$Αλγεβρα$
Μαθηματικά Γυμνασίου με Python

Δημήτρης Νικολός

1 Ιουνίου 2020

Κεφάλαιο 1

Δεκαδικοί αριθμοί

1.1 Εισαγωγή

Αν χωρίσουμε τη μονάδα σε 10 ίσα μέρη τότε μπορούμε να πάρουμε κλάσματα της μονάδας όπως $\frac{3}{10}$, $\frac{5}{10}$ κλπ. Τα κλάσματα είναι ομώνυμα συγκρίνονται εύκολα και βοηθάνε στις πράξεις. Γενικότερα, ονομάζουμε δεκαδικό κλάσμα οποιδήποτε κλάσμα έχει παρονομαστή μια δύναμη του 10. Κάθε δεκαδικό κλάσμα γράφεται σαν δεκαδικός αριθμός με τόσα δεκαδικά ψηφία όσα μηδενικά έχει ο παρονομαστής του. Η Python χειρίζεται τους δεκαδικούς αριθμούς όπως και τους υπόλοιπους. Δοκίμασε:

```
>>> 0.3 + 0.5
0.8
>>> type(0.7)
<class 'float'>
```

Βλέπουμε ότι οι δεκαδικοί αριθμοί δεν είναι int, όπως οι ακέραιο αλλά float. Το όνομα float έχει να κάνει με τον τρόπο με τον οποίο ο υπολογιστής αποθηκεύει αποδοτικά αυτούς τους αριθμούς.

Ας συνδυάσουμε τις γνώσεις από τα κλάσματα με τα κλάσματα που μάθαμε στο προηγούμενο κεφάλαιο.

```
>>> from fractions import Fraction
>>> x = Fraction(3,10)
>>> float(x)
0.3
```

Το `Fraction(3,10)` εννοεί το κλάσμα $\frac{3}{10}$ που είναι ίσο με 0,3. Ομως στην Python το 0,3 θα το γράφουμε με 0.3. Με τη συνάρτηση float μετατρέπουμε το $\frac{3}{10}$ σε δεκαδικό αριθμό.

Ασκηση 1.1.1 *(Στο βιβλίο βρίσκεται στη Σελ. 56) Γράψτε τους αριθμούς* $\frac{3}{10}$, $\frac{825}{1000}$, $\frac{53}{1000}$, $\frac{1004}{10000}$.

```
>>> float(Fraction(3,10))
0.625
>>> float(Fraction(825,100))
8.25
>>> float(Fraction(53,1000))
0.053
>>> float(Fraction(1004,10000))
0.1004
```

Η Python μπορεί να μετατρέψει τα κλάσματα σε δεκαδικό αριθμό ανεξάρτητα από τον παρονομαστή.

Ασκηση 1.1.2 *(Στο βιβλίο βρίσκεται στη Σελ. 59) Γράψε καθένα από τα παρακάτω κλάσματα, ως δεκαδικό αριθμό: (i) με προσέγγιση εκατοστού και (ii) με προσέγγιση χιλιοστού:*

(α) $\frac{7}{16}$
(β) $\frac{21}{17}$
(γ) $\frac{20}{95}$

```
>>> x = Fraction(7,16)
>>> float(x)
0.4375
>>> round(float(x),2)
0.44
>>> round(float(x),3)
0.438
>>> x = Fraction(21,17)
>>> float(x)
1.2352941176470589
>>> round(float(x),2)
1.24
>>> round(float(x),3)
1.235
>>> x = Fraction(20,95)
>>> float(x)
0.21052631578947367
>>> round(float(x),2)
0.21
>>> round(float(x),3)
0.211
```

Η στρογγυλοποίηση των δεκαδικών υλοποιείται στην Python με τη συνάρτηση round. Οπότε μπορείς να στρογγυλοποιήσεις εύκολα δεκαδικούς αριθμούς ως εξής:

Ασκηση 1.1.3 *Να στρογγυλοποιήσεις τους παρακάτω δεκαδικούς αριθμούς στο δέκατο, εκατοστό και χιλιοστό:*

(α) 9876,008,
(β) 67,8956,
(γ) 0,001,
(δ) 8,239,
(ε) 23,7048.

Θυμόμαστε να αλλάζουμε την υποδιαστολή από κόμμα σε τελεία:

```
def roundall(x):
    print(round(x,1))
    print(round(x,2))
    print(round(x,3))

roundall(9876.008)
roundall(67.8956)
roundall(0.001)
roundall(8.239)
roundall(23.7048)
```

Το αποτέλεσμα είναι:

```
67.9
67.9
67.896
0.0
0.0
0.001
8.2
8.24
8.239
23.7
23.7
23.705
```

Ασκηση 1.1.4 *(Στο βιβλίο βρίσκεται στη Σελ. 59) Στον αριθμό* $34,\square\square\square$ *λείπουν τα τελευταία τρία ψηφία του. Να συμπληρώσεις τον αριθμό με τα ψηφία 9, 5 και 2, έτσι ώστε κάθε ψηφίο να γράφεται μία μόνο φορά. Να γράψεις όλους τους δεκαδικούς που μπορείς να βρεις και να τους διατάξεις σε φθίνουσα σειρά.*

Πώς μπορεί η Python να βρει όλους τους πιθανούς συνδυασμούς του 9,5,2; Δοκίμασε τη βιβλιοθήκη itertools και συγκεκριμένα τη συνάρτηση permutations.

```
>>> from itertools import permutations
>>> x = permutations([1,2,3])
>>> print(x)
<itertools.permutations object at 0x012BE1B0>
```

```
>>> print(list(x))
[(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1)]
```

Ετσι με την permutations μπορείς να βρεις όλες τις αναδιατάξεις των αριθμών. Οπότε τώρα το πρόγραμμα μπορεί να γίνει ως εξής:

```
lista = []
from itertools import permutations
for p in permutations([9,5,2]):
    lista.append(34+p[0]/10+p[1]/100+p[2]/1000)
print(lista)
```

Που δίνει το αποτέλεσμα:

```
[34.952, 34.925000000000004, 34.592000000000006, 34.529,
34.29500000000001, 34.259]
```

