# Εισαγωγή στον Προγραμματισμό Η/Υ

Διδάσκοντες: Ι. Πολενάκης, Κ. Σκιαδόπουλος, Δ. Μουρατίδου

Τμήμα Πληροφορικής, Ιόνιο Πανεπιστήμιο

# 5. Συναρτήσεις

## Συναρτήσεις (functions) στην Python

**Μια συνάρτηση είναι ένα "κομμάτι" κώδικα που εκτελεί μια συγκεκριμένη εργασία.**

**Σκοπός της είναι να οργανώνει τον κώδικα και να αποφεύγει την επανάληψη.**

Στόχοι Μαθήματος:

- Κατανόηση συνάρτησης και γιατί τη χρησιμοποιούμε.
- Ορισμός και κλήση (εκτέλεση) συναρτήσεων στη Python.
- Πέρασμα ορισμάτων (arguments) και να επιστροφή τιμών (return values).
- Χρησιμοποίηση προεπιλεγμένων τιμών, μεταβλητό αριθμό ορισμάτων και ανώνυμες συναρτήσεις (lambda).
- Κατανόηση της διαφοράς τοπικών και καθολικών μεταβλητών.

Παράδειγμα χωρίς συνάρτηση:

```
print("Γεια σου, Μαρία!")
print("Γεια σου, Νίκο!")
print("Γεια σου, Άννα!")
```

# 5. Συναρτήσεις

## Συναρτήσεις (functions) στην Python

**Μια συνάρτηση είναι ένα "κομμάτι" κώδικα που εκτελεί μια συγκεκριμένη εργασία.**

**Σκοπός της είναι να οργανώνει τον κώδικα και να αποφεύγει την επανάληψη.**

Στόχοι Μαθήματος:

- Κατανόηση συνάρτησης και γιατί τη χρησιμοποιούμε.
- Ορισμός και κλήση (εκτέλεση) συναρτήσεων στη Python.
- Πέρασμα ορισμάτων (arguments) και να επιστροφή τιμών (return values).
- Χρησιμοποίηση προεπιλεγμένων τιμών, μεταβλητό αριθμό ορισμάτων και ανώνυμες συναρτήσεις (lambda).
- Κατανόηση της διαφοράς τοπικών και καθολικών μεταβλητών.

Με συνάρτηση:

```
def xairetismos(onoma):
    print("Γεια σου,", onoma)

xairetismos("Μαρία")
xairetismos("Νίκο")
xairetismos("Άννα")
```

# 5. Συναρτήσεις

- **Δομή μιας Συνάρτησης**

```
def onoma_synartisis(ορίσματα):
    # Σώμα συνάρτησης
    εντολές
    return τιμή
```

Παράδειγμα:

```
def athroisma(a, b):
    return a + b

print(athroisma(3, 5))  # Εκτυπώνει 8
```

# 5. Συναρτήσεις

- **Ορίσματα και Τιμές Επιστροφής**
  - Ορίσματα (arguments): τιμές που δίνουμε στη συνάρτηση.
  - Return: επιστρέφει αποτέλεσμα πίσω στο πρόγραμμα.def

Παράδειγμα:

```
def tetragono(x):
    return x ** 2

apotelesma = tetragono(4)
print(apotelesma)  # 16
```

# 5. Συναρτήσεις

- **Προεπιλεγμένες Τιμές Ορισμάτων**

Παράδειγμα:

```
def xairetismos(onoma="Φίλε"):
    print("Γεια σου,", onoma)

xairetismos()           # Γεια σου, Φίλε
xairetismos("Μαρία")    # Γεια σου, Μαρία
```

# 5. Συναρτήσεις

## Μεταβλητός Αριθμός Ορισμάτων

- *args / **kwargs (απροσδιόριστο αριθμό ονομαστικών παραμέτρων -*keyword*)

Παράδειγμα:

```
def athroisma(*arithmoi):
    return sum(arithmoi)

print(athroisma(1, 2, 3, 4))  # 10

def plirofories(**mathitis):
    print("Όνομα:", mathitis["onoma"])
    print("Ηλικία:", mathitis["ilikia"])

plirofories(onoma="Γιάννης", ilikia=15)
```

