MEDLAB
MEDICAL TECHNOLOGY & INTELLIGENT INFORMATION SYSTEMS

Εισαγωγή στην Python για Τεχνητή Νοημοσύνη και Επιστήμη Υλικών



- Τμήμα Μηχανικών Επιστήμης Υλικών
- Website: https://medlab.cc.uoi.gr/

- Ιωάννινα 2026 - 2027

Δεδομένα, χαρακτηριστικά και μοντέλα

Ολοκλήρωση Διάλεξης 1

Ενότητα 5 · πώς «βλέπει» ένα μοντέλο τα δεδομένα υλικών

	Χαρακτηριστικά (features) → X, είσοδος				Ετικέτα → y
Δείγμα	**C (wt%)**	**Cr (wt%)**	**T (°C)**	**t (min)**	**HV**
M1	0.20	1.0	800	30	**180**
M2	0.40	1.5	850	45	**235**
M3	0.60	1.0	900	60	**310**
M4	0.35	1.8	850	30	**260**
M5	0.50	1.2	875	45	**290**

- **Γραμμή:** ένα δείγμα, μία μέτρηση ή πειραματική περίπτωση
- **Στήλη:** ένα χαρακτηριστικό: σύσταση, κατεργασία, μικροδομή
- **Επιβλεπόμενη μάθηση:** μαθαίνουμε τη σχέση X → y από γνωστά ζεύγη

Εκπαίδευση: επανάληψη μέχρι η απώλεια να γίνει μικρή

Αρχικές τιμές → Πρόβλεψη ŷ → Απώλεια ŷ ↔ y → Ενημέρωση

Αλγόριθμος (algorithm)

η «συνταγή» της μάθησης: πώς προσαρμόζονται οι παράμετροι. Π.χ. γραμμική παλινδρόμηση, δέντρα απόφασης, k-means.

Μοντέλο (model)

το αποτέλεσμα της εκπαίδευσης, έτοιμο να προβλέψει:

$$\hat{HV} = \beta_0 + \beta_1 \cdot C + \beta_2 \cdot Cr + \beta_3 \cdot T + \beta_4 \cdot t$$

ΣΥΝΔΕΣΗ ΜΕ ΤΟ ΣΗΜΕΡΙΝΟ ΜΑΘΗΜΑ

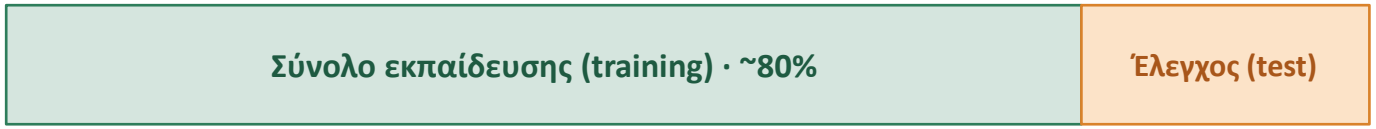
Σήμερα θα γράψουμε αυτόν τον πίνακα σε Python: κάθε γραμμή ένα **dictionary**, ο πίνακας μια **λίστα από dictionaries** (διαφάνεια 27). Στο Μάθημα 5 θα τον εκπαιδεύσουμε.

Εκπαίδευση, αξιολόγηση και γενίκευση

Ολοκλήρωση Διάλεξης 1

Ενότητα 6 · ένα μοντέλο αξίζει μόνο αν προβλέπει σωστά δεδομένα που δεν έχει δει

Διαχωρισμός δεδομένων



προσδιορίζει τις παραμέτρους του μοντέλου

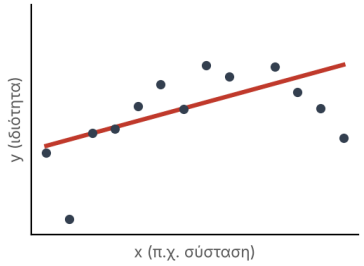
δεν τα «βλέπει» στην εκπαίδευση

Διασταυρωμένη επικύρωση (k-fold)



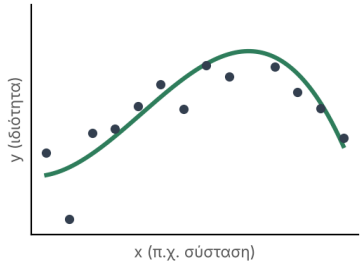
κάθε κομμάτι γίνεται μία φορά validation· μέρος της εκπαίδευσης

Υποπροσαρμογή (underfitting)



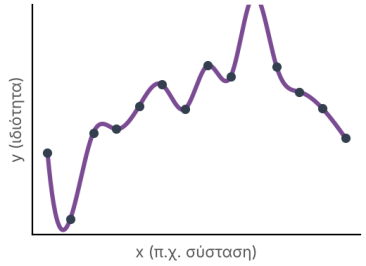
πολύ απλό μοντέλο · χαμηλή απόδοση σε εκπαίδευση και έλεγχο

Καλή προσαρμογή



αποτυπώνει τη γενική τάση · καλή γενίκευση

Υπερπροσαρμογή (overfitting)



άριστο στην εκπαίδευση, κακό στον έλεγχο · «μαθαίνει» τον θόρυβο

Σχηματικά, με συνθετικά σημεία.

ΚΥΡΙΟ ΜΗΝΥΜΑ

Στόχος δεν είναι να απομνημονεύσει το μοντέλο τα δεδομένα εκπαίδευσης, αλλά να προβλέπει αξιόπιστα τη σκληρότητα νέων κραμάτων: [C, Cr, T, t] → HV.

Ροή εργασίας και εφαρμογές στα υλικά

Ολοκλήρωση Διάλεξης 1

Ενότητες 7–8 · από ένα ερώτημα υλικών σε ένα σαφές πρόβλημα μηχανικής μάθησης

Ερώτημα
τι θέλουμε να μάθουμε;

→

Στόχος y
ποια είναι η έξοδος;

→

Χαρακτηριστικά x
ποιες είναι οι είσοδοι;

→

Εργασία ΜΜ
regression, classification, clustering

→

Δεδομένα
δείγματα με κοινή δομή

→

Αλγόριθμος
πώς θα μάθει το μοντέλο;

Παραδείγματα

Σκληρότητα ενός νέου κράματος	→ **Regression**	πρόβλεψη αριθμού
Κατηγορία μικροδομής από εικόνες	→ **Classification**	πρόβλεψη κατηγορίας
Ομάδες σε άγνωστες μικροδομές	→ **Clustering**	χωρίς ετικέτες

Προετοιμασία δεδομένων

Συχνά το μεγαλύτερο μέρος της δουλειάς:
ακατέργαστα → έλεγχος → καθαρισμός → μετασχηματισμός → επισήμανση → έτοιμο dataset

Συχνά προβλήματα: λίγα δείγματα, ελλιπείς ετικέτες, λανθασμένες ή θορυβώδεις τιμές, μη αντιπροσωπευτικό σύνολο.

ΕΦΑΡΜΟΓΗ ΣΤΑ ΥΛΙΚΑ

Η ΜΜ συνδέει σύσταση, επεξεργασία και μικροδομή με τις ιδιότητες και την απόδοση: πρόβλεψη ιδιοτήτων, ταξινόμηση μικροδομών, ανίχνευση ανωμαλιών, σχεδιασμός νέων υλικών.

ΚΥΡΙΟ ΜΗΝΥΜΑ

Η ποιότητα ενός μοντέλου δεν ξεπερνά την ποιότητα των δεδομένων του. Και κάθε βήμα αυτής της ροής γίνεται με κώδικα: γι' αυτό σήμερα μαθαίνουμε Python.