Τα ψηφία που εμφανίζονται στο τέλος των αριθμών προκύπτουν από την αναπαράσταση των δεκαδικών στον υπολογιστή που υπόκειται σε κάποιους περιορισμούς. Αν δεν θέλουμε να εμφανίζονται μπορούμε να αλλάξουμε το for σε:

```
for p in permutations([9,5,2]):
    ar = 34+p[0]/10+p[1]/100+p[2]/1000
    lista.append(round(ar,3))
```

Τώρα για να γράψουμε τους αριθμούς με φθίνουσα σειρά θα δοκιμάσουμε τη sorted. Η sorted ταξινομεί τους αριθμούς που δίνονται σε μια λίστα. Δοκίμασε:

```
>>> sorted([4,2,3])
[2, 3, 4]
```

Ετσι το συνολικό πρόγραμμα γίνεται:

```
lista = []
from itertools import permutations
for p in permutations([9,5,2]):
    ar = 34+p[0]/10+p[1]/100+p[2]/1000
    lista.append(round(ar,3))
print(sorted(lista))
```

Που δίνει το αποτέλεσμα:

```
[34.259, 34.295, 34.529, 34.592, 34.925, 34.952]
```

Ομως η άσκηση μας ζητάει να τυπώσουμε τη λίστα με φθίνουσα σειρά. Αυτό μπορεί να γίνει δηλώνοντας στη sorted ότι θέλουμε αντίστροφη σειρά γράφοντας `reverse=True`. Το τελικό πρόγραμμα είναι το εξής:

```
lista = []
from itertools import permutations
for p in permutations([9,5,2]):
```

```
    ar = 34+p[0]/10+p[1]/100+p[2]/1000
    lista.append(round(ar,3))
print(sorted(lista,reverse=True))
```

Μια μικρή τροποποίηση που μπορεί να γίνει για να εμφανιστούν οι αριθμοί σε διαφορετικές γραμμές είναι να τυπώσουμε τη λίστα με μια for.

```
lista = []
from itertools import permutations
for p in permutations([9,5,2]):
    ar = 34+p[0]/10+p[1]/100+p[2]/1000
    lista.append(round(ar,3))

for x in sorted(lista,reverse=True):
    print(x)
```

Ασκηση 1.1.5 *(Στο βιβλίο βρίσκεται στη Σελ. 61) Να υπολογίσεις τα αθροίσματα:*

(α) $48, 18 + 3, 256 + 7, 129$

(β) $3, 59 + 7, 13 + 8, 195$

```
>>> 48.18+3.256+7.129
58.565
>>> 3.59 + 7.13 + 8.195
18.915
```

Ασκηση 1.1.6 *(Στο βιβλίο βρίσκεται στη Σελ. 61) Να υπολογίσεις το μήκος της περιμέτρου των οικοπέδων: (Σχήμα —)*

```
>>> 26.14 + 80.19 + 29.13+38.13+23.24+57.89+80.19
334.91
>>> 39.93+80.19+57.89+47.73+44.75+48.9+47.19
366.58
```

Ασκηση 1.1.7 *(Στο βιβλίο βρίσκεται στη Σελ. 61) Να κάνεις τις διαρέσεις: (α)* $579 : 48$

(β) $314 : 25$

(γ) $520 : 5, 14$

(δ) $49, 35 : 7$

```
>>> 579/48
12.0625
>>> 314/25
12.56
```

```
>>> 520/5.14
101.16731517509729
>>> 49.35/7
7.05
```

Ασκηση 1.1.8 *(Στο βιβλίο βρίσκεται στη Σελ. 61) Να κάνεις τις πράξεις:*

(α) $520 \cdot 0,1 + 0,32 \cdot 100$

(β) $4,91 \cdot 0,01 + 0,819 \cdot 10$

```
>>> 520*0.1 + 0.32*100
84.0
>>> 4.91*0.01 + 0.819*10
8.239099999999999
\end
```

Σε αυτή την άσκηση βλέπουμε ότι ο υπολογιστής προσεγγίζει τα αποτελέσματα με τον δικό του τρόπο. Δοκίμασε:

```
>>> x = 520*0.1 + 0.32*100
>>> x
84.0
>>> type(x)
<class 'float'>
>>> y = int(x)
>>> type(y)
<class 'int'>
>>> x == y
True
```

Αυτό σημαίνει πως ο ακέραιος αριθμός 84, και κάθε ακέραιος, στην Python μπορεί να αναπαρασταθεί σαν ακέραιος αλλά και σαν float με μηδενικά δεκαδικά ψηφία. Στην δεύτερη πράξη παρατηρούμε ότι αντί για το σωστό αποτέλεσμα που είναι $0,0491 + 8,19 = 8,2391$ η Python εμφανίζει μια προσέγγιση που είναι 8.239099999999999. Η διαφορά είναι πολύ μικρή. Ωστόσο οι δύο ποσότητες δεν είναι ίσες. Δοκίμασε:

```
>>> 4.91*0.01 + 0.819*10 == 8.2391
False
>>> 8.2391 - 4.91*0.01 + 0.819*10
1.7763568394002505e-15
```

Ο αριθμός `1.7763568394002505e-15` σημαίνει πως η διαφορά είναι περίπου $1.77 \cot 10^{-15}$ που είναι πάρα πολύ μικρή και προκύπτει από τον τρόπο με τον οποίο η Python αποθηκεύει τους αριθμούς.

Ασκηση 1.1.9 *(Στο βιβλίο βρίσκεται στη Σελ. 61) Να κάνεις τις πράξεις:*

(α) $4,7 : 0,1 - 45 : 10$

(β) $0,98 : 0,0001 - 6785 : 1000$

```
>>> 4.7/0.1 - 45/10
42.5
>>> 0.98/0.0001 - 6785/1000
9793.215
```

Βλέπουμε ότι η Python υπολογίζει σωστά πρώτα τη διαίρεση και μετά την αφαίρεση.

