκη της γλώσσας
- Λίστα με βιβλιοθήκες της Python

Χρήση της Python

Χρήση της Python σε άλλες Επιστήμες

- Η Python χρησιμοποιείται ευρέως και εκτός του πεδίου των επιστημών πληροφορικής:
- **Βιολογία**: Ανάλυση γονιδιώματος, bioinformatics, προσομοιώσεις
- **Χημεία**: Μοριακή μοντελοποίηση, φασματοσκοπία
- **Γεωεπιστήμες**: Ανάλυση χωρικών δεδομένων (GIS), μετεωρολογία
- **Οικονομικά / Κοινωνικές επιστήμες**: Στατιστική ανάλυση, οικονομετρία
- **Φυσική**: Προσομοίωση φαινομένων, αριθμητική επίλυση εξισώσεων

Γιατί Python για μηχανικούς & επιστήμονες;

- Εύκολη καμπύλη εκμάθησης για ερευνητές χωρίς background προγραμματισμού
- Πολλά διαθέσιμα παραδείγματα και notebooks
- Δυνατότητα αυτοματοποίησης πειραμάτων και αναλύσεων
- Υποστήριξη διεθνών επιστημονικών κοινοτήτων
- Ισχυρά εργαλεία για visualisation, στατιστικά και machine learning

Python & Data Science

- Η Python είναι η πιο δημοφιλής γλώσσα για Data Science λόγω:
  - Απλής σύνταξης
  - Πληθώρας βιβλιοθηκών
  - Μεγάλης κοινότητας
  - Ενσωμάτωσης με εργαλεία visualisation και machine learning

Βασικές Βιβλιοθήκες στην Python για Data Science

- NumPy: αριθμητικοί υπολογισμοί και πίνακες
- Pandas: ανάλυση και επεξεργασία δεδομένων (DataFrames)
- Matplotlib / Seaborn: οπτικοποίηση δεδομένων
- Scikit-learn: μηχανική μάθηση
- Statsmodels: στατιστική ανάλυση
- TensorFlow / PyTorch: deep learning

Ενδεικτικές Εφαρμογές της Python στη Data Science

- Καθαρισμός και μετασχηματισμός δεδομένων
- Δημιουργία γραφημάτων και dashboards
- Πρόβλεψη μέσω αλγορίθμων μηχανικής μάθησης
- Επεξεργασία φυσικής γλώσσας (NLP)
- Ανάλυση εικόνας και video

Βασικά στοιχεία της γλώσσας Python

Δομή και Κανόνες Σύνταξης

Case Sensitivity: Η Python ξεχωρίζει τα πεζά από τα κεφαλαία. Η μεταβλητή Data είναι διαφορετική από την data.

Σχόλια (#): Χρησιμοποιούμε το σύμβολο της δίεσης για να γράφουμε σημειώσεις που η Python αγνοεί.

Παράδειγμα: # Αυτό είναι ένα σχόλιο

Τέλος Εντολής: Σε αντίθεση με άλλες γλώσσες (C++, Java), στην Python δεν χρησιμοποιούμε ερωτηματικό (;) στο τέλος κάθε γραμμής. Η αλλαγή γραμμής σημαίνει νέα εντολή.

Η σημασία του Indentation (Εσοχή)

Indentation (Εσοχή): Είναι το κενό στην αρχή της γραμμής.

Ο Κανόνας: Η Python χρησιμοποιεί την εσοχή για να ορίσει μπλοκ κώδικα (αντί για τις αγκύλες { } που έχουν άλλες γλώσσες).

Standard: 1 Tab ή 4 κενά διαστήματα (Space).

Προσοχή: Λάθος εσοχή σημαίνει IndentationError και το πρόγραμμα σταματάει αμέσως.