Στόχοι Μαθήματος

1 · Περιβάλλον

Μαθαίνουμε να σκεφτόμαστε και να προγραμματίζουμε με Python

Περιβάλλον
Colab, Jupyter

→

Μεταβλητές
πράξεις, μονάδες

→

Λίστες
και dictionaries

→

Ροή ελέγχου
if, for, while

→

Συναρτήσεις
def, return

→

Πρώτο πρόγραμμα
δοκίμια εφελκυσμού

Να δουλεύετε σε Jupyter / Google Colab

- να εκτελείτε κελιά με τη σωστή σειρά
- να διαβάζετε μηνύματα σφάλματος
- να ξεκινάτε από Restart & Run All

Να διαβάζετε και να γράφετε βασικό κώδικα

- μεταβλητές, τύποι, πράξεις με μονάδες
- λίστες, tuples, dictionaries
- συνθήκες, βρόχοι, συναρτήσεις

Να λύνετε ένα μικρό πρόβλημα υλικών

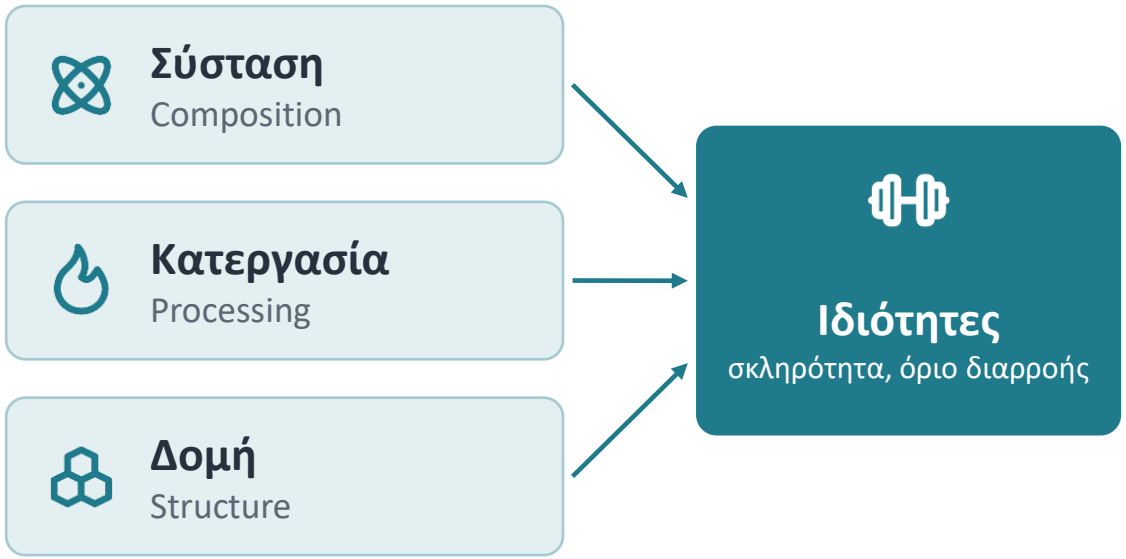
- οργάνωση μετρήσεων σε δεδομένα
- υπολογισμοί και έλεγχοι προδιαγραφής
- σύνοψη αποτελεσμάτων με f-strings

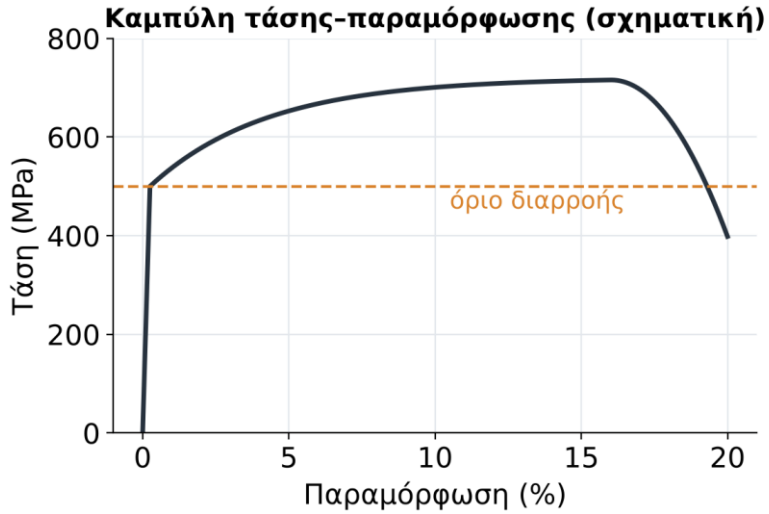
ΚΥΡΙΟ ΜΗΝΥΜΑ

Δεν χρειάζεται να απομνημονεύσετε εντολές. Χρειάζεται να καταλαβαίνετε τι κάνει κάθε γραμμή, ώστε να μπορείτε να τη διαβάσετε, να την αλλάξετε και να τη διορθώσετε.

1 · Περιβάλλον

Materials Informatics: σύνδεση σύστασης και κατεργασίας με τις ιδιότητες μέσω δεδομένων





Τέτοιες μετρήσεις θα επεξεργαστούμε σήμερα με κώδικα. Η καμπύλη είναι σχηματική.

Εύκολη σύνταξη
κώδικας που διαβάζεται σχεδόν σαν αγγλικά

Διερμηνευόμενη
τρέχει γραμμή-γραμμή, χωρίς μεταγλώττιση

Πλούσιες βιβλιοθήκες
NumPy, Pandas, scikit-learn (Μάθημα 5)

Μεγάλη κοινότητα
κυρίαρχη γλώσσα στο AI (Liu, Κεφ. 2)



Περιβάλλον Python

Το περιβάλλον μας και το πρώτο κελί

1 · Περιβάλλον

Ο κώδικας είναι ο ίδιος τοπικά και στο cloud· σήμερα δουλεύουμε στο Colab

Τοπικά (JupyterLab)

+ χωρίς internet, πλήρης έλεγχος
– απαιτεί εγκατάσταση (Παράρτημα)

Cloud (Google Colab) ← σήμερα

+ καμία εγκατάσταση, ίδιο περιβάλλον για όλους
– internet· η μνήμη σβήνει όταν λήξει η συνεδρία

- **Άνοιγμα:** colab.research.google.com → File → New notebook
- **Εκτέλεση:** κουμπί ▶ ή Shift + Enter
- **Νέο κελί:** + Code (κώδικας), + Text (σημειώσεις)

Python

```
print("Hello, World!")
print(2 + 3)
```

Αποτέλεσμα (output)

```
Hello, World!
5
```

ΚΥΡΙΟ ΜΗΝΥΜΑ

Αν βλέπετε αυτό το αποτέλεσμα, το περιβάλλον σας είναι έτοιμο, είτε στο Colab είτε τοπικά. Το notebook είναι ένα αρχείο .ipynb: στο Colab αποθηκεύεται στο Google Drive, τοπικά στον φάκελο του υπολογιστή σας.

Ανατομία ενός Notebook: κελιά και kernel

1 · Περιβάλλον

Το notebook συνδυάζει κείμενο, κώδικα και αποτελέσματα σε ένα έγγραφο

lab_02_python.ipynb — Kernel: Python 3

Μετρήσεις σκληρότητας
Κελί κειμένου (Markdown)

[1]:
```
hardness_HV = 240
```

[2]:
```
print(hardness_HV * 2)
```
```
480
```

[ ]:
```
| νέο κελί, δεν έχει εκτελεστεί
```

Κελί (cell)
Μικρό κομμάτι κώδικα ή κειμένου. Εκτελείται με Shift + Enter.

Αριθμός εκτέλεσης [n]
Δείχνει με ποια σειρά εκτελέστηκαν τα κελιά, όχι τη θέση τους.

Kernel
Η «μηχανή» που εκτελεί τον κώδικα και θυμάται όλες τις μεταβλητές.

Restart & Run All
Σβήνει τη μνήμη και τρέχει όλα τα κελιά από πάνω προς τα κάτω.

ΣΥΧΝΟ ΛΑΘΟΣ
Εκτέλεση του [2] χωρίς να έχει τρέξει το [1] → NameError: η μεταβλητή δεν υπάρχει ακόμη στη μνήμη του kernel.

Πώς διαβάζουμε ένα μήνυμα σφάλματος

1 · Περιβάλλον

Τα σφάλματα είναι φυσιολογικό μέρος του προγραμματισμού· το μήνυμα λέει τι πήγε στραβά

```
Traceback (most recent call last):
  Cell In[3], line 2
    print(hardnes_HV)
NameError: name 'hardnes_HV' is not defined
```

1. **Κοιτάμε την τελευταία γραμμή** Τύπος σφάλματος + εξήγηση: εδώ NameError, δηλαδή άγνωστο όνομα.
2. **Βρίσκουμε τη γραμμή** «line 2»: πού ακριβώς σταμάτησε η Python.
3. **Συγκρίνουμε προσεκτικά** hardnes_HV ≠ hardness_HV: λείπει ένα «s».