Ασκηση 1.1.10 *(Στο βιβλίο βρίσκεται στη Σελ. 61) Η περίμετρος ενός τετραγώνου είναι 20,2. Να υπολογίσεις την πλευρά του.*

```
>>> 20.2/4
5.05
```

Ασκηση 1.1.11 *(Στο βιβλίο βρίσκεται στη Σελ. 61) Η περίμετρος ενός ισοσκελούς τριγώνου είναι 48,52. Αν η βάση του είναι 10,7, πόσο είναι η κάθε μία από τις ίσες πλευρές του;*

Αφαιρούμε πρώτα από το 48,52 το 10,7. Το αποτελέσμα το διαιρούμε με το δυο.

```
>>> 48.52-10.7
37.82000000000001
>>> 37.82/2
18.91
```

Ασκηση 1.1.12 *(Στο βιβλίο βρίσκεται στη Σελ. 61) Να υπολογίσεις τις τιμές των αριθμητικών παραστάσεων:*

(α) $24 \cdot 5 - 2 + 3 \cdot 5$

(β) $3 \cdot 11 - 2 + 45,1 : 2$

```
>>> 24*5 - 2 +3*5
133
>>> 3*11 - 2 + 54.1/2
58.05
```

Ασκηση 1.1.13 *(Στο βιβλίο βρίσκεται στη Σελ. 61) Να υπολογίσεις τις δυνάμεις: (α)* $3,1^2$*, (β)* $7,01^2$*, (γ)* $4,5^2$*, (δ)* $0,5^2$*, (ε)* $0,2^2$*, (στ)* $0,3^3$

```
>>> 3.1**2
9.610000000000001
>>> 7.01**2
49.1401
>>> 4.5**2
```

```
20.25
>>> 0.5**2
0.25
>>> 0.2**2
0.04000000000000001
>>> 0.3*3
0.8999999999999999
```

Πάλι κάνουν την εμφάνισή τους μικρές προσεγγίσεις.

Ασκηση 1.1.14 *Τοποθέτησε ένα "x" στην αντίστοιχη θέση (ΣΩΣΤΟ ΛΑΘΟΣ) (α)* $2,75+0,05+1,40+16,80=21$ *(β)* $420,510+72,490+45,19+11,81=500$ *(γ)* $4–3,852=1,148$ *(δ)* $32,01–4,001=28,01$ *(ε)* $41900\cdot 0,0001–0,0419\cdot 1000=0$ *(στ)* $56,89\cdot 0,01+4311:10000=1$ *(ζ)* $(3,2+7,2\cdot 2+24\cdot 0,1):100=0,2$

(α)

```
>>> 2.75 + 0.05 + 1.40 + 16.80 == 21
True
>>> 2.75 + 0.05 + 1.40 + 16.80
21.0
```

Αρα Σωστό

(β)

```
>>> 420.510 + 72.490 + 45.19 + 11.81 == 500
False
>>> 420.510 + 72.490 + 45.19 + 11.81
550.0
```

Αρα Λάθος

(γ)

```
>>> 4 - 3.852 == 1.148
False
>>> 4 - 3.852
0.14800000000000013
```

Αρα Λάθος

(δ)

```
>>> 32.01 - 4.001 == 28.01
False
>>> 32.01 - 4.001
28.008999999999997
```

Αρα Λάθος

(ε)

```
>>> 41900*0.0001 - 0.0419*1000 == 0
False
>>> 41900*0.0001 - 0.0419*1000
-37.71
```

Αρα Λάθος

(στ)

```
>>> 56.89*0.01 + 4311 / 10000 == 1
True
>>> 56.89*0.01 + 4311 / 10000
1.0
```

Αρα Σωστό

και

(ζ)

```
>>> (3.2 + 7.2*2 + 24*0.1) / 100 == 0.2
True
>>> (3.2 + 7.2*2 + 24*0.1) / 100
0.2
```

Αρα Σωστό.

1.2 Τυποποιημένη μορφή μεγάλων αριθμών

Η μάζα του Ηλιου είναι 1983οοοοοοοοοοοοοοοοοοοοοοοοο κιλά. Για να διευκολυνθούμε μπορούμε να γράψουμε αυτόν τον αριθμό στην μορφή $\alpha \cdot 10$. Αρχικά μετράμε τα μηδενικά:

$$1983.000.000.000.000.000.000.000.000.000$$

ο αριθμός μας γίνεται:

$$1983 \cdot 10^{27}$$

Η τυποποιημένη μορφή απαιτεί το α να είναι μεγαλύτερο ή ίσο του 1 και μικρότερο του 10. Οπότε το 1983 πρέπει να γίνει 1,983 και ο αριθμός μας να γίνει

$$1,983 \cdot 10^{30}$$

σε τυποποιημένη μορφή μεγάλων αριθμών. Η Python υποστηρίζει την τυποποιημένη μορφή μεγάλων αριθμών με τη χρήση της εντολής format. Δοκίμασε:

```
>>> x = 1983000000000000000000000000000
>>> print(x)
1983000000000000000000000000000
>>> print('{:.3e}'.format(x))
1.983e+30
```

Η γενική μορφή είναι να χρησιμοποιείς `<'{:.Ne}'.format(x)>`όπου N είναι το πλήθος των δεκαδικών ψηφίων που θες να εμφανίζονται. Ετσι

```
>>> print('{:.2e}'.format(3140000000000000000))
3.14e+18
>>> print('{:.2e}'.format(234000000000000000))
2.34e+17
```

Η Python καταλαβαίνει την τυποποιημένη μορφή π.χ.:

```
>>> x = 3.14e+30
>>> print(x)
3.14e+30
>>> print('{:.0f}'.format(x))
3139999999999999741556248543232
```

Το ':.0f' σημαίνει πως ο αριθμός θα πρέπει να γραφεί σαν δεκαδικός (float) με μηδεν δεκαδικά ψηφία

Ασκηση 1.2.1 *(Στο βιβλίο βρίσκεται στη Σελ. 63) Να γράψεις τους παρακάτω αριθμούς στην τυποποιημένη μορφή: (α) 583.000 (β) 4.300.000 (γ) 7.960.000 (δ) 3.420.000.000 (ε) 4.800 (στ) 7.310 (ζ) 281.900 (η) 518.000.000 (θ) 131.000 (ι) 675.000.*

```
>>> print('{:.2e}'.format(583000))
5.83e+05
>>> print('{:.1e}'.format(4300000))
4.3e+06
>>> print('{:.2e}'.format(7960000))
7.96e+06
>>> print('{:.2e}'.format(3420000000))
3.42e+09
>>> print('{:.1e}'.format(4800))
4.8e+03
>>> print('{:.2e}'.format(7310))
7.31e+03
>>> print('{:.3e}'.format(281900))
2.819e+05
>>> print('{:.2e}'.format(518000000))
5.18e+08
>>> print('{:.2e}'.format(131000))
1.31e+05
>>> print('{:.2e}'.format(675000))
6.75e+05
```

