$Αλγεβρα$
Μαθηματικά Γυμνασίου με Python

Δημήτρης Νικολός

24 Μαΐου 2020

Κεφάλαιο 1

Τι θα χρησιμοποιήσουμε;

1.1 Η γλώσσα προγραμματισμού Python

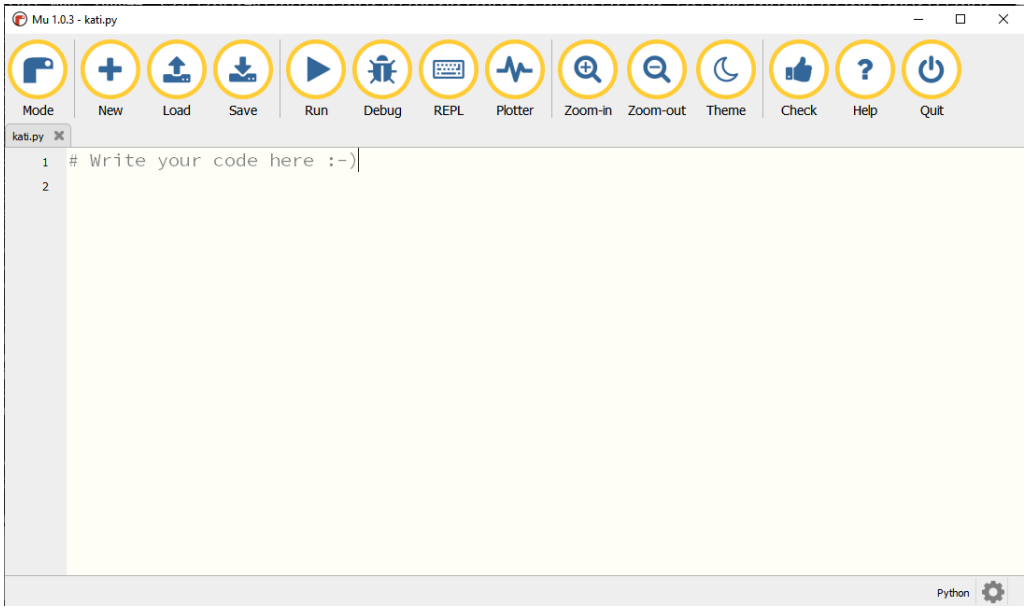
Σε αυτές τις σημειώσεις θα χρησιμοποιήσουμε τη γλώσσα προγραμματισμού Python και μάλιστα την έκδοση 3. Υπάρχει και Python 2 αλλά υπάρχουν σχέδια για την αντικατάστασή της από την Python 3. Για να εγκαταστήσεις την Python 3 θα πρέπει να την κατεβάσεις από το επίσημο site της Python www.python.org. Κατεβάστε την πιο πρόσφατη έκδοση που σας προτείνει θα είναι κάτι σαν 3.8.2 ή κάτι

1.2 Ο επεξεργαστής προγραμμάτων Mu

Μπορείς να γράψεις Python σε οποιοδήποτε πρόγραμμα υποστηρίζει απλό κείμενο, ακόμη και στο Σημειωματάριο, όμως σε αυτές τις σημειώσεις χρησιμοποιούμε τον επεξεργαστή Python, Mu Editor ή πιο απλά Mu που μπορείς να τον κατεβάσεις από τη σελίδα codewith.mu. Μόλις το ανοίξεις θα δεις την εικόνα 1.2.

Μπορείς να πατήσεις την εκτέλεση (κουμπί Run) και τότε θα δεις ότι το παράθυρο χωρίζεται σε δύο τμήματα (Εικόνα 1.2). Αν θες να δοκιμάσεις ένα ολόκληρο πρόγραμμα μπορείς να το πληκτρολογήσεις στο βασικό παράθυρο (τώρα γράφει ‘#Write your code here‘). Ενώ αν θες να δοκιμάσεις κάποια εντολή τότε μπορείς να την πληκτρολογήσεις στο κάτω παράθυρο (τώρα γράφει >>>). Το κάτω παράθυρο ονομάζεται REPL, από τα αρχικά των λέξεων Read, Eval, Print, Loop δηλαδή Διάβασε, Εκτέλεσε (την εντολή-/έκφραση), Τύπωσε, Επανάλαβε. Το REPL θα διαβάσει την εντολή, θα την εκτελέσει και θα μας δώσει το αποτέλεσμα.