Τα πιο συνηθισμένα σφάλματα

SyntaxError
λάθος γραφή: ξεχασμένη «:» ή παρένθεση

NameError
άγνωστο όνομα: τυπογραφικό ή κελί που δεν έτρεξε

TypeError
πράξη σε ασύμβατους τύπους, π.χ. "5" + 1

IndentationError
λάθος εσοχή

IndexError
θέση εκτός λίστας

KeyError
κλειδί που δεν υπάρχει σε dictionary

Από το πρόβλημα στον κώδικα: η υπολογιστική σκέψη

1 · Περιβάλλον

Πριν γράψουμε κώδικα, μετατρέπουμε το πρόβλημα σε ακολουθία βημάτων

Πρόβλημα
τι θέλουμε να βρούμε;

→ **Δεδομένα**
τι γνωρίζουμε; με ποιες μονάδες;

→ **Σχέση**
ποιος τύπος τα συνδέει;

→ **Βήματα**
με ποια σειρά υπολογίζουμε;

→ **Κώδικας**
κάθε βήμα = μία γραμμή

→ **Έλεγχος**
είναι λογικό το αποτέλεσμα;

Παράδειγμα: πυκνότητα ενός δοκιμίου

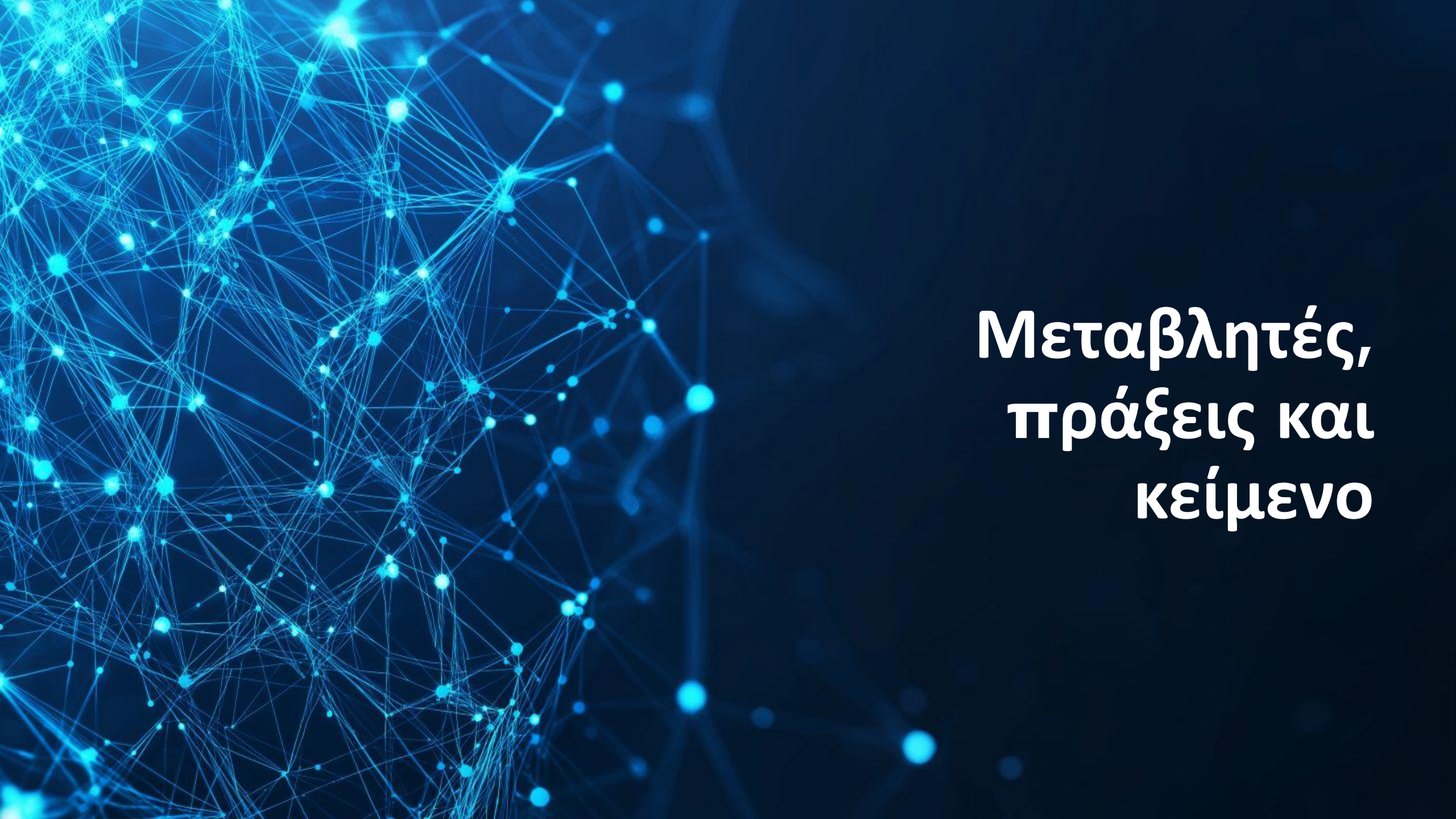
- **Πρόβλημα:** ποια είναι η πυκνότητα του δοκιμίου;
- **Δεδομένα:** m = 15.7 g, V = 2.0 cm³
- **Σχέση:** ρ = m / V
- **Βήματα:** αποθηκεύω m, V → διαιρώ → τυπώνω με μονάδα
- **Έλεγχος:** 7.85 g/cm³ ≈ χάλυβας ✓

Python

```
mass_g = 15.7                       # δεδομένο
volume_cm3 = 2.0                    # δεδομένο
density = mass_g / volume_cm3       # σχέση ρ = m / V
print(density, "g/cm³")             # αποτέλεσμα
```

Αποτέλεσμα (output)

```
7.85 g/cm³
```



Μεταβλητές, πράξεις και κείμενο

Η πρώτη εντολή: print() και σχόλια

2 · Μεταβλητές & πράξεις

Πώς ζητάμε από την Python να εμφανίσει πληροφορία στην οθόνη

Python

```
# Το πρώτο μας πρόγραμμα
print("Καλώς ήρθατε στο εργαστήριο!")
print("Υλικό:", "χάλυβας S355")
print(7.85)    # πυκνότητα σε g/cm³
```

Αποτέλεσμα (output)

```
Καλώς ήρθατε στο εργαστήριο!
Υλικό: χάλυβας S355
7.85
```

ΕΦΑΡΜΟΓΗ ΣΤΑ ΥΛΙΚΑ

Τα σχόλια (#) είναι η «εργαστηριακή σημείωση» του κώδικα: γράφουμε μονάδες και υποθέσεις.

```
print("Hello")
```

`print`	όνομα συνάρτησης (function)
`( )`	εδώ μπαίνουν τα ορίσματα (arguments)
`"…"`	κείμενο (string) σε εισαγωγικά
`#`	σχόλιο: αγνοείται από την Python

ΣΥΧΝΟ ΛΑΘΟΣ

Print("...") με κεφαλαίο P → NameError. Η Python ξεχωρίζει κεφαλαία από πεζά (case-sensitive).

Μεταβλητές, ανάθεση τιμής και ονόματα με μονάδες

2 · Μεταβλητές & πράξεις

Μια μεταβλητή είναι ένα όνομα που δείχνει σε μια τιμή αποθηκευμένη στη μνήμη

temperature_C	density_g_cm3	yield_strength_MPa
850	7.85	510

Python

```
temperature_C = 850
density_g_cm3 = 7.85
yield_strength_MPa = 510
yield_strength_MPa = 525    # νέα τιμή
print(yield_strength_MPa)
```

Αποτέλεσμα (output)

```
525
```

Κανόνες και καλή πρακτική

- Ξεκινούν με γράμμα, χωρίς κενά: όχι 2nd_sample
- Κεφαλαία ≠ πεζά: Hardness ≠ hardness
- Όχι δεσμευμένες λέξεις: lambda, for, if
- Πεζά με _ και μονάδα στο όνομα: _MPa, _C

ΣΥΧΝΟ ΛΑΘΟΣ

Το = σημαίνει «αποθήκευσε», όχι «ισούται». Διαβάζεται από δεξιά προς αριστερά. Η σύγκριση γράφεται == (Ενότητα 4).