Η σημασία του Indentation (Εσοχή)

- **Σωστό Παράδειγμα**
- **Ο Κανόνας:** Όλες οι εντολές που ανήκουν στο ίδιο «μπλοκ» κώδικα (π.χ. μέσα σε μια if, for ή συνάρτηση) πρέπει να ξεκινούν με την ίδια ακριβώς εσοχή.
- **Standard:** Χρησιμοποιούμε **4 κενά** (ή ένα Tab).

```
x = 5
if x > 2:
    print("Το x είναι μεγαλύτερο από το 2.")  # Εσοχή 4 κενών
    print("Αυτή η εντολή ανήκει στο ίδιο μπλοκ.") # Ίδια εσοχή
print("Αυτό εκτελείται πάντα (εκτός if).") # Επιστροφή στην αρχή
```

Η σημασία του Indentation (Εσοχή)

- **Λάθος Παράδειγμα (IndentationError)**
- **Σφάλμα:** Η Python θα σταματήσει αμέσως. Δεν θα εκτελέσει ούτε την πρώτη γραμμή, γιατί βλέπει ότι η δομή του κώδικα είναι κατεστραμμένη.
- **Σημείωση:** Αν ξεχάσετε την εσοχή ή βάλετε λάθος αριθμό κενών, η Python θα σταματήσει με ένα IndentationError.

```
x = 5
if x > 2:
print("Αυτό θα προκαλέσει λάθος!") # ΛΕΙΠΕΙ Η ΕΣΟΧΗ!
    print("Αυτό είναι σωστό, αλλά η προηγούμενη το χάλασε.")
```

Μεταβλητές (1/2)

Το όνομα μιας μεταβλητής:

- μπορεί να περιλαμβάνει λατινικούς χαρακτήρες ή αριθμούς (0…9) καθώς και το _
- αλλά **δεν** μπορεί να αρχίζει με αριθμητικό ψηφίο.

Παραδείγματα:

```
y = 7
firstname = “Νίκος“
pi        = 3.14
```

Μεταβλητές (2/2)

Τύποι αριθμητικών μεταβλητών:

- **<int>** ακέραιος (integer)
- **<float>** δεκαδικός (floating point)
- **<str>** συμβολοσειρά (string)

Με την εντολή type(x) εμφανίζεται ο τύπος μιας μεταβλητής

Εκχώρηση τιμής σε μεταβλητή (1/2)

Παραδείγματα:

```
b = 5
type(a)

z = z**3 + 10
x = “Κώστας ” - ”Νίκος”
```

Σύμβολα πράξεων:

+	Πρόσθεση
-	Αφαίρεση
*	Πολλαπλασιασμός
/	Διαίρεση
//	Ακέραια διαίρεση (πηλίκο)
%	modulo (υπόλοιπο διαίρεσης)
**	Ύψωση σε δύναμη

Εκχώρηση τιμής σε μεταβλητή (2/2)

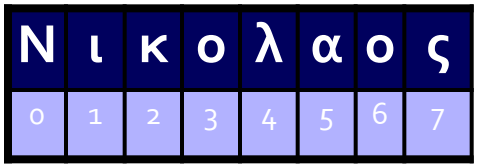
Συντομογραφίες πράξεων:

- **x += y** Πρόσθεση του y στην τιμή του x (x = x + y)
- **x -= y** Αφαίρεση του y από την τιμή του x (x = x – y)
- **x *= y** Πολλαπλασιασμός του y με την τιμή του x (x = x * y)
- **x /= y** Διαίρεση του y με την τιμή του x (x = x / y)
- **x //= y** Ακέραια διαίρεση του y με x (x = x // y)
- **x %= y.** Υπόλοιπο της διαίρεσης του y με x (x = x % y)
- **x **= y** ΄Υψωση του y στο x (x = x **y)

Δημιουργία συμβολοσειράς (string)

Μια συμβολοσειρά είναι μια ακολουθία χαρακτήρων που περικλείεται από μονά ή διπλά εισαγωγικά
Κάθε χαρακτήρας έχει τη θέση του στο αλφαριθμητικό οι οποίες ξεκινούν από τη θέση 0

Παραδείγματα:



name="Νικόλαος" (8 χαρακτήρες)
city= 'Πάτρα'

name[0] : N

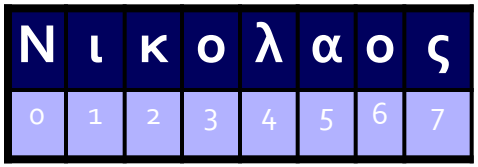
Οι χαρακτήρες αποθηκεύονται ως ακέραιοι μήκους από 1 έως 4 bytes
Η συνάρτηση ord(character) επιστρέφει τον κωδικό του χαρακτήρα
Η συνάρτηση chr(integer) επιστρέφει το χαρακτήρα που αντιστοιχεί ο κωδικός.