# 5. Συναρτήσεις

## Ανώνυμες Συναρτήσεις (Lambda)

Παράδειγμα:

```
tetragono = lambda x: x ** 2
print(tetragono(5))  # 25
```

**Χρήσιμες για σύντομες λειτουργίες, π.χ. μέσα σε map() ή filter():**

```
arithmoi = [1, 2, 3, 4]
tetragwna = list(map(lambda x: x**2, arithmoi))
print(tetragwna)  # [1, 4, 9, 16]
```

# 5. Συναρτήσεις

## Ανώνυμες Συναρτήσεις (Lambda)

Παράδειγμα:

```
tetragono = lambda x: x ** 2
print(tetragono(5))  # 25
```

**Χρήσιμες για σύντομες λειτουργίες, π.χ. μέσα σε map() ή filter():**

```
arithmoi = [1, 2, 3, 4]
tetragwna = list(map(lambda x: x**2, arithmoi))
print(tetragwna)  # [1, 4, 9, 16]
```

Τα **map()** και **filter()** είναι **ενσωματωμένες συναρτήσεις** (built-in functions) της Python που μας βοηθούν να επεξεργαζόμαστε **σειρές δεδομένων** (όπως λίστες, tuples, sets κ.λπ.) με **καθαρό και σύντομο τρόπο**, χωρίς να χρειάζεται να γράφουμε βρόχους for.

# 5. Συναρτήσεις

- **Ανώνυμες Συναρτήσεις (Lambda) - map**

Η map() εφαρμόζει μια συνάρτηση σε κάθε στοιχείο μιας λίστας (ή άλλης συλλογής) και επιστρέφει έναν νέο iterator με τα αποτελέσματα.

Παράδειγμα 1: **-- map(function, iterable)**

```
def tetragono(x):
    return x ** 2


arithmoi = [1, 2, 3, 4, 5]
apotelesma = map(tetragono, arithmoi)
print(list(apotelesma))
```

```
[1, 4, 9, 16, 25]
```

# 5. Συναρτήσεις

## Ανώνυμες Συναρτήσεις (Lambda) - map

Η map() εφαρμόζει μια συνάρτηση σε κάθε στοιχείο μιας λίστας (ή άλλης συλλογής) και επιστρέφει έναν νέο iterator με τα αποτελέσματα.

Παράδειγμα 2 (lamda): **-- map(function, iterable)**

```
arithmoi = [1, 2, 3, 4, 5]
apotelesma = map(lambda x: x * 10, arithmoi)

print(list(apotelesma))
```

```
[10, 20, 30, 40, 50]
```

# 5. Συναρτήσεις

- **Ανώνυμες Συναρτήσεις (Lambda) - filter**

Η filter() κρατά μόνο τα στοιχεία που ικανοποιούν μια συνθήκη (True). Η function επιστρέφει True ή False για κάθε στοιχείο.

Παράδειγμα 1: **--filter(function, iterable)**

```
def einai_arthios(x):
    return x % 2 == 0

arithmoi = [1, 2, 3, 4, 5, 6]
apotelesma = filter(einai_arthios, arithmoi)

print(list(apotelesma))
```

[2, 4, 6]

# 5. Συναρτήσεις

- **Ανώνυμες Συναρτήσεις (Lambda) - filter**

Η filter() κρατά μόνο τα στοιχεία που ικανοποιούν μια συνθήκη (True). Η function επιστρέφει True ή False για κάθε στοιχείο.

Παράδειγμα 2 (με lambda): **--filter(function, iterable)**

```
arithmoi = [5, 10, 15, 20, 25]
apotelesma = filter(lambda x: x > 12, arithmoi)

print(list(apotelesma))
```

```
[15, 20, 25]
```

# 5. Συναρτήσεις

- **Ανώνυμες Συναρτήσεις (Lambda) – map/filter**

Βρες τα τετράγωνα των άρτιων αριθμών από μια λίστα:

```
arithmoi = [1, 2, 3, 4, 5, 6]
apotelesma = map(
    lambda x: x ** 2,                    # τι θα κάνουμε
    filter(lambda x: x % 2 == 0, arithmoi)   # σε ποια στοιχεία
)
print(list(apotelesma))
```

```
[4, 16, 36]
```