Τύποι δεδομένων: int, float και η συνάρτηση type()

2 · Μεταβλητές & πράξεις

Κάθε τιμή έχει έναν τύπο, που καθορίζει τι πράξεις μπορούμε να κάνουμε

Τύπος	Τι είναι	Παραδείγματα	Στα υλικά
`int`	ακέραιος	`12, 850, -3`	πλήθος δοκιμίων, κύκλοι κόπωσης
`float`	δεκαδικός	`7.85, 0.20, 850.0`	πυκνότητα, τάση, % κ.β.
`str`	κείμενο	`"S355", "Ti-6Al-4V"`	ονόματα υλικών, κωδικοί
`bool`	λογική τιμή	`True, False`	«περνά / δεν περνά» (Ενότητα 4)

Python

```
n_samples = 12
density = 7.85
print(type(n_samples), type(density))
print(type(850.0), type("850"))
```

Αποτέλεσμα (output)

```
<class 'int'> <class 'float'>
<class 'float'> <class 'str'>
```

ΠΡΟΒΛΕΨΕ ΠΡΙΝ ΕΚΤΕΛΕΣΕΙΣ

Τι τύπος είναι το 850.0; Και το "850"; Σκεφτείτε το πριν τρέξετε την τελευταία γραμμή.

ΚΥΡΙΟ ΜΗΝΥΜΑ

Η Python καταλαβαίνει τον τύπο από τον τρόπο που γράφουμε την τιμή: υποδιαστολή → float, εισαγωγικά → str.

Αριθμητικοί τελεστές και σειρά πράξεων

2 · Μεταβλητές & πράξεις

Η Python ακολουθεί την ίδια προτεραιότητα με τα μαθηματικά

Τελεστής	Πράξη	Παράδειγμα
+	πρόσθεση	`T + 273.15`
-	αφαίρεση	`L - L0`
*	πολλαπλασιασμός	`rho * V`
/	διαίρεση (πάντα float)	`F / A`
**	δύναμη	`d ** 2`
//	ακέραιη διαίρεση	`7 // 2 → 3`
%	υπόλοιπο	`7 % 2 → 1`

Σειρά: παρενθέσεις → ** → * / // % → + –

Python

```
print(2 + 3 * 4)
print((2 + 3) * 4)
print(7 / 2, 7 // 2, 7 % 2)
print(2 ** 3)
```

Αποτέλεσμα (output)

```
14
20
3.5 3 1
8
```

ΠΡΟΣΟΧΗ

Η δύναμη γράφεται ** και όχι ^ όπως στο MATLAB. Στην Python το 2 ^ 3 δίνει 1 (άλλη πράξη).

Υπολογισμοί με μονάδες: πυκνότητα και μετατροπές

2 · Μεταβλητές & πράξεις

Κάθε αριθμός στην Επιστήμη Υλικών έχει μονάδα· ο κώδικας πρέπει να τη σέβεται

Python

```python
mass_g = 15.7
volume_cm3 = 2.0
density_g_cm3 = mass_g / volume_cm3
density_kg_m3 = density_g_cm3 * 1000    # g/cm³ → kg/m³

T_C = 850
T_K = T_C + 273.15                      # °C → K

ys_MPa = 355
ys_ksi = ys_MPa * 0.145038              # MPa → ksi

print(density_g_cm3, density_kg_m3)
print(T_K, round(ys_ksi, 1))
```

Αποτέλεσμα (output)

```
7.85 7850.0
1123.15 51.5
```

Μονάδα	Ισοδυναμεί με
1 g/cm³	1000 kg/m³
T (K)	T (°C) + 273.15
1 MPa	0.145038 ksi
1 N/mm²	1 MPa
x wt%	x / 100 (κλάσμα μάζας)

ΣΥΜΒΟΥΛΗ

Ένας συντελεστής μετατροπής ανά γραμμή, με σχόλιο. Αργότερα θα τους κάνουμε συναρτήσεις (Ενότητα 5).

Υπολογισμοί με φυσικό νόημα: τάση και παραμόρφωση

2 · Μεταβλητές & πράξεις

Η Python ως «επιστημονική αριθμομηχανή» που θυμάται και τεκμηριώνει

Python

```
force_N = 12000
diameter_mm = 6.0
area_mm2 = 3.1416 * (diameter_mm / 2) ** 2
stress_MPa = force_N / area_mm2          # N/mm² = MPa

L0_mm = 50.0
L_mm = 50.6
strain = (L_mm - L0_mm) / L0_mm          # αδιάστατο

print(round(area_mm2, 2), round(stress_MPa, 1))
print(strain)
```

Αποτέλεσμα (output)

```
28.27 424.4
0.012000000000000028
```

$$\sigma = F / A$$

$$\varepsilon = (L - L_0) / L_0$$

1 N/mm² = 1 MPa, άρα με F σε N και A σε mm² η τάση βγαίνει κατευθείαν σε MPa.

ΠΡΟΒΛΕΨΕ ΠΡΙΝ ΕΚΤΕΛΕΣΕΙΣ

Γιατί το strain βγαίνει 0.012000000000000028 και όχι 0.012; (Απάντηση: οι υπολογιστές αποθηκεύουν τους δεκαδικούς σε δυαδική μορφή· λύση στη διαφάνεια 16.)

ΣΥΜΒΟΥΛΗ

round(x, 2): στρογγυλεύει σε 2 δεκαδικά.

Strings και f-strings: κείμενο και αναφορά αποτελεσμάτων

2 · Μεταβλητές & πράξεις

Ονόματα υλικών, κωδικοί δειγμάτων, μονάδες, και προτάσεις με τιμές μεταβλητών

Python

```
material = "Ti-6Al-4V"
print(len(material), "Al" in material)
print("Ti" + "-" + "6Al-4V")

density = 4.4321
strain = 0.012000000000000028
print(f"{material}: {density:.2f} g/cm³")
print(f"Παραμόρφωση: {strain:.3f} ({strain:.1%})")
```

Αποτέλεσμα (output)

```
9 True
Ti-6Al-4V
Ti-6Al-4V: 4.43 g/cm³
Παραμόρφωση: 0.012 (1.2%)
```

Ανατομία ενός f-string

```
f"{density:.2f} g/cm³"
```

- **f** πριν τα εισαγωγικά
- **{ }** θέση όπου μπαίνει η τιμή
- **:.2f** δύο δεκαδικά ψηφία
- **:.1%** ποσοστό, ένα δεκαδικό

ΣΥΧΝΟ ΛΑΘΟΣ

`material = steel` → NameError: χωρίς εισαγωγικά η Python ψάχνει μεταβλητή με όνομα steel.

ΚΥΡΙΟ ΜΗΝΥΜΑ

Δείχνουμε όσα ψηφία έχουν φυσικό νόημα: το 0.012000000000000028 γίνεται 0.012.

Μετατροπή τύπων: int(), float(), str()

2 · Μεταβλητές & πράξεις

Αριθμοί που έρχονται ως κείμενο, από αρχεία ή από τον χρήστη, πρέπει να μετατραπούν

Python

```
value = "7.85"                # π.χ. διαβάστηκε από αρχείο
print(type(value))
print(float(value) * 10)

print(int("850") + 50)
print(str(850) + " °C")
print(int(7.9))               # κόβει, δεν στρογγυλεύει
```

Αποτέλεσμα (output)

```
<class 'str'>
78.5
900
850 °C
7
```

ΣΥΧΝΟ ΛΑΘΟΣ

`"7.85" + 1`

→ TypeError: δεν προσθέτουμε κείμενο και αριθμό. Πρώτα float("7.85").

ΠΡΟΣΟΧΗ

`"5" * 3 → "555"`

Χωρίς σφάλμα, αλλά λάθος αποτέλεσμα: το κείμενο επαναλαμβάνεται.

ΕΦΑΡΜΟΓΗ ΣΤΑ ΥΛΙΚΑ

Οι μετρήσεις που εξάγουν τα όργανα συχνά διαβάζονται αρχικά ως κείμενο.

Άσκηση 1: πυκνότητα ενός δοκιμίου χάλυβα

2 · Μεταβλητές & πράξεις

Εφαρμόστε τα βήματα της υπολογιστικής σκέψης

Το πρόβλημα

Ένα ορθογώνιο δοκίμιο έχει διαστάσεις 4.0 × 2.0 × 1.0 cm και μάζα 62.8 g. Υπολογίστε την πυκνότητά του σε g/cm³ και σε kg/m³ και τυπώστε μια αναφορά με f-string.

- [x] Αποθήκευσε μήκος, πλάτος, ύψος και μάζα σε μεταβλητές με μονάδες
- [x] Υπολόγισε τον όγκο: V = L · W · H
- [x] Υπολόγισε την πυκνότητα και μετάτρεψε σε kg/m³
- [x] Τύπωσε με f-string, 2 δεκαδικά για g/cm³

Αναμενόμενο αποτέλεσμα

```
Πυκνότητα: 7.85 g/cm³ = 7850 kg/m³
```

ΣΥΜΒΟΥΛΗ

Ονόματα όπως length_cm, mass_g, volume_cm3. Ο έλεγχος στο τέλος: είναι 7.85 g/cm³ λογική τιμή για χάλυβα;

ΥΠΟΔΕΙΞΗ

Για {x:.0f}: ακέραια εμφάνιση χωρίς δεκαδικά, π.χ. 7850.