Ασκηση 1.2.2 *(Στο βιβλίο βρίσκεται στη Σελ. 63) Να γράψεις τη δεκαδική μορφή των αριθμών: (α)* $3,1 \cdot 106$ *(β)* $4,820 \cdot 105$ *(γ)* $3,25 \cdot 104$ *(δ)* $7,4 \cdot 103$ *(ε)* $9,2 \cdot 102$*.*

```
>>> print(4.820 * 10**5)
482000.0
>>> print(3.25 * 10**4)
```

```
32500.0
>>> print(7.4 * 10**3)
7400.0
>>> print(9.2 * 10**2)
919.9999999999999
```

Ειδικά για το τελευταίο παρατηρούμε ότι έχουμε μια προσέγγιση και αντί για 920 που είναι το σωστό αποτέλεσμα προκύπτει 919.9999999999999. Η Python κάνει προσεγγίσεις όταν χρειάζεται να κάνει πράξεις ή όταν ο αριθμός δεν μπορεί να αναπαρασταθεί στα όρια του υπολογιστή. Το 920 δεν ανήκει στην δεύτερη κατηγορία οπότε το λάθος προκύπτει από την πράξη (πολλαπλασιαμός με το 100). Στην πραγματικότητα ο πολλαπλασιασμός αυτός δεν είναι αναγκαίος μπορείς να γράψεις 9.2e2 αντί για 9.2*10**2 και η Python θα καταλάβει τον αριθμό που θέλεις:

```
>>> 9.2e2
920.0
```

Ασκηση 1.2.3 *Να εκφραστεί το μήκος των 2.754,389 m, σε όλες τις υποδιαιρέσεις του m.*

Οι υποδιαιρέσεις του μέτρου (m) είναι τα δεκατόμετρα (dm), τα εκατοστόμετρα (cm) και τα χιλιοστόμετρα (mm). Για να εκφραστεί το μήκος σε κάθε μία από τις υποδιαιρέσεις θα πρέπει να το πολλαπλασιάζουμε με το 10. Δοκίμασε:

```
>>> x = 2754.389
>>> x = 10*x
>>> print(x)
27543.89
>>> x = 10*x
>>> print(x)
275438.9
>>> x = 10*x
>>> print(x)
2754389.0
```

Οπότε έχουμε 27543.89 δεκατόμετρα, 275438.9 εκατοστόμετρα και 2754389 χιλιοστόμετρα. Μπορούμε να μετατρέψουμε τα παραπάνω σε πρόγραμμα ως εξής:

```
x = float(input('Μήκος σε μέτρα:'))
for i in range(3):
    x = 10*x
    print(x)
```

Που δίνει το εξής αποτέλεσμα:

```
   Μήκος
 σε μέτρα:2754.389
27543.89
275438.9
2754389.0
```

Ακόμη καλύτερα θα ήταν να κάνουμε:

```
x = float(input('Μήκος σε μέτρα:'))
monades = ['dm','cm','mm']
for i in range(3):
    x = 10*x
    print(x,monades[i])
```

Αν ξέχασες το float πριν το input θα δεις ότι ο πολλαπλασιασμός μεταξύ ακεραίου και αλφαρηθμιτικού δουλεύει, και αντιγράφει το ίδιο αλφαριθμητικό πολλές φορές, δοκίμασε:

```
>>> x = input('Μήκος σε μέτρα:')Μήκος
 σε μέτρα:2754.389
>>> 10*x
'2754.3892754.3892754.3892754.3892754.3892754.3892754.3892754.3892754.3892754.389'
```

Ασκηση 1.2.4 *Η επιφάνεια ενός κύβου έχει εμβαδόν 96 cm^2. Να βρεθεί ο όγκος του.*

Η λύση του βιβλίου είναι:

Επειδή ο κύβος έχει 6 έδρες, η κάθε έδρα του θα έχει εμβαδόν $96\text{cm}^2 : 6 = 16\text{cm}^2$.

Αλλά είναι $16\text{cm}^2 = 4\text{cm} \cdot 4\text{cm} = (4\text{cm})^2$, άρα, η ακμή του κύβου είναι 4cm.

Επομένως, ο όγκος του κύβου είναι: $(4\text{cm})^3 = 4\text{cm} \cdot 4\text{cm} \cdot 4\text{cm} = 64\text{cm}^3$

Σε Python η λύση προχωράει ως εξής:

```
>>> epifaneia = 96
>>> epifaneiaPlevras = epifaneia/6
16
```

Εμείς ξέρουμε ότι $4^2 = 16$ και μπορούμε να το θυμηθούμε. Ο υπολογιστής όμως δεν το ξέρει και πρέπει να έχει μια συνάρτηση για να υπολογίσει τον αριθμό που αν τον υψώσουμε στο τετράγωνο θα κάνει 16. Ευτυχώς, αυτή η συνάρτηση υπάρχει λέγεται sqrt και βρίσκεται στη βιβλιοθήκη math. Οπότε μπορούμε να τη χρησιμοποιήσουμε ως εξής:

```
>>> import math
>>> math.sqrt(16)
4.0
```

και από εκεί μπορούμε να υπολογίσουμε τον όγκο υψώνοντας στην τρίτη.