Τεμαχισμός συμβολοσειράς (slice)

Ένα τμήμα μιας συμβολοσειράς αποτελείται από το χαρακτήρα της αρχής μέχρι το χαρακτήρα τέλους, χωρίς να περιλαμβάνεται ο τελευταίος.

Παράδειγμα:

s="Νικόλαος" (8 χαρακτήρες)



s[2:6]
'κολα'

Πράξεις με συμβολοσειρές

Οι συμβολοσειρές αποτελούν αμετάβλητες ακολουθίες χαρακτήρων και **δεν επιτρέπεται η αλλαγή τους.**

Επιτρέπεται όμως η **πρόσθεση** συμβολοσειρών όπως και ο **πολλαπλασιασμός** συμβολοσειράς με ακέραιο.

τελεστής	**αποτέλεσμα**
<seq> + <seq>	συνένωση
<seq> * <int>	επανάληψη
<seq>[]	δείκτης
len(<seq>)	μήκος ακολουθίας
<seq>[:]	τεμαχισμός
for <var> in <seq>:	επανάληψη
<expr> in <seq>	συμμετοχή (Boolean)

Επιπλέον η συνάρτηση len(s) μας δίνει το πλήθος χαρακτήρων της συμβολοσειράς

Μέθοδοι συμβολοσειρών

Υπάρχουν πολλές μέθοδοι που μπορούν να εφαρμοστούν σε μια συμβολοσειρά string για αναζήτηση, μετατροπή, εξαγωγή λέξεων, έλεγχο αν το κείμενο αποτελείται από αριθμούς ή γράμματα κ.α.

Μερικές από αυτές είναι οι παρακάτω:

islower() - αληθές αν το str έχει μόνο πεζά
isdigit () - αληθές αν το str έχει μόνο αριθμούς
strip([chars]) – αφαιρεί τους χαρακτήρες από την αρχή και το τέλος του str
upper() - μετατρέπει τα κεφαλαία σε μικρά
lower() – μετατρέπει τα πεζά σε κεφαλαία
count() – μέτρηση του αριθμού εμφάνισης ενός χαρακτήρα σε ένα str
isalpha() – αληθές αν περιέχονται μόνο χαρακτήρες

Εντολές output

print()

print(ορίσματα, sep =‘δ’, end = ‘ε’)

δ : διαχωριστής

ε : τερματικός χαρακτήρας

Εντολές output

```
>>> x = 124.456

>>> print("η τιμή του χ είναι:", x)

η τιμή του χ είναι: 124.456

>>> print("η τιμή του χ είναι %f" % (x))

η τιμή του χ είναι 124.456000

>>> print("η τιμή του χ είναι %1.2f" % (x))

η τιμή του χ είναι 124.46
```

Εντολές input

input()

x=input(“Δώσε μια τιμή:”)

Προσοχή: Η input() επιστρέφει συμβολοσειρά

Τι είναι λάθος στο παρακάτω παράδειγμα:

x = input(“Δωσε 1ο αριθμό”)

y = input(“Δωσε 2ο αριθμό”)

s=x+y

Δομή δεδομένων Λεξικού (dictionary)

Ορισμός λεξικού ακολουθία από ζευγάρια τύπου:
dict={κλειδί: τιμή, …} μέσα σε { }

katalogos = {“Κώστας”:123456,”Σπύρος”:122344}

Τα κλειδιά πρέπει να είναι **μοναδικά** - επιτρέπονται αμετάβλητοι τύποι δεδομένων όπως αριθμοί, string, tuples, αλλά όχι list, dictionaries

Μέθοδοι λεξικών:
keys() – επιστρέφει τα κλειδιά
values() – επιστρέφει τις τιμές των στοιχείων
items() – επιστρέφει μια λίστα από τα αντικείμενα

Regular expressions (1)

Η βιβλιοθήκη re επιτρέπει τη χρήση της γλώσσας αναγνώρισης προτύπων σε κείμενο Regular Expressions (regex).