Λίστες και dictionaries

Λίστες (lists): συλλογές μετρήσεων

3 · Λίστες & dictionaries

Μία μεταβλητή που κρατά πολλές τιμές με σειρά

Πέντε αποτυπώματα Vickers στο ίδιο δοκίμιο:

180	195	210	225	240

↑ *νέα τιμή με append()*

Python

```
hardness_HV = [180, 195, 210, 225]
hardness_HV.append(240)       # νέα μέτρηση
print(hardness_HV)
print(len(hardness_HV))       # πλήθος
print(sum(hardness_HV) / len(hardness_HV))    # μέση τιμή
```

Αποτέλεσμα (output)

```
[180, 195, 210, 225, 240]
5
210.0
```

Τι είναι μια λίστα

- Γράφεται με αγκύλες [ ] και κόμματα
- Διατηρεί τη σειρά των στοιχείων
- Μπορεί να αλλάξει: append(), remove()
- Επιτρέπει μικτούς τύπους: [5, 2.1, "steel"]

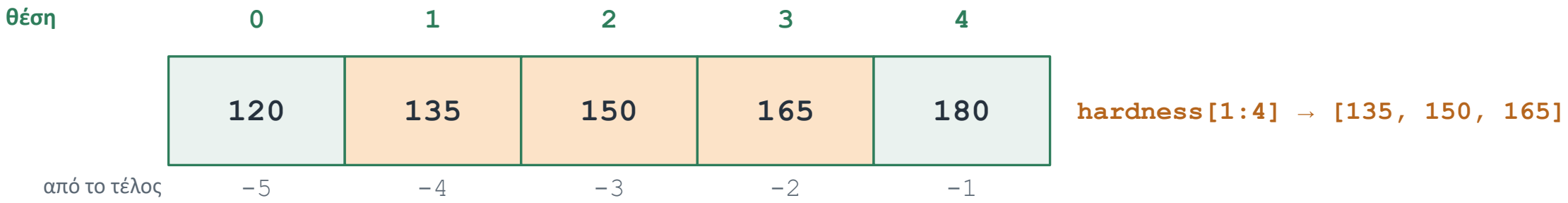
ΕΦΑΡΜΟΓΗ ΣΤΑ ΥΛΙΚΑ

Λίστα = επαναλαμβανόμενες μετρήσεις: σκληρότητα, θερμοκρασίες κατεργασίας, διάμετροι κόκκων.

Δεικτοδότηση (indexing) και τεμαχισμός (slicing)

3 · Λίστες & dictionaries

Πώς επιλέγουμε ένα ή περισσότερα στοιχεία από μια λίστα



`hardness[1:4]` → `[135, 150, 165]`

Python

```
hardness = [120, 135, 150, 165, 180]
print(hardness[0])        # πρώτο στοιχείο
print(hardness[-1])       # τελευταίο
print(hardness[1:4])      # θέσεις 1, 2, 3
print(hardness[:2])       # τα δύο πρώτα
```

Αποτέλεσμα (output)

```
120
180
[135, 150, 165]
[120, 135]
```

ΠΡΟΣΟΧΗ

Η αρίθμηση ξεκινά από 0, όχι από 1 (διαφορά από MATLAB). Στο [1:4] η αρχή περιλαμβάνεται, το τέλος όχι. Το hardness[5] δίνει IndexError.

Επεξεργασία λιστών: αλλαγή, αφαίρεση, ταξινόμηση

3 · Λίστες & dictionaries

Οι λίστες αλλάζουν: διορθώνουμε, αφαιρούμε και ταξινομούμε μετρήσεις

Python

```
hv = [205, 180, 240, 195]

hv[1] = 185              # διόρθωση μέτρησης
hv.remove(240)           # αφαίρεση τιμής
print(hv)

hv.sort()                # ταξινόμηση
print(hv)
print(min(hv), max(hv), sum(hv))
```

Αποτέλεσμα (output)

```
[205, 185, 195]
[185, 195, 205]
185 205 585
```

`lst[i] = x`	αλλάζει το στοιχείο στη θέση i
`.append(x)`	προσθέτει στο τέλος
`.remove(x)`	αφαιρεί την πρώτη τιμή x
`.sort()`	ταξινομεί τη λίστα επιτόπου
`min max sum`	ελάχιστο, μέγιστο, άθροισμα

ΕΦΑΡΜΟΓΗ ΣΤΑ ΥΛΙΚΑ

Μια μέτρηση σε άκρη αποτυπώματος ή σε πόρο; Την αφαιρούμε, αλλά πάντα καταγράφουμε γιατί.

Λεξικά (dictionaries): ζεύγη κλειδί → τιμή

3 · Λίστες & dictionaries

Ένα δείγμα υλικού με ονομασμένα πεδία

Dictionary: ένα δείγμα υλικού

`"id"`	→	`"S001"`
`"material"`	→	`"steel"`
`"C_wt_pct"`	→	`0.25`
`"hardness_HV"`	→	`185`

κλειδί (key)

τιμή (value)

Python

```
sample = {"id": "S001", "material": "steel",
          "C_wt_pct": 0.25, "hardness_HV": 185}
print(sample["hardness_HV"])
sample["yield_strength_MPa"] = 410      # νέο κλειδί
print(len(sample))
print(list(sample.keys()))
```

Αποτέλεσμα (output)

```
185
5
['id', 'material', 'C_wt_pct',
 'hardness_HV', 'yield_strength_MPa']
```

ΣΥΧΝΟ ΛΑΘΟΣ

sample["Hardness_HV"] → KeyError: το κλειδί γράφεται ακριβώς όπως ορίστηκε.

ΚΑΙ ΤΑ TUPLES;

specimen_mm = (10.0, 50.0): σαν λίστα με ( ) που δεν αλλάζει, π.χ. σταθερές διαστάσεις δοκιμίου.

Λίστα από dictionaries: ο πρώτος μας πίνακας δεδομένων

3 · Λίστες & dictionaries

Πολλά δείγματα, το ένα κάτω από το άλλο: η δομή κάθε dataset υλικών

Python

```
samples = [
    {"id": "S01", "C_wt_pct": 0.20, "hardness_HV": 160},
    {"id": "S02", "C_wt_pct": 0.30, "hardness_HV": 190},
    {"id": "S03", "C_wt_pct": 0.45, "hardness_HV": 230},
]
print(len(samples))
print(samples[1]["hardness_HV"])
```

Αποτέλεσμα (output)

```
3
190
```

ΣΥΜΒΟΥΛΗ

Το κόμμα μετά το τελευταίο στοιχείο επιτρέπεται.

	id	C_wt_pct	hardness_HV
0	S01	0.20	160
1	S02	0.30	190
2	S03	0.45	230

samples[1]["hardness_HV"]: γραμμή 1, στήλη «hardness_HV»

ΚΥΡΙΟ ΜΗΝΥΜΑ

Κάθε dictionary = μία γραμμή (ένα δείγμα).
Κάθε κλειδί = μία στήλη (μία πληροφορία).
Αυτή η δομή γίνεται DataFrame στο Pandas (Μάθημα 5).

Άσκηση 2: επεξεργασία μετρήσεων σκληρότητας

3 · Λίστες & dictionaries

Πέντε αποτυπώματα Vickers στο ίδιο δοκίμιο

212	205	198	341	208

↑ *ύποπτη τιμή: σκληρή φάση;*

Python

```
hv = [212, 205, 198, 341, 208]
```

- [ ] Τύπωσε πλήθος, μέση, ελάχιστη και μέγιστη τιμή
- [ ] Τύπωσε την πρώτη και την τελευταία μέτρηση με indexing
- [ ] Αφαίρεσε την ύποπτη τιμή και ξαναϋπολόγισε τη μέση
- [ ] Τύπωσε τη νέα μέση τιμή με 1 δεκαδικό και μονάδα

Αναμενόμενο αποτέλεσμα

```
5 232.8 198 341
212 208
Μέση χωρίς 341: 205.8 HV
```

ΕΦΑΡΜΟΓΗ ΣΤΑ ΥΛΙΚΑ

Μία ακραία τιμή ανεβάζει τη μέση κατά 27 HV. Την αφαιρούμε μόνο με φυσική αιτιολόγηση (πόρος, άκρη, σκληρή φάση) και το καταγράφουμε.