Συνολικά το πρόγραμμά μας γίνεται:

```
import math
epifaneia = 96
epifaneiaPlevras = epifaneia/6
akmi = math.sqrt(epifaneiaPlevras)
ogkos = akmi**3
print(ogkos)
```

που δίνει το αποτέλεσμα:

```
64.0
```

Το ίδιο πρόγραμμα μπορούμε να το χρησιμοποιήσουμε και για να βρούμε τον όγκο ενός κύβου με επιφάνεια 54. Αφού έχουμε $54 : 6 = 9$ άρα η ακμή του κύβου είναι 3 και ο όγκος του 27. Το παρακάτω πρόγραμμα δίνει το σωστό αποτέλεσμα:

```
import math
epifaneia = 54
epifaneiaPlevras = epifaneia/6
akmi = math.sqrt(epifaneiaPlevras)
ogkos = akmi**3
print(ogkos)
```

Μπορούμε να γράψουμε ένα πρόγραμμα που να του δίνουμε την επιφάνεια ενός κύβου και να μας βρίσκει τον όγκο του το πρόγραμμα αυτό είναι το εξής:

```
import math
epifaneia = int(input('Δώσε επιφάνεια κύβου: '))
epifaneiaPlevras = epifaneia/6
akmi = math.sqrt(epifaneiaPlevras)
ogkos = akmi**3
print('Ο όγκος του κύβου είναι: ', ogkos)
```

Τέλος, μπορούμε να γράψουμε μια συνάρτηση που να υπολογίζει τον όγκο ενός κύβου από την επιφάνειά του:

```
import math
def ogkosapoepifaneia(epifaneia):
    epifaneiaPlevras = epifaneia/6
    akmi = math.sqrt(epifaneiaPlevras)
    ogkos = akmi**3
    return(ogkos)
epifaneia = int(input('Δώσε επιφάνεια κύβου: '))
print('Ο όγκος του κύβου είναι: ', ogkosapoepifaneia(epifaneia))
```

Ασκηση 1.2.5 *(Στο βιβλίο βρίσκεται στη Σελ. 66) Μια αμαξοστοιχία διανύει την απόσταση Αθήνας - Πύργου σε 4 ώρες και 57 λεπτά. Αν η αμαξοστοιχία ξεκινά από την Αθήνα στις 9:10 π.μ. το πρωί, ποια ώρα θα φτάσει στον Πύργο;*

Πώς μπορούμε στην Python να κάνουμε πράξεις με τις ώρες; Υπάρχουν δύο τρόποι: α) Να τα υπολογίσουμε με όσα γνωρίζουμε: Ετσι αν έχουμε θέλουμε να προσθέσουμε 9h και 10m με 4h και 57m οπότε ξεκινάμε από τα λεπτά και βρίσκουμε τις ώρες μετά. Ενας τρόπος είναι λοιπόν ο εξής:

```
anaxWra = 9
anaxLepta = 10
diarkeiaWra = 4
diarkeiaLepta = 57
athroismaLepta = anaxLepta + diarkeiaLepta
telikaLepta = athroismaLepta % 60
telikiWra = anaxWra + diarkeiaWra + athroismaLepta // 60
if telikiWra > 12:
    print(str(telikiWra ) + ':' + str(telikaLepta – 12)+ ' μμ..')
else:
    print(str(telikiWra)+':'+str(telikaLepta) + 'πμ..')
```

Οπως καταλαβαίνεις το θέμα δεν είναι να υπολογίσεις μια φορά το αποτέλεσμα αλλά να φτιάξεις ένα πρόγραμμα που να υπολογίζει το αποτέλεσμα αν ο χρήστης δίνει την ώρα αναχώρησης και τη διάρκεια του ταξιδιού. Αυτό μπορεί να γίνει με την input και τη split.

```
anax = input('Αναχώρηση μορφή( ωωλλ:)>')
diarkeia = input('Διάρκεια μορφή( ωωλλ:)>')
anaxWra = int(anax.split(':')[0])
anaxLepta = int(anax.split(':')[1])
diarkeiaWra = int(diarkeia.split(':')[0])
diarkeiaLepta = int(diarkeia.split(':')[1])
athroismaLepta = anaxLepta + diarkeiaLepta
telikaLepta = athroismaLepta % 60
telikiWra = anaxWra + diarkeiaWra + athroismaLepta // 60
if telikiWra > 12:
    print(str(telikiWra ) + ':' + str(telikaLepta –12)+ ' μμ..')
else:
    print(str(telikiWra)+':'+str(telikaLepta) + 'πμ..')
```

Το παραπάνω πρόγραμμα μας δίνει τη σωστή απάντηση σε πολλές περιπτώσεις. Οχι όμως σε όλες. Δείτε:

```
   Αναχώρηση
 μορφή( ωωλλ:)>19:30Διάρκεια
 μορφή( ωωλλ:)>5:40
25:-2 μμ..
```

Θα πρέπει λοιπόν να φτιάξουμε το άθροισμα της ώρας να μην ξεπερνάει το 24.

```
anax = input('Αναχώρηση μορφή( ωωλλ:)>')
diarkeia = input('Διάρκεια μορφή( ωωλλ:)>')
anaxWra = int(anax.split(':')[0])
anaxLepta = int(anax.split(':')[1])
```

```
diarkeiaWra = int(diarkeia.split(':')[0])
diarkeiaLepta = int(diarkeia.split(':')[1])
athroismaLepta = anaxLepta + diarkeiaLepta
telikaLepta = athroismaLepta % 60
telikiWra = (anaxWra + diarkeiaWra + athroismaLepta // 60) % 24
if telikiWra > 12:
    print(str(telikiWra ) + ':' + str(telikaLepta -12)+ ' μμ..')
else:
    print(str(telikiWra)+':'+str(telikaLepta) + 'πμ..')
```

Τότε παίρνουμε το σωστό αποτέλεσμα:

```
   Αναχώρηση
 μορφή( ωωλλ:)>19:30Διάρκεια
 μορφή( ωωλλ:)>5:40πμ
1:10..
```

β) Ο δεύτερος τρόπος είναι να χρησιμοποιήσουμε τη βιβλιοθήκη datetime η οποία χρειάζεται κάποιους ειδικούς χειρισμούς.

```
import datetime

anax = input('Αναχώρηση μορφή( ωωλλ:)>')
diarkeia = input('Διάρκεια μορφή( ωωλλ:)>')
anaxWra = int(anax.split(':')[0])
anaxLepta = int(anax.split(':')[1])
diarkeiaWra = int(diarkeia.split(':')[0])
diarkeiaLepta = int(diarkeia.split(':')[1])
t = datetime.time(9,10)
t = datetime.datetime.combine(datetime.date(2020,5,25),t)
d = datetime.timedelta(hours = 4, minutes=57)
print((t+d).strftime('%H:%M'))
```