# 5. Συναρτήσεις

- **Ανώνυμες Συναρτήσεις (Lambda) – map/filter**

| Συνάρτηση | Χρήση |
|---|---|
| `map()` | Όταν θέλεις να **μετασχηματίσεις** τα στοιχεία (π.χ. να τα πολλαπλασιάσεις, αλλάξεις, κ.λπ.) |
| `filter()` | Όταν θέλεις να **φιλτράρεις** τα στοιχεία (κρατώντας μόνο αυτά που ικανοποιούν μια συνθήκη) |

- Στην Python συχνά προτιμούμε τη **list comprehension**, γιατί είναι πιο αναγνώσιμη:

| Με map() και filter() | Με list comprehension |
|---|---|
| `list(map(lambda x: x**2, filter(lambda x: x%2==0, arithmoi)))` | `[x**2 for x in arithmoi if x%2==0]` |

# 5. Συναρτήσεις

- **Εμβέλεια Μεταβλητών (Scope)**

```
x = 10  # καθολική μεταβλητή

def emv():
    x = 5   # τοπική μεταβλητή
    print("Μέσα στη συνάρτηση:", x)

emv()
print("Έξω από τη συνάρτηση:", x)
```

```
Μέσα στη συνάρτηση: 5
Έξω από τη συνάρτηση: 10
```

# 5. Συναρτήσεις

- **Κλήση Μιας Συνάρτησης Μέσα σε Άλλη**

```
def tetragono(x):
    return x**2

def synolo_tetragonwn(a, b):
    return tetragono(a) + tetragono(b)

print(synolo_tetragonwn(3, 4))  # 25
```

```
Μέσα στη συνάρτηση: 5
Έξω από τη συνάρτηση: 10
```

# 5. Συναρτήσεις

- **Ενσωματωμένες Συναρτήσεις Python**

| Συνάρτηση | Περιγραφή |
| --- | --- |
| `len()` | Μήκος λίστας ή συμβολοσειράς |
| `sum()` | Άθροισμα στοιχείων |
| `max()`, `min()` | Μέγιστο και ελάχιστο |
| `sorted()` | Ταξινόμηση λίστας |
| `type()` | Τύπος δεδομένου |

# 5. Συναρτήσεις

- **Ενσωματωμένες Συναρτήσεις Python**

```
# Συνάρτηση που υπολογίζει τον μέσο όρο μιας λίστας
def mo_timon(lista):
    if len(lista) == 0:
        return 0  # αποφυγή διαίρεσης με το μηδέν
    return sum(lista) / len(lista)
```

# 5. Συναρτήσεις

- **Ενσωματωμένες Συναρτήσεις Python**

```
# Συνάρτηση που ελέγχει αν ένας αριθμός είναι άρτιος
def einai_arthios(x):
    return x % 2 == 0

# Συνάρτηση που εκτυπώνει μόνο τους άρτιους αριθμούς μιας λίστας
def ektypwse_arthious(lista):
    print("Άρτιοι αριθμοί στη λίστα:")
    for arithmos in lista:
        if einai_arthios(arithmos):
            print(arithmos)
```

# 5. Συναρτήσεις

- **Ενσωματωμένες Συναρτήσεις Python**

```
# Παράδειγμα χρήσης
arithmoi = [3, 6, 9, 12, 15, 18]

print("Μέσος όρος:", mo_timon(arithmoi))
ektypwse_arthious(arithmoi)
```

# 5. Συναρτήσεις

- **Ενσωματωμένες Συναρτήσεις Python**

```
# Παράδειγμα χρήσης
arithmoi = [3, 6, 9, 12, 15, 18]

print("Μέσος όρος:", mo_timon(arithmoi))
ektypwse_arthious(arithmoi)
```

```
Μέσος όρος: 10.5
Άρτιοι αριθμοί στη λίστα:
6
12
18
```

# 5. Συναρτήσεις

- **Παράδειγμα Εφαρμογής: Να γράψετε ένα πρόγραμμα σε Python που να περιλαμβάνει τις εξής συναρτήσεις:**
  1. **mo_timon(lista) — δέχεται μια λίστα από αριθμούς και επιστρέφει τον μέσο όρο των αριθμών. Αν η λίστα είναι κενή, να επιστρέφει 0.**
  2. **einai_arthios(x) — δέχεται έναν ακέραιο αριθμό και επιστρέφει True αν είναι άρτιος, αλλιώς False.**
  3. **ektypwse_arthious(lista) — δέχεται μια λίστα και εκτυπώνει μόνο τους άρτιους αριθμούς της, αξιοποιώντας τη συνάρτηση einai_arthios.**
  - Στο κύριο μέρος του προγράμματος:
  - Να δοθεί μια λίστα αριθμών.
  - Να υπολογιστεί και να εκτυπωθεί ο μέσος όρος.
  - Να εκτυπωθούν οι άρτιοι αριθμοί της λίστας.