ΣΥΜΒΟΥΛΗ

Χρήσιμα: len(), sum(), min(), max(), hv[0], hv[-1], hv.remove()



Ροή ελέγχου: συνθήκες και βρόχοι

Λογικές τιμές (Boolean) και συγκρίσεις

4 · Ροή ελέγχου

True ή False: η βάση κάθε απόφασης στον κώδικα

`==` ίσο	`!=` διάφορο	`>` μεγαλύτερο
`<` μικρότερο	`>=` μεγαλύτερο ή ίσο	`<=` μικρότερο ή ίσο

`and` και οι δύο συνθήκες True

`or` τουλάχιστον μία True

`not` αντιστρέφει το αποτέλεσμα

ΕΦΑΡΜΟΓΗ ΣΤΑ ΥΛΙΚΑ

Έλεγχος προδιαγραφής: ο χάλυβας S355 απαιτεί ελάχιστο όριο διαρροής 355 MPa (για μικρά πάχη). Μια σύγκριση απαντά «περνά / δεν περνά».

Python

```python
yield_strength_MPa = 372
print(yield_strength_MPa >= 355)
print(yield_strength_MPa == 355)

hv = 240
print(hv >= 200 and hv < 300)
```

Αποτέλεσμα (output)

```
True
False
True
```

ΣΥΧΝΟ ΛΑΘΟΣ

`if ys = 355:` → SyntaxError. Η σύγκριση θέλει ==

Συνθήκες ελέγχου: if, elif, else

4 · Ροή ελέγχου

Ο κώδικας παίρνει διαφορετική διαδρομή ανάλογα με μια συνθήκη

Python

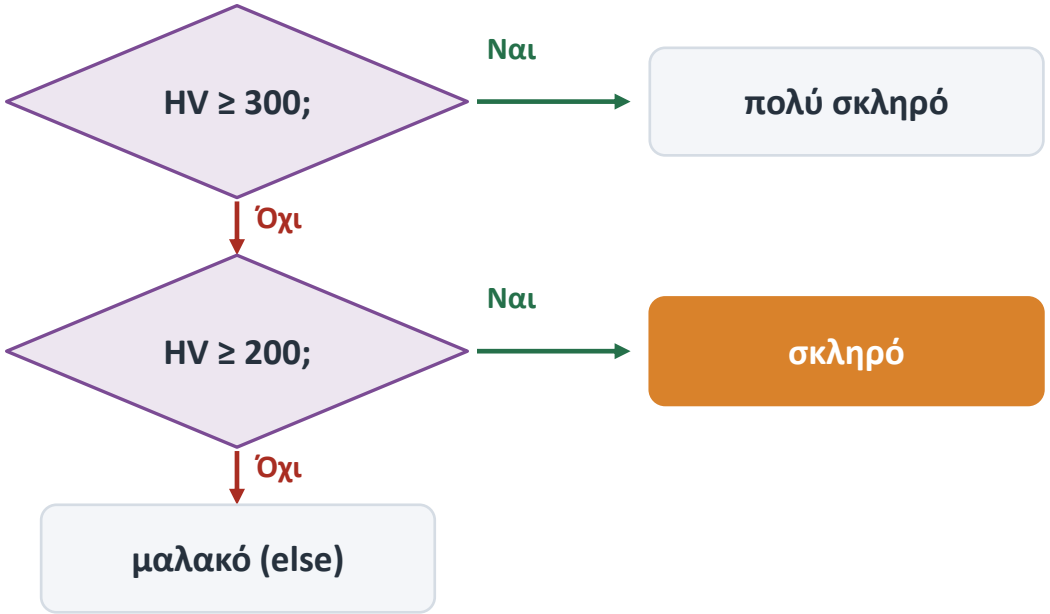
```
hardness_HV = 240

if hardness_HV >= 300:
    category = "πολύ σκληρό"
elif hardness_HV >= 200:
    category = "σκληρό"
else:
    category = "μαλακό"

print(category)
```

Αποτέλεσμα (output)

```
σκληρό
```



ΠΡΟΣΟΧΗ

Άνω-κάτω τελεία (:) στο τέλος κάθε if / elif / else και **εσοχή 4 κενών** (indentation) στις εντολές που ανήκουν στη συνθήκη.

Επανάληψη με βρόχο for

4· Ροή ελέγχου

Εκτελούμε τις ίδιες εντολές για κάθε στοιχείο μιας συλλογής

Python

```
hardness_values = [150, 175, 210, 230]

for value in hardness_values:
    if value >= 200:
        label = "πάνω από 200 HV"
    else:
        label = "κάτω από 200 HV"
    print(f"{value} HV: {label}")
```

Αποτέλεσμα (output)

```
150 HV: κάτω από 200 HV
175 HV: κάτω από 200 HV
210 HV: πάνω από 200 HV
230 HV: πάνω από 200 HV
```

Σε κάθε επανάληψη το value παίρνει την επόμενη τιμή:

επανάληψη 1	`value = 150`
επανάληψη 2	`value = 175`
επανάληψη 3	`value = 210`
επανάληψη 4	`value = 230`

ΠΡΟΣΟΧΗ

Άνω-κάτω τελεία και εσοχή, όπως στο if. Το print εδώ έχει εσοχή, άρα εκτελείται σε κάθε επανάληψη.

Το μοτίβο του «συσσωρευτή»: άθροισμα, πλήθος, μέγιστο

4 · Ροή ελέγχου

Πώς υπολογίζουμε στατιστικά με βρόχο, βήμα-βήμα

Python

```
hardness_values = [150, 175, 210, 230]

total = 0                    # 1. αρχική τιμή
count = 0
for value in hardness_values:
    total = total + value    # 2. ενημέρωση σε κάθε βήμα
    count = count + 1

print(total, count, total / count)    # 3. αποτέλεσμα
```

Αποτέλεσμα (output)

```
765 4 191.25
```

Επανάληψη	value	total	count
αρχή	—	0	0
1	150	150	1
2	175	325	2
3	210	535	3
4	230	**765**	4

ΚΥΡΙΟ ΜΗΝΥΜΑ

Τρία βήματα: αρχική τιμή πριν τον βρόχο, ενημέρωση μέσα, αποτέλεσμα μετά. Έτσι δουλεύουν και τα sum() και max().

Φιλτράρισμα με βρόχο: νέα λίστα με όσα δείγματα περνούν

4 · Ροή ελέγχου

Συνδυάζουμε for, if και append()

Python

```
ys_MPa = [340, 372, 410, 351, 395]

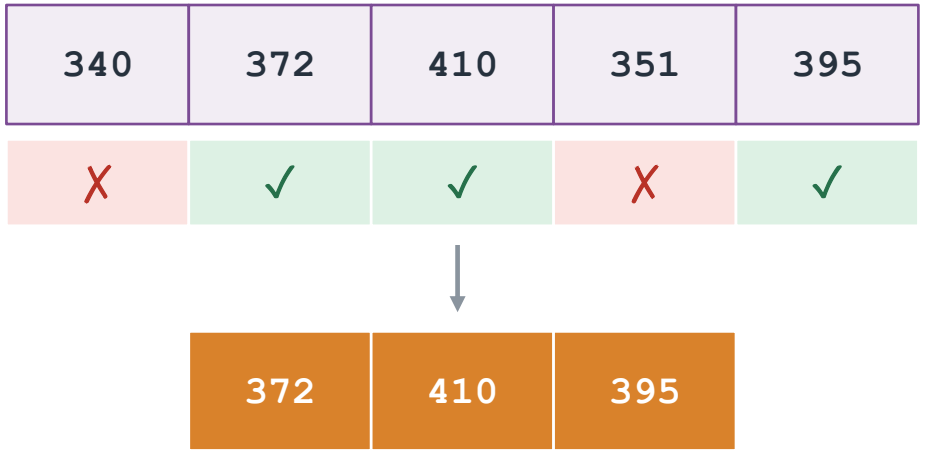
passed = []                          # κενή λίστα
for value in ys_MPa:
    if value >= 355:
        passed.append(value)

print(passed)
print(f"{len(passed)} από {len(ys_MPa)} περνούν")
```

Αποτέλεσμα (output)

```
[372, 410, 395]
3 από 5 περνούν
```

Από όλα τα δοκίμια στα αποδεκτά:



ΕΦΑΡΜΟΓΗ ΣΤΑ ΥΛΙΚΑ

Ποια δοκίμια πληρούν την προδιαγραφή S355; Στο Μάθημα 5 το ίδιο φιλτράρισμα θα γίνεται σε μία γραμμή με NumPy και Pandas.

range() και while: επανάληψη με αριθμούς ή με συνθήκη

4 · Ροή ελέγχου

Δύο ακόμη τρόποι επανάληψης, για συγκεκριμένες ανάγκες

range(): επανάληψη με αριθμούς

Python

```
for i in range(3):
    print("Δοκίμιο", i + 1)

temps_C = list(range(700, 901, 50))
print(temps_C)
```

Αποτέλεσμα (output)

```
Δοκίμιο 1
Δοκίμιο 2
Δοκίμιο 3
[700, 750, 800, 850, 900]
```

ΠΡΟΣΟΧΗ

range(a, b, βήμα): το b εξαιρείται, όπως στο slicing. Γι' αυτό 901.

while: επανάληψη όσο ισχύει μια συνθήκη

Python

```
T = 850            # °C
count = 0
while T > 100:
    T = T - 150        # απλοποιημένη ψύξη
    count = count + 1
print(minutes, T)
```

Αποτέλεσμα (output)

```
5 100
```

ΣΥΧΝΟ ΛΑΘΟΣ

Αν η συνθήκη δεν γίνει ποτέ False, ο βρόχος τρέχει για πάντα: κουμπί ■ Stop.

Η γραμμική ψύξη είναι τεχνητή απλοποίηση.

Άσκηση 3: πρόβλεψε και γράψε

4 · Ροή ελέγχου

Θερμοκρασίες θερμικής κατεργασίας · 4 λεπτά (1' πρόβλεψη, 3' κώδικας)

750	800	850	900	°C

Python

```
temps = [750, 800, 850, 900]
count = 0
for T in temps:
    if T >= 800:
        count = count + 1
print(count, temps[-2])
```

- [ ] Μέρος Α: γράψε σε χαρτί τι θα τυπώσει το κελί
- [ ] Τρέξε το κελί και έλεγξε την πρόβλεψή σου
- [ ] Μέρος Β: φτιάξε λίστα hot με τις θερμοκρασίες ≥ 800 °C
- [ ] Τύπωσε τη λίστα hot και πόσα στοιχεία έχει

Αναμενόμενο αποτέλεσμα (Μέρος Β)

```
[800, 850, 900] 3
```

ΠΡΟΒΛΕΨΕ ΠΡΙΝ ΕΚΤΕΛΕΣΕΙΣ

Πόσες φορές εκτελείται το σώμα του for; Για ποιες τιμές ισχύει T >= 800; Ποιο στοιχείο είναι το temps[-2];

ΣΥΜΒΟΥΛΗ

Χρήσιμα: hot = [], hot.append(T), len(hot). Ίδιο μοτίβο με το φιλτράρισμα της διαφάνειας 34.

Λύση: στο notebook, ενότητα 4.



Συναρτήσεις και το πρώτο μας πρόγραμμα

Συναρτήσεις (functions): ορισμός, παράμετροι, return

5 · Συναρτήσεις & πρόγραμμα

Γράφουμε μία φορά έναν υπολογισμό και τον ξαναχρησιμοποιούμε όσες φορές θέλουμε

Python

```
def mpa_to_ksi(stress_MPa):
    stress_ksi = stress_MPa * 0.145038
    return round(stress_ksi, 1)

print(mpa_to_ksi(510))
print(mpa_to_ksi(355))
```

Αποτέλεσμα (output)

```
74.0
51.5
```

ΣΥΧΝΟ ΛΑΘΟΣ

print αντί για return: η συνάρτηση εμφανίζει κάτι, αλλά δεν επιστρέφει τιμή που μπορούμε να αποθηκεύσουμε.

`def`	λέξη-κλειδί: «ορίζω συνάρτηση»
`mpa_to_ksi`	όνομα της συνάρτησης
`(stress_MPa)`	παράμετρος: η είσοδος
`εσοχή`	το σώμα της συνάρτησης
`return`	η τιμή που επιστρέφεται (έξοδος)



5 · Συναρτήσεις & πρόγραμμα

Τύποι και κανόνες της Επιστήμης Υλικών, με όνομα

Σκληρότητα Vickers από τη διαγώνιο του αποτυπώματος

HV = 1.8544 · F / d² (F σε kgf, d σε mm)

Python

```
def vickers_HV(F_kgf, d_mm):
    return 1.8544 * F_kgf / d_mm ** 2

hv = vickers_HV(10, 0.30)
print(round(hv, 1))
```

Αποτέλεσμα (output)

```
206.0
```

ΠΡΟΣΟΧΗ

Η σειρά των ορισμάτων μετράει: vickers_HV(0.30, 10) ≠ vickers_HV(10, 0.30). Για ασφάλεια: vickers_HV(F_kgf=10, d_mm=0.30).

Έλεγχος αποδοχής για πολλά δοκίμια

min_MPa=355: προεπιλεγμένη τιμή (default)

Python

```
def passes_spec(ys_MPa, min_MPa=355):
    return ys_MPa >= min_MPa

for ys in [340, 372, 410]:
    print(ys, passes_spec(ys))
print(passes_spec(400, min_MPa=460))
```

Αποτέλεσμα (output)

```
340 False
372 True
410 True
False
```

5 · Συναρτήσεις & πρόγραμμα

Τα βήματα της υπολογιστικής σκέψης σε ένα ρεαλιστικό πρόβλημα

Το πρόβλημα

Έχουμε 6 κυλινδρικά δοκίμια χάλυβα διαμέτρου 6 mm. Για καθένα γνωρίζουμε τη διαδρομή κατεργασίας και τη δύναμη στο όριο διαρροής. Θέλουμε να ελέγξουμε ποια πληρούν το ελάχιστο όριο διαρροής 355 MPa και να συνοψίσουμε τα αποτελέσματα.

id	route	force_N
T1	hot_rolled	10300
T2	hot_rolled	9600
T3	normalized	11800
T4	quenched_tempered	16900
T5	normalized	9900
T6	quenched_tempered	18400

Βήματα

1. Δεδομένα
λίστα από dictionaries

2. Υπολογισμός
συνάρτηση: σ = F / A

3. Έλεγχος
for + if: PASS / FAIL

4. Σύνοψη
συσσωρευτές + f-strings

ΠΡΟΣΟΧΗ
Οι τιμές είναι ενδεικτικές, για εξάσκηση.

Το πρόγραμμα βήμα-βήμα: έλεγχος δοκιμίων εφελκυσμού

5 · Συναρτήσεις & πρόγραμμα

Δεδομένα → συνάρτηση → έλεγχος → σύνοψη

Python

```python
def yield_strength_MPa(force_N, diameter_mm):          # 2
    area_mm2 = 3.1416 * (diameter_mm / 2) ** 2
    return force_N / area_mm2

tests = [                                              # 1
    {"id": "T1", "route": "hot_rolled",
     "force_N": 10300, "diameter_mm": 6.0},
    # ... T2–T6 με την ίδια μορφή
]

MIN_YS_MPa = 355
for test in tests:                                     # 3
    ys = yield_strength_MPa(test["force_N"], test["diameter_mm"])
    test["ys_MPa"] = ys
    if ys >= MIN_YS_MPa:
        status = "PASS"
    else:
        status = "FAIL"
    print(f"{test['id']}: {ys:.0f} MPa → {status}")
```

Python

```python
n_pass = 0                                             # 4
total = 0
for test in tests:
    total = total + test["ys_MPa"]
    if test["ys_MPa"] >= MIN_YS_MPa:
        n_pass = n_pass + 1
print(f"Πέρασαν {n_pass} από {len(tests)}")
print(f"Μέσο: {total / len(tests):.0f} MPa")
```

Αποτέλεσμα (output)

```
T1: 364 MPa → PASS
T2: 340 MPa → FAIL
T3: 417 MPa → PASS
T4: 598 MPa → PASS
T5: 350 MPa → FAIL
T6: 651 MPa → PASS
Πέρασαν 4 από 6
Μέσο: 453 MPa
```

Άσκηση σύνθεσης: αναφορά σκληρότητας

Άσκηση & Homework

Checkpoint 1
Μεταβλητές

- F και d ενός αποτυπώματος
- υπολογισμός HV
- f-string με 1 δεκαδικό

Checkpoint 2
Λίστες & βρόχοι

- λίστα με διαγωνίους
- for: HV κάθε αποτυπώματος
- μέση τιμή με συσσωρευτή

Checkpoint 3
Συναρτήσεις

- vickers_HV(F, d)
- classify(hv) με if/elif
- έλεγχος με 3 τιμές

Checkpoint 4
Αναφορά

- λίστα από dictionaries
- βρόχος: HV + κατηγορία
- σύνοψη με f-strings

ΣΥΜΒΟΥΛΗ

Σε κάθε checkpoint σηκώνετε το χέρι. Κολλήσατε; Τελευταία γραμμή του σφάλματος, μετά εσοχές και εισαγωγικά.