Από εδώ και πέρα όταν βλέπετε στις σημειώσεις τα τρία σύμβολα “μεγαλύτερο από” (>>>) θα πληκτρολογείτε τις αντίστοιχες εντολές στο κάτω



Σχήμα 1.1: Mu: Ενας επεξεργαστής προγραμμάτων Python

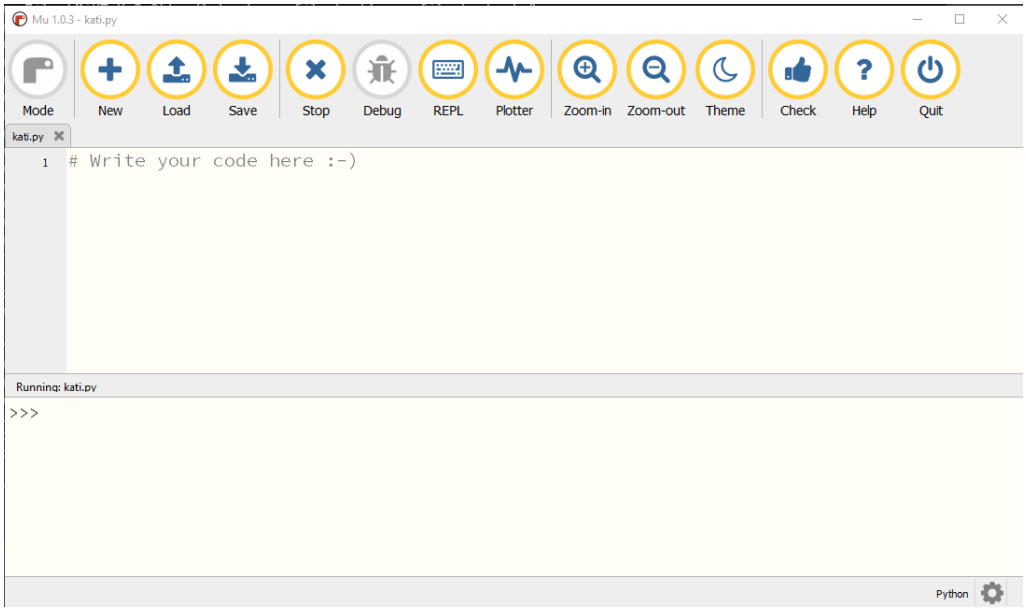
παράθυρο (REPL). Τα μεγαλύτερα προγράμματα που δεν θα έχουν αυτό το σύμβολο θα τα πληκτρολογείτε στο πάνω παράθυρο.

> **Συμβουλή:** Αν χρησιμοποιείτε την ηλεκτρονική έκδοση αυτών των σημειώσεων, θυμηθείτε να πληκτρολογείτε τις εντολές και να μην τις κάνετε αντιγραφή επικόλληση.

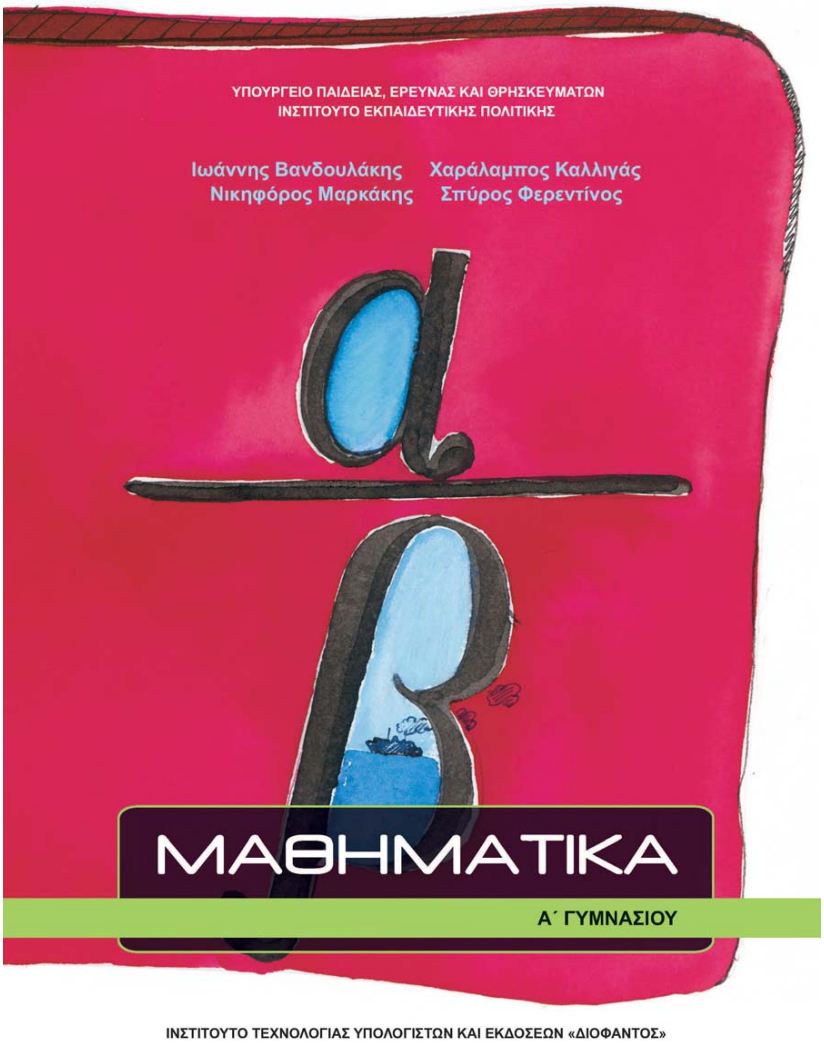
Στην αρχή θα δοκιμάσεις κάποια πράγματα στο κάτω παράθυρο, όμως μην ανησυχείς σύντομα θα γράφεις τα δικά σου προγράμματα στο πάνω παράθυρο.