# 5. Συναρτήσεις

- **Παράδειγμα Εφαρμογής: Να γράψετε ένα πρόγραμμα σε Python που να περιλαμβάνει τις εξής συναρτήσεις:**

```
# ----------------------------------------
# Συνάρτηση 1: Υπολογισμός μέσου όρου
# ----------------------------------------
def mo_timon(lista):
    """
    Επιστρέφει τον μέσο όρο των αριθμών της λίστας.
    Αν η λίστα είναι άδεια, επιστρέφει 0.
    """
    if len(lista) == 0:          # έλεγχος για κενή λίστα
        return 0
    athroisma = sum(lista)       # υπολογίζει το άθροισμα όλων των στοιχείων
    mesos_oros = athroisma / len(lista)  # διαίρεση με το πλήθος των στοιχείων
    return mesos_oros
```

# 5. Συναρτήσεις

- **Παράδειγμα Εφαρμογής: Να γράψετε ένα πρόγραμμα σε Python που να περιλαμβάνει τις εξής συναρτήσεις:**

```
# ----------------------------------------
# Συνάρτηση 2: Έλεγχος αν ένας αριθμός είναι άρτιος
# ----------------------------------------
def einai_arthios(x):
    """
    Επιστρέφει True αν ο αριθμός x είναι άρτιος (διαιρείται με το 2
                                                    χωρίς υπόλοιπο),
    αλλιώς False.
    """
    return x % 2 == 0
```

# 5. Συναρτήσεις

- **Παράδειγμα Εφαρμογής: Να γράψετε ένα πρόγραμμα σε Python που να περιλαμβάνει τις εξής συναρτήσεις:**

```
# ----------------------------------------
# Συνάρτηση 3: Εκτύπωση μόνο των άρτιων αριθμών
# ----------------------------------------
def ektypwse_arthious(lista):
    """
    Εκτυπώνει όλους τους άρτιους αριθμούς μιας λίστας
    χρησιμοποιώντας τη συνάρτηση einai_arthios().
    """
    print("Άρτιοι αριθμοί στη λίστα:")
    for arithmos in lista:
        if einai_arthios(arithmos):  # καλούμε τη δεύτερη συνάρτηση
            print(arithmos)
```

# 5. Συναρτήσεις

- **Παράδειγμα Εφαρμογής: Να γράψετε ένα πρόγραμμα σε Python που να περιλαμβάνει τις εξής συναρτήσεις:**

```
# ----------------------------------------
# Κύριο πρόγραμμα
# ----------------------------------------

# Δημιουργούμε μια λίστα με αριθμούς
arithmoi = [3, 6, 9, 12, 15, 18, 21, 24]

# Υπολογίζουμε και εμφανίζουμε τον μέσο όρο
mo = mo_timon(arithmoi)
print("Ο μέσος όρος των αριθμών είναι:", mo)

# Εκτυπώνουμε τους άρτιους αριθμούς
ektypwse_arthious(arithmoi)
```

```
Ο μέσος όρος των αριθμών είναι: 13.5
Άρτιοι αριθμοί στη λίστα:
6
12
18
24
```

# 5. Συναρτήσεις

- **Παράδειγμα Εφαρμογής: Να γράψετε ένα πρόγραμμα σε Python που να περιλαμβάνει τις εξής συναρτήσεις:**

  - Το πρόγραμμα είναι χωρισμένο σε σαφώς ορισμένες συναρτήσεις, γεγονός που:
  - Διευκολύνει την ανάγνωση και συντήρηση του κώδικα.
  - Προωθεί την επαναχρησιμοποίηση.
  - Η mo_timon() διαχειρίζεται σωστά την περίπτωση κενής λίστας (καλή πρακτική).
  - Η einai_arthios() είναι σύντομη και καθαρή, χρησιμοποιώντας απευθείας τη συνθήκη x % 2 == 0.
  - Η ektypwse_arthious() επανεκμεταλλεύεται τη δεύτερη συνάρτηση — δείχνοντας σύνθεση συναρτήσεων, μια σημαντική δεξιότητα στον προγραμματισμό.