Μέσω της import re μπορεί να γίνει αναζήτηση προτύπων σε κείμενα, καθώς και εκτέλεση πράξεων όπως ανάκτηση, μετατροπή κ.α. βάσει αυτών των προτύπων.

Regular expressions (2)

Η βιβλιοθήκη re επιτρέπει τη χρήση Regular Expressions (regex), μια ισχυρή γλώσσα για τον εντοπισμό και τη διαχείριση προτύπων κειμένου. Μέσω της re μπορούμε να:

- Αναζητούμε πρότυπα σε κείμενα: Εντοπισμός συγκεκριμένων χαρακτήρων, λέξεων ή μοτίβων.

- Αντικαθιστούμε/Μετατρέπουμε κείμενο: Χρήση regex για αναγνώριση και αντικατάσταση κειμένων με άλλες τιμές ή μοτίβα.

- Διαχωρίζουμε κείμενο: Χρήση regex για διαχωρισμό κειμένου σε βάση προτύπων (π.χ. χώρους, κόμματα).

- Εξαγωγή πληροφοριών: Εξαγωγή συγκεκριμένων κομματιών πληροφορίας από κείμενα, όπως email ή αριθμούς τηλεφώνου.

•Regular Expressions

Character	Description	Example
[]	A set of characters	"[a-m]"
\	Signals a special sequence (can also be used to escape special characters)	"\d"
.	Any character (except newline character)	"he..o"
^	Starts with	"^hello"
$	Ends with	"world$"
*	Zero or more occurrences	"aix*"
+	One or more occurrences	"aix+"
{}	Exactly the specified number of occurrences	"al{2}"
|	Either or	"falls|stays"

•Regular Expressions

Character	Description
\A	Returns a match if the specified characters are at the beginning of the string
\b	Returns a match where the specified characters are at the beginning or at the end of a word (the "r" in the beginning is making sure that the string is being treated as a "raw string")
\B	Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word (the "r" in the beginning is making sure that the string is being treated as a "raw string")
\d	Returns a match where the string contains digits (numbers from 0-9)
\D	Returns a match where the string DOES NOT contain digits
\s	Returns a match where the string contains a white space character
\S	Returns a match where the string DOES NOT contain a white space character
\w	Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore _ character)
\W	Returns a match where the string DOES NOT contain any word characters
\Z	Returns a match if the specified characters are at the end of the string

Regular expressions – Κλάσεις χαρακτήρων

\w Οποιοσδήποτε αλφαριθμητικός χαρακτήρας [a-zA-Z0-9_]
\W Οτιδήποτε εκτός αλφαριθμητικών χαρακτήρων [^a-zA-Z0-9_]
\d Αριθμός, [0-9]
\D Οτιδήποτε εκτός αριθμού, [^0-9]
\s Κενό [\n\f\t\v\r]
\S Οτιδήποτε εκτός του κενού [^[\n\f\t\v\r]

\b Όριο μεταξύ αλφαριθμητικού και μη-αλφαριθμητικού (αρχή ή τέλος λέξης)

Regular expressions – Ειδικοί Χαρακτήρες

Ειδικοί χαρακτήρες: `. ^ $ * + ? { } [ ] \ | ( )`

Quantifiers)

`*` (Αστέρι): Το προηγούμενο σύμβολο υπάρχει **0 ή περισσότερες φορές**.
`+` (Συν): Το προηγούμενο σύμβολο υπάρχει **τουλάχιστον 1 φορά** (1 ή περισσότερες).
`?` (Ερωτηματικό): Το προηγούμενο σύμβολο είναι **προαιρετικό** (0 ή 1 φορά).
`{ }` (Αγκύλες): Ορίζουν **ακριβή αριθμό** επαναλήψεων (π.χ. `{3}` για ακριβώς τρεις φορές).