Μπορούμε να μικρύνουμε λίγο ακόμη το πρόγραμμα ως εξής:

```
import datetime

anax = input('Αναχώρηση μορφή( ωωλλ:)>')
diarkeia = input('Διάρκεια μορφή( ωωλλ:)>')
t = datetime.time(int(anax.split(':')[0]),int(anax.split(':')[1]))
t = datetime.datetime.combine(datetime.date(2020,5,25),t)
d = datetime.timedelta(hours = int(diarkeia.split(':')[0]),
    minutes=int(diarkeia.split(':')[1]))
print((t+d).strftime('%H:%M'))
```

Το αποτέλεσμα του προγράμματος με τα δεδομένα του προβλήματος είναι:

```
   Αναχώρηση
 μορφή( ωωλλ:)>9:10Διάρκεια
 μορφή( ωωλλ:)>4:57
14:07
```

Αν δώσουμε διαφορετικά δεδομένα παίρνουμε σωστές απαντήσεις:

```
   Αναχώρηση
 μορφή( ωωλλ:)>14:10Διάρκεια
 μορφή( ωωλλ:)>5:30
19:40
```

και για τις ειδικές περιπτώσεις:

```
   Αναχώρηση
 μορφή( ωωλλ:)>19:30Διάρκεια
 μορφή( ωωλλ:)>5:40
01:10
```

Ασκηση 1.2.6 *(Στο βιβλίο βρίσκεται στη Σελ. 67) Να βρεθεί η περίμετρος του σχήματος: (α) σε μέτρα, (β) σε εκατοστά και (γ) σε χιλιόμετρα.*

Λύση (α) Η περίμετρος σε μέτρα είναι ίση με το άθροισμα των μηκών των πλευρών του, δηλαδή:

```
>>> 26.6+23.5+22.17+38.53
111.8
```

Για να το μετατρέψουμε σε εκατοστά θα πολλαπλασιάσουμε με το 100

```
>>> 111.8*100
11180
```

Για να το μετατρέψουμε σε χιλιόμετρα θα διαιρέσουμε με το 1000:

```
>>> 111.8/1000
0,1118
```

Ασκηση 1.2.7 *(Στο βιβλίο βρίσκεται στη Σελ. 67) Μια δεξαμενή νερού τρύπησε και χύνονται 2 σταγόνες κάθε δευτερόλεπτο. Αν οι 25 σταγόνες έχουν μάζα 1,5 g, να βρεθεί η μάζα του νερού που χάνεται κάθε ώρα, σε κιλά.*

Κάθε δευτερόλεπτο χάνονται 2 σταγόνες νερού επομένως κάθε ώρα χάνονται:

```
>>> 2* 60 * 60
7200
```

Αυτές τις 7200 τις διαιρούμε με το 25 και τις πολλαπλασιάζουμε με τη μάζα των 25 σταγόνων και έχουμε:

```
>>> 7200/25*1.5
432
```

Αυτή η μάζα είναι σε γραμμάρια για να βρούμε σε κιλά διαιρούμε με το 1000.

```
>>> 432/1000
0.432
```

Μπορούμε να γράψουμε και ένα πρόγραμμα για να υπολογίζει τη μάζα του νερού που χάνεται σε μια πιο γενική περίπτωση. Ας πούμε ότι τα δεδομένα μας θα είναι πόσες σταγόνες χάνονται το δευτερόλεπτο και η μάζα της σταγόνας. Να γραφεί ένα πρόγραμμα που όταν δίνεται η μάζα μιας σταγόνας και το πλήθος των σταγόνων που χύνεται κάθε δευτερόλεπτο, να υπολογίζει τη μάζα του νερού που χύνεται κάθε ώρα σε κιλά. Το πρόγραμμα θα είναι το εξής:

```
plithos = int(input('Σταγόνες το δευτερόλεπτο:'))
maza = float(input('Μάζα κάθε σταγόνας:'))
grammaria = plithos*maza*60*60
kila = grammaria / 1000
print('Χάνονται ',kila, ' κιλά.')
```

Ενα παράδειγμα εκτέλεσης του παραπάνω προγράμματος είναι:

```
   Σταγόνες
 το δευτερόλεπτο:2Μάζα
 κάθε σταγόνας:0.05Χάνονται
  0.36  κιλά.
```

Ασκηση 1.2.8 *(Στο βιβλίο βρίσκεται στη Σελ. 67) Να συμπληρώσεις τα κενά: (α) 23 dm = ... cm, (β) 3,1 m = ... Km, (γ) 45,83 cm = ... m, (δ) 67,2 Km = ... mm, (ε) 95,5 mm = ... cm.*

Μπορούμε να φτιάξουμε ένα πρόγραμμα που να λύνει αυτή την άσκηση. Οι μονάδες μέτρησης που μας ενδιαφέρουν είναι:

mm,cm,dm,m,Km

Αν υποθέσουμε ότι ξέρουμε τα χιλιοστά τότε οι υπόλοιπες μονάδες είναι 10mm, 100mm, 1000mm, και 1000000mm. Μπορούμε λοιπόν να φτιάξουμε έναν μετατροπέα από χιλιοστά σε οποιδήποτε άλλη μονάδα:

```
def frommmto(num,mon):
    if mon == 'mm':
        return(num)
    elif mon=='cm':
        return(num/10)
    elif mon=='dm':
        return(num/100)
    elif mon=='m':
        return(num/1000)
    elif mon=='Km':
        return(num/1000000)
    else:
        return(None)
```

Αντίστοιχα μπορούμε να φτιάξουμε έναν μετατροπέα από οποιαδήποτε μονάδα σε χιλιοστά:

```
def tommfrom(num,mon):
    if mon == 'mm':
        return(num)
    elif mon=='cm':
        return(num*10)
    elif mon=='dm':
        return(num*100)
    elif mon=='m':
        return(num*1000)
    elif mon=='Km':
        return(num*1000000)
    else:
        return(None)
```

Συνδυάζοντας αυτούς τους δύο μετατροπείς θα έχουμε έναν μετατροπέα από οποιαδήποτε μονάδα σε οποιαδήποτε:

```
changeUnit(num,arx_mon,tel_mon):
    mms = tommfrom(num,arx_mon)
    if mms is not None:
        result = frommmto(mms,tel_mon)
        return(result)
    else:
        return(None)
```