# 5. Συναρτήσεις

- **Παράδειγμα Εφαρμογής: Να γράψετε ένα πρόγραμμα σε Python που να περιλαμβάνει τις εξής συναρτήσεις:**

  - **Η ίδια λειτουργία με πιο "Pythonic" τρόπο — δηλαδή με λίστες και filter():**

```
arithmoi = [3, 6, 9, 12, 15, 18, 21, 24]

# Υπολογισμός μέσου όρου
mo = sum(arithmoi) / len(arithmoi)

# Εύρεση άρτιων αριθμών με filter + lambda
arthioi = list(filter(lambda x: x % 2 == 0, arithmoi))

print("Μέσος όρος:", mo)
print("Άρτιοι:", arthioi)
```

Μέσος όρος: 13.5
Άρτιοι: [6, 12, 18, 24]

# 5. Συναρτήσεις

- **Ένα αυτοκίνητο κινείται με σταθερή ταχύτητα. Γνωρίζουμε την απόσταση που πρέπει να διανύσει και την ταχύτητά του. Να γράψετε ένα πρόγραμμα σε Python που:**

  1. Ζητά από τον χρήστη την απόσταση (σε km) και την ταχύτητα (σε km/h).
  2. Υπολογίζει τον χρόνο που απαιτείται για να ολοκληρωθεί η διαδρομή.
  3. Εμφανίζει τον χρόνο τόσο σε ώρες όσο και σε λεπτά.
  4. Ελέγχει αν η ταχύτητα είναι μηδενική και εμφανίζει μήνυμα σφάλματος.

# 5. Συναρτήσεις

- **Ένα αυτοκίνητο κινείται με σταθερή ταχύτητα. Γνωρίζουμε την απόσταση που πρέπει να διανύσει και την ταχύτητά του. Να γράψετε ένα πρόγραμμα σε Python που:**
  1. Ζητά από τον χρήστη την απόσταση (σε km) και την ταχύτητα (σε km/h).
  2. Υπολογίζει τον χρόνο που απαιτείται για να ολοκληρωθεί η διαδρομή.
  3. Εμφανίζει τον χρόνο τόσο σε ώρες όσο και σε λεπτά.
  4. Ελέγχει αν η ταχύτητα είναι μηδενική και εμφανίζει μήνυμα σφάλματος.

Από τον νόμο της ομαλής κίνησης έχουμε:

$$t = \frac{s}{v}$$

όπου:

$s$= απόσταση (σε km), $v$= ταχύτητα (σε km/h), $t$= χρόνος (σε ώρες)

Για μετατροπή σε λεπτά:

$$t_{\lambda\varepsilon\pi\tau\alpha} = t \times 60$$

# 5. Συναρτήσεις

- **Ένα αυτοκίνητο κινείται με σταθερή ταχύτητα. Γνωρίζουμε την απόσταση που πρέπει να διανύσει και την ταχύτητά του. Να γράψετε ένα πρόγραμμα σε Python που:**

```
# ----------------------------------------
# Υπολογισμός Χρόνου Διαδρομής Αυτοκινήτου
# ----------------------------------------
def ypologismos_xronou(apostasi, taxytita):
    """
    Υπολογίζει τον χρόνο διαδρομής ενός οχήματος
    με βάση την απόσταση (km) και την ταχύτητα (km/h).
    Επιστρέφει τον χρόνο σε ώρες και λεπτά.
    """
    if taxytita <= 0:
        return None  # μη αποδεκτή ταχύτητα

    xronos_ores = apostasi / taxytita
    xronos_lepta = xronos_ores * 60

    return xronos_ores, xronos_lepta
```