Θέση και Όρια
`.` (Τελεία): Ο **"Μπαλαντέρ"**. Αντιπροσωπεύει **οποιονδήποτε** χαρακτήρα (γράμμα, αριθμό, σύμβολο).
`^` (Caret): Δηλώνει ότι το μοτίβο πρέπει να βρίσκεται στην **αρχή** της γραμμής.
`$` (Δολάριο): Δηλώνει ότι το μοτίβο πρέπει να βρίσκεται στο **τέλος** της γραμμής.

Regular expressions – Ειδικοί Χαρακτήρες

Ειδικοί χαρακτήρες: . ^ $ * + ? { } [ ] \ | ( )

Ομαδοποίηση και Επιλογή
[ ] (Square Brackets): Μια **λίστα επιλογών**. Ψάχνει οποιονδήποτε χαρακτήρα περιέχεται μέσα (π.χ. [αειου] για φωνήεντα).
| (Pipe): Το **"Ή" (OR)**. Επιλέγει ανάμεσα σε δύο ολόκληρα μοτίβα (π.χ. μήλο|πορτοκάλι).
( ) (Parentheses): Ομαδοποίηση. Αντιμετωπίζει μια ομάδα χαρακτήρων ως μία ενότητα.

Escape Character
\ (Backslash): Ακυρώνει τη δράση των ειδικών χαρακτήρων.
Παράδειγμα: Αν θέλεις να ψάξεις μια πραγματική τελεία, γράφεις **\.**

Regular expressions – Ειδικοί Χαρακτήρες

Ειδικοί χαρακτήρες: . ^ $ * + ? { } [ ] \ | ( )

Σύμβολο	Πρότυπο	Τι θα βρει;	Εξήγηση
?	χρώμ?α	**χρώα** & **χρώμα**	Το 'μ' υπάρχει **0 ή 1** φορά (προαιρετικό).
+	χρώμ+α	**χρώμα** & **χρώμμα**	Το 'μ' υπάρχει **1 ή περισσότερες** φορές.
*****	χρώμ*α	**χρώα**, **χρώμα**, **χρώμμα**	Το 'μ' υπάρχει **0, 1 ή πολλές** φορές.
.	χρ.μα	**χρώμα**, **χράμα**, **χρήμα**	Η τελεία αντικαθιστά **οποιοδήποτε** ένα γράμμα.

Regular expressions - Συναρτήσεις

re.**compile**(pattern): αντικείμενο pattern* (μετάφραση προτύπου)

re.match(), re.search(): Αναζητούν πρότυπα στην αρχή του κειμένου ή οποιοδήποτε μέρος του.

re.findall(), re.finditer(): Επιστρέφουν όλα τα ευρήματα ως λίστα ή iterator.

re.sub(): Αντικαθιστά τα ευρήματα με νέο κείμενο.

re.split(): Χρησιμοποιεί πρότυπα για να διαχωρίσει το κείμενο.

Regular expressions - Συναρτήσεις

re.split(): Χρησιμοποιεί πρότυπα για να διαχωρίσει το κείμενο.

```
import re
data =
"μήλα;πορτοκάλια,μπανάνες|κεράσια"
# Χώρισε όπου βρεις ; ή , ή |
fruits = re.split(r"[;,|]", data)

print(f"Λίστα φρούτων: {fruits}")
```

```
Λίστα φρούτων: ['μήλα', 'πορτοκάλια',
'μπανάνες', 'κεράσια']
```

Regular expressions - Συναρτήσεις

re.match(), re.search(): Αναζητούν πρότυπα στην αρχή του κειμένου ή οποιοδήποτε μέρος του.

```
import re
text = "Python is amazing"

# match: ψάχνει στην αρχή
m = re.match(r"amazing", text)
# search: ψάχνει σε όλο το κείμενο
s = re.search(r"amazing", text)

print(f"Match result: {m}")
print(f"Search result: {s.group() if s else
'None'}")
```

```
Match result: None
Search result: amazing
```