Και ένα πρόγραμμα για δοκιμή είναι:

```
changeUnit(23,'dm','cm')
changeUnit(3.1,'m','Km')
changeUnit(45.83,'cm','m')
changeUnit(67.2,'Km','mm')
changeUnit(95.5,'mm','cm')
```

και το αποτέλεσμα είναι:

```
230.0
0.0031
0.45829999999999993
67200000.0
9.55
```

Ασκηση 1.2.9 *Ενα ορθογώνιο παραλληλεπίπεδο έχει ακμές μήκους α = 3,1 m, β = 4,2 m και γ= 2,3 m. Να υπολογίσεις το μήκος των ακμών του σε mm και να το γράψεις σε τυποποιημένη μορφή.*

Χρησιμοποιώντας την ίδια συνάρτηση όπως και παραπάνω μπορούμε να γράψουμε:

```
>>> mikos = changeUnit(3.1+4.2+2.3,'m','mm')
>>> mikos
9600.000000000002
>>> '{:.2e}'.format(mikos)
'9.60e+03'
```

Ασκηση 1.2.10 *(Ασκηση 3 του βιβλίου, Σελ. 67) Γράψε τα παρακάτω μήκη σε αύξουσα σειρά: 986 m, 0,023 Km, 456 cm, 678 dm.*

Μπορούμε να χρησιμοποιήσουμε τη συνάρτηση tommfrom από παραπάνω ώστε να γίνουν πρώτα όλα ίδιες μονάδες για να συγκριθούν.

```
a = [(986,'m'),(0.023,'Km'),(456,'cm'),(678,'dm')]
print(sorted(a,key=lambda x:tommfrom(x[0],x[1])))
```

Η έκφραση lambda x:tommfrom(x[0],x[1]) είναι μια σύντομη μορφή του εξής:

```
def onoma_sinartisis(x):
    return(tommfrom(x[0],x[1]))
```

Το όνομα της συνάρτησης δεν παίζει ρόλο. Αυτό που συμβαίνει είναι ότι η Python ταξινομεί τα αντικείμενα της λίστας με βάση ένα κλειδί, το κλειδί είναι να μετατρέπει το αντικείμενο σε χιλιοστά χρησιμοποιώντας το πρώτο κομμάτι του αντικειμένου σαν αριθμό και το δεύτερο σαν μονάδα.

```
[(456, 'cm'), (0.023, 'Km'), (678, 'dm'), (986, 'm')]
```

Ασκηση 1.2.11 *(Ασκηση 4 του βιβλίου, Σελ. 67) Ενα ορθογώνιο παραλληλόγραμμο έχει διαστάσεις πλευρών α=23 cm και β=45 cm. Να βρεις το εμβαδόν του, σε cm^2 και σε mm^2.*

Μπορούμε να τη λύσουμε στο REPL ως εξής:

```
>>> emvadosecm2 = 23*45
>>> emvadosecm2
1035
>>> emvadosecm2*100
103500
```

Ασκηση 1.2.12 *(Ασκηση 5 του βιβλίου, Σελ. 67) Συμπλήρωσε τα κενά: (α) 56 Km^2=.....m^2, (β) 0,987 στρέμματα=.....m^2, (γ) 350 στρέμματα=.....m^2.*

Στο REPL:

```
>>> 56 * 1000000
56000000
>>> 0.987 * 1000
987
>>> 350 * 1000
350000
```

(α) 56 Km^2=56000000m^2, (β) 0,987 στρέμματα=987m^2, (γ) 350 στρέμματα=350000m^2.

Ασκηση 1.2.13 *(Ασκηση 6 του βιβλίου, Σελ. 67) Ενα οικόπεδο έχει σχήμα τετραγώνου με πλευρά 210 m. Να υπολογίσεις το εμβαδόν του σε m^2 και σε στρέμματα.*

```
>>> emvado = 210*210
>>> emvado
44100
>>> emvado / 1000
44.1
```

Ασκηση 1.2.14 *(Ασκηση 7 του βιβλίου, Σελ. 67) Μια αυλή, σχήματος ορθογωνίου παραλληλογράμμου, έχει διαστάσεις 5 m και 7,2 m. Θέλουμε να τη στρώσουμε, με τετράγωνες πλάκες, πλευράς 40 cm.Πόσες πλάκες θα χρειαστούμε;*

```
>>> emvadoorth = 5*7.2
>>> emvadoplakas = 0,4*0,4
>>> emvadoorth/emvadoplakas
224.99999999999994
```

Δηλαδή περίπου 225 πλάκες.

Ασκηση 1.2.15 *(Ασκηση 8 του βιβλίου, Σελ. 67) Ο όγκος ενός στερεού είναι 15 dm^3 29 cm^3. Να βρεις τον όγκο του στερεού σε cm^3, m^3 και mm^3.*

```
>>> 15 * 100** 3 + 29 * 10 **3
15029000
>>> 15029000 / 1000
15029
>>> 15029/ 100**3
0.01529
```

Ασκηση 1.2.16 *(Ασκηση 9 του βιβλίου, Σελ. 67) Ενας οινοπαραγωγός έχει αποθηκεύσει το κρασί του σε 3 ίσες δεξαμενές, σχήματος ορθογωνίου παραλληλεπιπέδου, με διαστάσεις 3 m, 2 m και 5 m. Αν πουλήσει το κρασί του προς 4 ευρώ το λίτρο, πόσα χρήματα θα εισπράξει;*

```
>>> 3 * 3 * 2 *5
90
>>> 90 * 10 **3
90000
>>> 90000 * 4
360000
```

Ασκηση 1.2.17 *(Ασκηση 10 του βιβλίου, Σελ. 67) Να υπολογίσεις τον χρόνο, από τις 8h 10min το πρωί, ως τις 5h 20min το απόγευμα.*

```
>>> 20 - 10
10
>>> 17 - 8
9
```

Αρα 9h και 10min. Η Python μπορεί να κάνει την πράξη χρησιμοποιώντας το module datetime.

```
>>> import datetime
>>> t1str = '2015-08-12 08:10'
>>> t1 = datetime.datetime.strptime(t1str, '%Y-%m-%d %H:%M')
>>> t2str = '2015-08-12 17:20'
>>> t2 = datetime.datetime.strptime(t2str, '%Y-%m-%d %H:%M')
>>> print(t2-t1)
9:10:00
```