# 5. Συναρτήσεις

- **Ένα αυτοκίνητο κινείται με σταθερή ταχύτητα. Γνωρίζουμε την απόσταση που πρέπει να διανύσει και την ταχύτητά του. Να γράψετε ένα πρόγραμμα σε Python που:**

```
# ----------------------------------------
# Κύριο πρόγραμμα
# ----------------------------------------

# Είσοδος δεδομένων από τον χρήστη
apostasi = float(input("Δώσε την απόσταση (km): "))
taxytita = float(input("Δώσε την ταχύτητα (km/h): "))

# Κλήση της συνάρτησης
apotelesma = ypologismos_xronou(apostasi, taxytita)

# Έλεγχος και εμφάνιση αποτελεσμάτων
if apotelesma is None:
    print("Η ταχύτητα πρέπει να είναι μεγαλύτερη από 0 km/h!")
else:
    ores, lepta = apotelesma
    print(f"\n Χρόνος διαδρομής:")
    print(f"{ores:.2f} ώρες ή περίπου {lepta:.0f} λεπτά.")
```

# 5. Συναρτήσεις

- **Ένα αυτοκίνητο κινείται με σταθερή ταχύτητα. Γνωρίζουμε την απόσταση που πρέπει να διανύσει και την ταχύτητά του. Να γράψετε ένα πρόγραμμα σε Python που:**

```
Δώσε την απόσταση (km): 150
Δώσε την ταχύτητα (km/h): 75

Χρόνος διαδρομής:
2.00 ώρες ή περίπου 120 λεπτά.
```

  - **Επεξήγηση Κώδικα**
    - Συνάρτηση ypologismos_xronou(apostasi, taxytita)
    - Υλοποιεί τον τύπο $t = s/v$.
    - Επιστρέφει δύο τιμές: τον χρόνο σε ώρες και σε λεπτά.
    - Ελέγχει αν η ταχύτητα είναι $\leq 0$ και χειρίζεται την περίπτωση αυτή.
  - **Κύριο πρόγραμμα**
    - Χρησιμοποιεί input() για είσοδο δεδομένων.
    - Καλεί τη συνάρτηση και λαμβάνει πίσω τις τιμές.
    - Εκτυπώνει καθαρά το αποτέλεσμα με μορφοποίηση (:.2f για 2 δεκαδικά ψηφία).

# 5. Συναρτήσεις

- **Ένα αυτοκίνητο κινείται με σταθερή ταχύτητα. Γνωρίζουμε την απόσταση που πρέπει να διανύσει και την ταχύτητά του. Να γράψετε ένα πρόγραμμα σε Python που:**

Έλεγχος με διαφορετικά δεδομένα

| Απόσταση (km) | Ταχύτητα(km/h) | Χρόνος (ώρες) | Χρόνος (λεπτά) |
|---|---|---|---|
| 120 | 60 | 2.00 | 120 |
| 150 | 100 | 1.50 | 90 |
| 200 | 80 | 2.50 | 150 |
| 0 | 100 | 0.00 | 0 |
| 100 | 0 | Σφάλμα | — |

# 5. Συναρτήσεις

- **Ένας οδηγός θέλει να υπολογίσει πόσο καύσιμο θα χρειαστεί και πόσο θα κοστίσει ένα ταξίδι με το αυτοκίνητό του.**

  **Γνωρίζει:**

  - **Την απόσταση του ταξιδιού (σε km)**
  - **Την μέση κατανάλωση του αυτοκινήτου (σε λίτρα/100 km)**
  - **Την τιμή του καυσίμου ανά λίτρο (σε ευρώ/λίτρο)**

  **<u>Να γράψετε ένα πρόγραμμα που</u>:**

  1. Ζητά από τον χρήστη τα παραπάνω στοιχεία.
  2. Υπολογίζει το **σύνολο καυσίμου** που θα καταναλωθεί.
  3. Υπολογίζει το **συνολικό κόστος** του ταξιδιού.
  4. Εμφανίζει καθαρά τα αποτελέσματα.
  5. Ελέγχει αν τα δεδομένα είναι έγκυρα (π.χ. όχι αρνητικές τιμές).

**Μαθηματικό Μοντέλο**

$$\text{Κατανάλωση (λίτρα)} = \frac{\text{Απόσταση} \times \text{Κατανάλωση ανά 100 km}}{100}$$