Regular expressions - Συναρτήσεις

re.findall(), re.finditer(): Επιστρέφουν όλα τα ευρήματα ως λίστα ή iterator.

```
import re
log = "User1: 125, User2: 450, User3: 880"
# \d+ σημαίνει "ένας ή περισσότεροι αριθμοί"
numbers = re.findall(r"\d+", log)

print(f"Λίστα τιμών: {numbers}")
```

Λίστα τιμών: ['125', '450', '880']

Regular expressions - Συναρτήσεις

re.sub(): Αντικαθιστά τα ευρήματα με νέο κείμενο.

```
import re
phone = "210-123-4567 # Αυτό είναι το
τηλέφωνο"

# Αφαιρούμε οτιδήποτε δεν είναι ψηφίο (\D)
clean_phone = re.sub(r"\D", "", phone)

print(f"Καθαρό τηλέφωνο: {clean_phone}")
```

```
Καθαρό τηλέφωνο: 2101234567
```

Regular expressions - Συναρτήσεις

re.**compile**(pattern): αντικείμενο pattern* (μετάφραση προτύπου)

```
import re

# 1. Προετοιμασία (Compile): Φτιάχνουμε το
"καλούπι" για ημερομηνία (π.χ. 12/05/2024)
date_pattern = re.compile(r"\d{2}/\d{2}/\d{4}")

# 2. Χρήση: Εφαρμόζουμε το ίδιο καλούπι σε
διαφορετικά strings
text1 = "Η συνάντηση είναι στις 15/04/2026."
text2 = "Η προθεσμία λήγει στις 20/05/2026."

match1 = date_pattern.search(text1)
match2 = date_pattern.search(text2)

if match1: print(f"Βρέθηκε 1: {match1.group()}")
if match2: print(f"Βρέθηκε 2: {match2.group()}")
```

```
Βρέθηκε 1:
15/04/2026
Βρέθηκε 2:
20/05/2026
```

Regular expressions - Συναρτήσεις

re.**compile**(pattern): αντικείμενο pattern* (μετάφραση προτύπου)

r (Raw String): Λέει στην Python να διαβάσει το κείμενο ακριβώς όπως είναι, χωρίς να παρερμηνεύσει τις ανάποδες κάθετους (\).

\d (Digit): Αντιπροσωπεύει οποιοδήποτε ψηφίο από το 0 έως το 9. Είναι το ίδιο με το [0-9].

{n} (Quantifier): Ορίζει το πλήθος των επαναλήψεων του προηγούμενου συμβόλου.

{2}: Ακριβώς 2 ψηφία.

{4}: Ακριβώς 4 ψηφία.

/ (Literal): Αναζητά τον χαρακτήρα της κάθετου ακριβώς όπως είναι.

re.compile(r"\d{2}/\d{2}/\d{4}")

15/04/2026 NAI
5.4.2025 OXI
15-04-2026 OXI

•Regular Expressions - Παραδείγματα

Στο παρακάτω παράδειγμα γίνεται ο έλεγχος
αν το string **txt** αρχίζει με “**The**” και τελειώνει με “**Italy**”

```
import re

txt = "The weather in Italy"
x = re.search("^The.*Italy$", txt)
```

•Regular Expressions - Παραδείγματα

Στο παρακάτω παράδειγμα γίνεται split στο string txt σε κάθε white space χαρακτήρα

```
import re

txt = "The weather in Italy"
x = re.split("\s", txt)
```

•Regular Expressions - Παραδείγματα

Στο παρακάτω παράδειγμα γίνεται split στο string txt μόνο στο πρώτο white space χαρακτήρα

```
import re

txt = "The weather in Italy"
x = re.split("\s", txt, 1)
```

•Regular Expressions - Παραδείγματα

Στο παρακάτω παράδειγμα γίνεται αντικατάσταση στο string txt σε κάθε χαρακτήρα white space με το νούμερο 5

```
import re

txt = "The weather in Italy"
x = re.sub("\s", "5" txt)
```