Ασκηση 1.2.18 *(Ασκηση 11 του βιβλίου, Σελ. 67) Συμπλήρωσε τα κενά: (α) 4h 52min=.....min=.....s, (β) 3h 12min=.....min=.....s, (γ) 5h 20min 30s=.....min=.....s, (δ) 56min 45s=.....min=.....s*

Γι' αυτή την άσκηση θα φτιάξουμε ένα πρόγραμμα ώστε να ζητάμε από τον χρήστη ώρες λεπτά και δευτερόλεπτα και να μας μετατρέπει το χρόνο σε λεπτά και σε δευτερόλεπτα.

```
def xronosseld(wres,lepta,deutera):
    xronossedeutera = wres * 60 * 60 + lepta * 60 + deutera
    xronosselepta = xronossedeutera / 60
    print(xronosselepta,' min=',xronossedeutera,' s')

while True:
    wres = int(input('Ώρες:'))
    lepta = int(input('Λεπτα:'))
    deutera = int(input('Δευτερόλεπτα:'))
    xronosseld(wres,lepta,deutera)
```

Οταν εκτελέσουμε το παραπάνω πρόγραμμα θα έχουμε το εξής αποτέλεσμα:

```
Ώρες:4
Λεπτα:52
Δευτερόλεπτα:0
292.0  min= 17520  s
Ώρες:3
Λεπτα:12
Δευτερόλεπτα:0
192.0  min= 11520  s
Ώρες:5
Λεπτα:20
Δευτερόλεπτα:30
320.5  min= 19230  s
Ώρες:0
Λεπτα:56
Δευτερόλεπτα:45
56.75  min= 3405  s
Ώρες:^C
```

Επειδή έχουμε βάλει **while** True η επανάληψη γίνεται για πάντα. Για να βγούμε από το πρόγραμμα θα πατήσουμε το πλήκτρο Ctrl και το πλήκτρο C (Ĉ).

Ασκηση 1.2.19 *(Ασκηση 12 του βιβλίου, Σελ. 67) Να υπολογίσεις: (α) το $\frac{1}{10}$ της ώρας, (β) το $\frac{1}{5}$ της ώρας, (γ) το $\frac{1}{6}$ της ώρας.*

```
>>> 1/10*60
6
>>> 1/5*60
12
>>> 1/6*60
10
```

Αρα είναι 6 λεπτά, 12 λεπτά και 10 λεπτά.

Ασκηση 1.2.20 *(Ασκηση 13 του βιβλίου, Σελ. 68) Διαθέτουμε σταθμά των 50 g, 500 g και δύο σταθμά του 1 Kg. Πώς θα ζυγίσουμε ένα βάρος (α) 3 Kg και 600g και (β) 2 Kg και 450 g.*

Θα προσπαθήσουμε να γράψουμε ένα πρόγραμμα που να λύνει το πρόβλημα. Τα δεδομένα είναι το βάρος σε κιλά και γραμμάρια και μια λίστα με τα σταθμά. Στη συνέχεια μετατρέπουμε το βάρος που δίνεται σε κιλά και γραμμάρια αποκλειστικά σε γραμμάρια. Ξεκινάμε από το μεγαλύτερο από τα σταθμά και βρίσκουμε πόσες φορές χωράει στο βάρος. Συνεχίζουμε την ίδια διαδικασία με το υπόλοιπο του βάρους και τα υπόλοιπα σταθμά.

```
def zygisi(barosKila,barosg,stathmaseg):
    baros = barosKila * 1000 + barosg
    stathmos = 0
```

```
    res = [0]*len(stathmaseg)
    while baros>=stathmaseg[-1]:
        if baros >= stathmaseg[stathmos]:
            res[stathmos]=baros//stathmaseg[stathmos]
            baros = baros % stathmaseg[stathmos]
        stathmos += 1
    return(res)

print(zygisi(3,600,[1000,500,50]))
print(zygisi(2,450,[1000,500,50]))
```

Το οποίο δίνει το σωστό αποτέλεσμα:

```
[3, 1, 2]
[2, 0, 9]
```

Αυτό το πρόγραμμα έχει ένα μειονέκτημα. Η σωστή του λειτουργία βασίζεται στο να δώσει ο χρήστης τα σταθμά ξεκινώντας από το μεγαλύτερο προς το μικρότερο. Δες τι θα συμβεί αν εκτελέσουμε:

```
print(zygisi(3,600,[1000,50,500]))
print(zygisi(2,450,[50,500,100]))
```

```
[3, 12, 0]
[49, 0, 0]
```

Μπορείς να το αποφύγεις με κάποιες αλλαγές στον κώδικα. Η βασική αλλαγή είναι πως αντί να ελέγχεις το stathmaseg[stathmos] βασιζόμενος ότι αυτό θα είναι το μεγαλύτερο να ελέγχεις με το max(stathmaseg). Θα πρέπει όμως μετά να βάλεις το αποτέλεσμα στη σωστή θέση. Η θέση αυτή είναι stathmaseg.index(max(stathmaseg)). Ομως θα πρέπει να διαγράφουμε κάθε φορά το μεγαλύτερο στοιχείο ώστε η max να μην το ξαναεπιστρέψει. Για αυτό θα χρειαστεί να κάνουμε ένα αντίγραφο της λίστας πάνω στο οποίο δουλεύουμε χωρίς να πειράξουμε το stathmaseg. Συνολικά:

```
def zygisi(barosKila,barosg,stathmaseg):
    baros = barosKila * 1000 + barosg
    stathmos = 0
    res = [0]*len(stathmaseg)
    s = list(stathmaseg)
    while baros>=min(s):
        if baros >= max(s):
            res[stathmaseg.index(max(s))]=baros//max(s)
            baros = baros % max(s)
        s.pop(s.index(max(s)))
        if not s:
            break
    return(res)

print(zygisi(3,600,[1000,50,500]))
```

```
print(zygisi(2,450,[50,500,1000]))
```

Με αυτόν τον τρόπο έχεις το σωστό αποτέλεσμα που είναι:

```
[3, 2, 1]
[9, 0, 2]
```