Τι μάθαμε σήμερα και τα πιο συχνά λάθη

Σύνοψη

Στο τέλος του μαθήματος ο/η φοιτητής/τρια μπορεί να…

Περιβάλλον

- δουλεύει σε Colab / Jupyter
- διαβάζει μηνύματα σφάλματος
- αναλύει ένα πρόβλημα σε βήματα

Μεταβλητές & πράξεις

- χρησιμοποιεί μεταβλητές και τύπους
- υπολογίζει με σωστές μονάδες
- γράφει αναφορές με f-strings

Λίστες & dictionaries

- αποθηκεύει μετρήσεις σε λίστες
- indexing και slicing
- οργανώνει δείγματα σε dictionaries

Ροή ελέγχου

- if / elif / else
- for, range, while
- συσσωρευτής και φίλτρο

Συναρτήσεις

- ορίζει def με παραμέτρους
- επιστρέφει τιμές με return
- γράφει ένα μικρό πρόγραμμα

ΤΑ 6 ΠΙΟ ΣΥΧΝΑ ΛΑΘΗ

string χωρίς εισαγωγικά · = αντί για == · ξεχασμένη «:» ή εσοχή · θέση 1 αντί για 0 · print αντί για return · κελί που δεν έτρεξε (NameError)

Homework 1 — Python για την ανάλυση μετρήσεων υλικών

Άσκηση & Homework

Παράδοση ως notebook (.ipynb) που τρέχει με Restart & Run All

Core — υποχρεωτικό

- [x] Συναρτήσεις μετατροπής: °C → K, MPa → ksi
- [x] Λίστα από dictionaries με 8 δείγματα (δίνεται)
- [x] Υπολογισμός ιδιότητας για κάθε δείγμα
- [x] Κατηγοριοποίηση με if / elif / else
- [x] Φιλτράρισμα σε νέα λίστα
- [x] Μέση τιμή, ελάχιστο, μέγιστο με βρόχο
- [x] Αναφορά με f-strings και σωστές μονάδες

Challenge — προαιρετικό

- [x] Διάμεσος με ταξινόμηση (χωρίς βιβλιοθήκες)
- [x] Τυπική απόκλιση από τον ορισμό της
- [x] Μέσος όρος ανά διαδρομή κατεργασίας με dictionary
- [x] Έλεγχος εγκυρότητας: αρνητικές ή αδύνατες τιμές

ΕΦΑΡΜΟΓΗ ΣΤΑ ΥΛΙΚΑ

Σύντομη ερμηνεία (100–150 λέξεις) σε κελί κειμένου: ποια δείγματα ξεχωρίζουν και γιατί; Είναι οι τιμές φυσικά λογικές;

ΚΡΙΤΗΡΙΑ

Σχόλια (#) στον κώδικα, μονάδες σε κάθε αποτέλεσμα.

Εξάσκηση στο σπίτι: 14 ασκήσεις για να συμπληρώσετε

Άσκηση & Homework

Στο notebook του μαθήματος, ενότητα 8. Σκανάρετε τον κωδικό για να το ανοίξετε στο Colab

https://drive.google.com/file/d/1Qob_D8or8hvTytU8utcxKnJWjYvFiw6d/view?usp=sharing

Ασκήσεις	Θέμα	Πλήθος
8.1–8.3	Μεταβλητές, πράξεις με μονάδες, f-strings	**3**
8.4–8.6	Λίστες, indexing, dictionaries	**3**
8.7–8.10	if / else, for, συσσωρευτής, φίλτρο, range	**4**
8.11–8.13	Συναρτήσεις με return	**3**
8.14	Όλα μαζί: έλεγχος τεσσάρων δοκιμίων	**1**

Πώς δουλεύετε

1. αντικαθιστάτε τα ... με τον δικό σας κώδικα (οδηγός: οι γραμμές με 🖍)
2. τρέχετε το κελί και συγκρίνετε με το «Αναμενόμενο αποτέλεσμα»
3. κολλήσατε; οι λύσεις είναι στο τέλος του notebook

ΣΥΜΒΟΥΛΗ

Μικρές ασκήσεις, μία για κάθε ιδέα του μαθήματος. Κάντε τις **πριν από το Homework 1** και δοκιμάστε πρώτα μόνοι σας, πριν δείτε τη λύση.

Επόμενο βήμα: γιατί θα χρειαστούμε βιβλιοθήκες

Σύνοψη

Μάθημα 5: Python for Machine Learning I: από τη γλώσσα στα εργαλεία ανάλυσης δεδομένων

NumPy

- αριθμητικοί πίνακες
- οι βρόχοι σας γίνονται μία πράξη

Pandas

- πίνακες δεδομένων
- η λίστα από dictionaries γίνεται DataFrame

Matplotlib

- γραφήματα
- οι μετρήσεις σας γίνονται εικόνα

scikit-learn

- μηχανική μάθηση
- πρόβλεψη ιδιοτήτων από δεδομένα

Βιβλιογραφία

- Z. “L.” Liu, Artificial Intelligence for Engineers: Basics and Implementations, Κεφ. 2 (Springer, 2025): ενότητες 2.2.1–2.2.7.
- Επίσημο tutorial: docs.python.org/3/tutorial

Ερωτήσεις;

ΣΥΜΒΟΥΛΗ

Πριν το Μάθημα 5: τελειώστε το εργαστήριο και το Homework 1.

Παράρτημα: τοπική εγκατάσταση Python και JupyterLab

Παράρτημα

Οδηγός για το σπίτι: τέσσερα βήματα, μία φορά για κάθε υπολογιστή, στο τερματικό (Terminal / Command Prompt)

1 Εγκατάσταση Python 3
Από το python.org κατεβάζουμε την τελευταία έκδοση Python 3.

```
python.org  →  Downloads
```

2 Έλεγχος εγκατάστασης
Αν εμφανιστεί Python 3.x, ο υπολογιστής βρίσκει την Python.

```
python --version
python3 --version
```

3 Εγκατάσταση JupyterLab
Το pip είναι ο διαχειριστής πακέτων: κατεβάζει και εγκαθιστά εργαλεία.

```
python -m pip install jupyterlab
```

4 Εκκίνηση
Ανοίγει το JupyterLab στον browser· εκεί δημιουργούμε νέο notebook.

```
jupyter lab
```

ΠΡΟΣΟΧΗ
Windows: στην εγκατάσταση τσεκάρετε «Add python.exe to PATH». macOS/Linux: συχνά η σωστή εντολή είναι python3 αντί για python. Τις επιστημονικές βιβλιοθήκες θα τις εγκαταστήσουμε στο Μάθημα 5.

Παράρτημα: πίνακας συνηθισμένων σφαλμάτων

Παράρτημα

Οδηγός αναφοράς για το εργαστήριο και το homework

Σφάλμα	Παράδειγμα	Διόρθωση
SyntaxError	`if T > 800  (λείπει «:»)`	Πρόσθεσε «:» ή κλείσε την παρένθεση
NameError	`print(hardnes_HV)`	Έλεγξε την ορθογραφία ή τρέξε το κελί που ορίζει τη μεταβλητή
TypeError	`"7.85" + 1`	Μετάτρεψε: float("7.85") + 1
IndentationError	`print χωρίς εσοχή μέσα σε if`	4 κενά μπροστά από κάθε εντολή του μπλοκ
IndexError	`hv[5] σε λίστα 5 στοιχείων`	Οι θέσεις είναι 0 έως len − 1
KeyError	`sample["Hardness_HV"]`	Το κλειδί γράφεται ακριβώς όπως ορίστηκε
ValueError	`hv.remove(341) δεύτερη φορά`	Η τιμή δεν υπάρχει πια στη λίστα
ZeroDivisionError	`mass_g / 0`	Έλεγξε ότι ο παρονομαστής δεν είναι 0

Πάντα: διαβάζουμε πρώτα την τελευταία γραμμή του μηνύματος, μετά τη γραμμή κώδικα που αναφέρει.