•Regular Expressions - Παραδείγματα

Στο παρακάτω παράδειγμα γίνεται αντικατάσταση στο string txt μόνο στις δυο πρώτες εμφανίσεις του χαρακτήρα white space με το νούμερο 5

```
import re

txt = "The weather in Italy"
x = re.sub("\s", "5" txt, 2)
```

•Regular Expressions - Παραδείγματα

Στο παρακάτω παράδειγμα γίνεται αναζήτηση στο string txt για το τμήμα που περιέχει ένα χαρακτήρα I με κεφαλαία στην αρχή μιας λέξης. Χρησιμοποιούμε το property group στην εκτύπωση για να μας επιστρέψει ολόκληρο το τμήμα που έγινε match

```
import re

txt = "The weather in Italy"
x = re.search(r"\bI\w+", txt)
Print(x.group())
```

Web scaping - api

- Το web scraping είναι μια τεχνική για τη συλλογή δεδομένων ή πληροφοριών σε ιστοσελίδες.
- Είναι μια μέθοδος εξαγωγής δεδομένων από έναν ιστότοπο που δεν διαθέτει API ή όταν θέλουμε να εξάγουμε ΠΟΛΛΑ δεδομένα, κάτι που δεν μπορούμε να κάνουμε μέσω ενός API λόγω περιορισμού ταχύτητας.
- Μπορούμε να κάνουμε εξαγωγή οποιωνδήποτε δεδομένων θέλουμε κατά την περιήγηση στο διαδίκτυο

ΠΑΡΑΔΕΙΓΜΑΤΑ ΧΡΗΣΗΣ WEB SCRAPING

- Εξαγωγή πληροφοριών προϊόντος
- Εξαγωγή αγγελιών εργασίας και πρακτικής άσκησης
- Εξαγωγή προσφορών και εκπτώσεων από ιστότοπους με προσφορές της ημέρας
- Εξαγωγή δεδομένων για τη δημιουργία μηχανής αναζήτησης
- Συλλογή δεδομένων καιρού

WEB SCRAPING ή χρήση API

- Η συλλογή δεδομένων από ιστοσελίδες δεν υπόκειται σε περιορισμούς ταχύτητας
- Ανώνυμη πρόσβαση στον ιστότοπο και συλλογή δεδομένων
- Ορισμένοι ιστότοποι δεν διαθέτουν API
- Ορισμένα δεδομένα δεν είναι προσβάσιμα μέσω API

Ροη εργασιών WEB SCRAPING

- Λήψη στοιχείων του ιστότοπου - χρησιμοποιώντας τη βιβλιοθήκη HTTP
- Ανάλυση του εγγράφου html - χρησιμοποιώντας οποιαδήποτε βιβλιοθήκη ανάλυσης
- Αποθήκευση των αποτελεσμάτων - είτε σε βάση δεδομένων, csv, αρχείο κειμένου κ.α.

WEB SCRAPING LIBRARIES

Βιβλιοθήκη	**Ταχύτητα**	**Ευκολία Χρήσης**
BeautifulSoup	Πολύ αργή (245x πιο αργή από re)	Πολύ εύκολη
lxml	Πολύ γρήγορη	Μέτρια (χρήση XPath)
re (Regex)	**Η ταχύτερη όλων** (7.186 ms)	Δύσκολη/Περίπλοκη
Selenium	Πάρα πολύ αργή (Η πιο αργή, λόγω εξομοίωσης browser)	Μέτρια (Απαιτεί drivers και διαχείριση δυναμικών στοιχείων)

Ανάκτηση δεδομένων από το διαδίκτυο

Βιβλιοθήκη urllib.request

```
req_object = urllib.request.Request(διαδ. πόρος)
mypage = urllib.request.urlopen(req_object)
html = mypage.read().decode()
```

HTTP LIBRARIES

- Requests

r = requests.get('https://www.google.com').html

requests.get(url) στέλνει ένα **GET** αίτημα στη διεύθυνση URL.