$$\text{Κόστος} = \text{Κατανάλωση (λίτρα)} \times \text{Τιμή καυσίμου}$$

# 5. Συναρτήσεις

- **Ένας οδηγός θέλει να υπολογίσει πόσο καύσιμο θα χρειαστεί και πόσο θα κοστίσει ένα ταξίδι με το αυτοκίνητό του.**

```
# ------------------------------------------
# Υπολογισμός Κατανάλωσης και Κόστους Ταξιδιού
# ------------------------------------------

def ypologismos_kaysimou(apostasi, katanalosi_ana100, timi_litrou):
    """
    Υπολογίζει την ποσότητα καυσίμου και το συνολικό κόστος ταξιδιού.
    Επιστρέφει (litres, kostos) αν τα δεδομένα είναι έγκυρα,
    αλλιώς None.
    """
    # Έλεγχος εγκυρότητας δεδομένων
    if apostasi < 0 or katanalosi_ana100 <= 0 or timi_litrou <= 0:
        return None

    # Υπολογισμός καυσίμου και κόστους
    litra = (apostasi * katanalosi_ana100) / 100
    kostos = litra * timi_litrou

    return litra, kostos
```

# 5. Συναρτήσεις

- **Ένας οδηγός θέλει να υπολογίσει πόσο καύσιμο θα χρειαστεί και πόσο θα κοστίσει ένα ταξίδι με το αυτοκίνητό του.**

```
# ------------------------------------------
# Κύριο πρόγραμμα
# ------------------------------------------

# Είσοδος δεδομένων από τον χρήστη
apostasi = float(input("Δώσε την απόσταση του ταξιδιού (km): "))
katanalosi = float(input("Δώσε τη μέση κατανάλωση (λίτρα/100km): "))
timi = float(input("Δώσε την τιμή καυσίμου (€/λίτρο): "))

# Κλήση της συνάρτησης
apotelesma = ypologismos_kaysimou(apostasi, katanalosi, timi)

# Έλεγχος και εκτύπωση αποτελεσμάτων
if apotelesma is None:
    print("Δώσε θετικές τιμές για απόσταση, κατανάλωση και τιμή!")
else:
    litra, kostos = apotelesma
    print("\n ΑΠΟΤΕΛΕΣΜΑΤΑ ΤΑΞΙΔΙΟΥ:")
    print(f"Συνολικά λίτρα καυσίμου: {litra:.2f} L")
    print(f"Συνολικό κόστος ταξιδιού: {kostos:.2f} €")
```

# 5. Συναρτήσεις

- **Ένας οδηγός θέλει να υπολογίσει πόσο καύσιμο θα χρειαστεί και πόσο θα κοστίσει ένα ταξίδι με το αυτοκίνητό του.**

```
Δώσε την απόσταση του ταξιδιού (km): 350
Δώσε τη μέση κατανάλωση (λίτρα/100km): 6.5
Δώσε την τιμή καυσίμου (€/λίτρο): 1.85

ΑΠΟΤΕΛΕΣΜΑΤΑ ΤΑΞΙΔΙΟΥ:
Συνολικά λίτρα καυσίμου: 22.75 L
Συνολικό κόστος ταξιδιού: 42.09 €
```

# 5. Συναρτήσεις

- **Ένας οδηγός θέλει να υπολογίσει πόσο καύσιμο θα χρειαστεί και πόσο θα κοστίσει ένα ταξίδι με το αυτοκίνητό του.**

**Ανάλυση και Επεξήγηση Κώδικα**

- **Συνάρτηση ypologismos_kaysimou()**
  - Ελέγχει αν τα δεδομένα είναι λογικά (θετικές τιμές).
  - Χρησιμοποιεί τους φυσικούς τύπους υπολογισμού.
  - Επιστρέφει δύο τιμές: λίτρα καυσίμου και κόστος.
- **Κύριο πρόγραμμα:**
  - Παίρνει είσοδο από τον χρήστη.
  - Εκτελεί τον υπολογισμό.
  - Εμφανίζει μορφοποιημένα αποτελέσματα με δύο δεκαδικά ψηφία (:.2f).

# 5. Συναρτήσεις

- **Ένας οδηγός θέλει να υπολογίσει πόσο καύσιμο θα χρειαστεί και πόσο θα κοστίσει ένα ταξίδι με το αυτοκίνητό του.**

**Έλεγχος με διαφορετικά δεδομένα**

| Απόσταση (km) | Κατανάλωση (L/100km) | Τιμή (€/L) | Λίτρα | Κόστος (€) |
|---|---|---|---|---|
| 100 | 5 | 1.80 | 5.00 | 9.00 |
| 250 | 7 | 1.90 | 17.50 | 33.25 |
| 400 | 8 | 1.75 | 32.00 | 56.00 |