- urllib

html = urllib.request.urlopen('http://python.org/').read()()

Η urlopen() ανοίγει τον σύνδεσμο και read() διαβάζει τα bytes της απάντησης.

- httplib/httplib2

h = httplib2.Http(".cache")

(resp_headers, content) = h.request("http://pydelhi.org/", "GET")

Η httplib2 είναι εξωτερική βιβλιοθήκη που υποστηρίζει caching, authentication, redirects, κ.λπ.

- Http(".cache"): Δημιουργεί HTTP client που αποθηκεύει απαντήσεις σε τοπικό cache directory.
- h.request(...) επιστρέφει:

--resp_headers: Λεξικό με HTTP headers απόκρισης.

--content: Το σώμα της απάντησης (συνήθως HTML ή JSON).

PARSING LIBRARIES

- **BeautifulSoup (bs4)**

```
# Import the BeautifulSoup class from the bs4 module
from bs4 import BeautifulSoup
# Parse the HTML document using BeautifulSoup (defaults to 'html.parser')
tree = BeautifulSoup(html_doc)
# Access and return the <title> element of the HTML document
tree.title
```

- **lxml**

```
# Import lxml's HTML parsing module
import lxml.html

# Parse the HTML document into lxml tree structure
tree = lxml.html.fromstring(html_doc)
# Use XPath to extract the text inside the <title> tag
title = tree.xpath('/title/text()')
```

- **re**

```
# Import the regular expressions module
import re
# Use a regular expression to search for content inside the <title> tag
title = re.findall('<title>(.*?)</title>', html_doc)
```

BEAUTIFULSOUP

- μπορούμε να το μάθουμε γρήγορα
- πολύ εύκολο στη χρήση
- αποκλειστικά σε Python
- πολύ αργό

SELENIUM

- Διαχείριση δυναμικού περιεχομένου (JavaScript)
- Πλήρης αυτοματοποίηση του προγράμματος περιήγησης (Browser)
- Δυνατότητα αλληλεπίδρασης (κλικ, συμπλήρωση φορμών, scrolling)
- Πάρα πολύ αργό (λόγω πλήρους φόρτωσης της σελίδας)

LXML

- πολύ γρήγορο
- όχι αποκλειστικά σε Python
- το lxml λειτουργεί με όλες τις εκδόσεις python από 2.x έως 3.x

RE

- Αναγκαστική γνώση των συμβόλων του
- μπορεί να γίνει περίπλοκο
- αποκλειστικά σε Python
- μέρος της τυπικής βιβλιοθήκης
- πολύ γρήγορο
- για κάθε έκδοση Python

Είναι νόμιμο το web scraping;

- Ασφαλώς, η ενέργεια του web scraping δεν είναι παράνομη.
- Ωστόσο, πρέπει να ακολουθούνται ορισμένοι κανόνες.
- Το web scraping μπορεί να γίνει παράνομο όταν εξάγονται δεδομένα που δεν είναι δημόσια διαθέσιμα.

Αποτέλεσμα

- Desktop$ python test.py
- bs_test took 1851.457 ms
- lxml_test took 232.942 ms
- regex_test took 7.186 ms
- selenium_test took ~5000ms - 8000ms (5-8 δευτερόλεπτα)

- lxml took 32x more time than re, BeautifulSoup took 245x!

more time than re

Παράδειγμα web scraping & Regex

```python
import re
import requests
import time

# Ορισμός του URL
url = "https://www.python.org"

# Λήψη του περιεχομένου της σελίδας
response = requests.get(url)

# Μέτρηση χρόνου εκτέλεσης
start = time.time()

# Χρήση Regular Expression για την εύρεση του περιεχομένου του <title>
title = re.findall(r'<title>(.*?)</title>', response.text)

end = time.time()

# Έλεγχος αν βρέθηκε τίτλος και εκτύπωση αποτελέσματος
if title:
    print(f"Regex found: {title[0]} in {(end-start)*1000 :.3f} ms")
else:
    print("No title found.")
```

```
Regex found: Welcome to Python.org in 0.523 ms
```

Παράδειγμα web scraping & BeautifulSoup