1.3 Το βιβλίο μαθηματικών της Α΄ Γυμνασίου

Σε αυτές τις σημειώσεις οι περισσότερες ασκήσεις είναι από το βιβλίο Μαθηματικών της Α΄ Γυμνασίου των Βανδουλάκη, Καλλιγά, Μαρκάκη και Φερεντίνου (Εικόνα 1.3).



Σχήμα 1.2: Το πρόγραμμα Mu όταν εκτελείτε ένας κώδικας



Σχήμα 1.3: Το εξώφυλλο του βιβλίου των Μαθηματικών που θα χρησιμοποιήσουμε

Κεφάλαιο 2

Φυσικοί αριθμοί

2.1 Οι αριθμοί και η Python

Οι φυσικοί αριθμοί είναι οι αριθμοί από 0, 1, 2, 3, 4, 5, 6, … , 98, 99, 100, … , 1999, 2000, 2001, …

Η Python μπορεί να χειριστεί φυσικούς αριθμούς. Δοκιμάστε να γράψετε στο REPL έναν φυσικό αριθμό, θα δείτε ότι η Python θα τον επαναλάβει. Π.χ. δείτε τον αριθμό εκατόν είκοσι τρια (123).

```
>>> 123
123
```

Στην Python όμως θα πρέπει να ακολουθείς κάποιους επιπλέον κανόνες. Για παράδειγμα στους αριθμούς δεν πρέπει να βάζεις τελείες στις χιλιάδες όπως στο χαρτί. Αν το κάνεις στην καλύτερη περίπτωση θα προκύψει κάποιο λάθος, στην χειρότερη ο υπολογιστής θα καταλάβει διαφορετικό αριθμό από αυτόν που εννοείς. Δείτε το παρακάτω παράδειγμα στο REPL.

```
>>> 1.000.000
  File "<stdin>", line 1
    1.000.000
            ^
SyntaxError: invalid syntax
>>> 100.000
100.0
```

Σε αυτό το παράδειγμα, η Python δεν καταλαβαίνει καθόλου τον αριθμό 1.000.000 γραμμένο με τελείες ενώ μεταφράζει το 100.000 σε 100.0, που για την Python σημαίνει 100 (εκατό). Γι' αυτόν τον λόγο δεν βάζουμε καθόλου τελείες έτσι αν θέλουμε να γράψουμε το ένα εκατομμύριο θα γράψουμε 1000000.

```
>>> 1000000
1000000
```

2.2 Πρόσθεση, αφαίρεση και πολλαπλασιασμός φυσικών αριθμών

Μια γλώσσα προγραμματισμού μπορεί να εκτελέσει απλές πράξεις πολύ εύκολα. Στο βιβλίο των μαθηματικών σου μπορείς να βρεις πολλές ασκήσεις με πράξεις. Μπορείς να τις λύσεις με την Python.

Ασκηση 2.2.1 *(Στο βιβλίο βρίσκεται στη Σελ. 16) Να υπολογιστούν τα γινόμενα:*
(α) $35 \cdot 10$,
(β) $421 \cdot 100$,
(γ) $5 \cdot 1.000$,
(δ) $27 \cdot 10.000$

Η python μπορεί να κάνει αυτές τις πράξεις ως εξής:

```
>>> 35*10
350
>>> 421*100
42100
>>> 5*1000
5000
>>> 27*10000
270000
```

Ο τελεστής του πολλαπλασιασμού είναι το αστεράκι ***** (SHIFT+8) στο πληκτρολόγιο. Εναλλακτικά, μπορείτε να το βρείτε στο αριθμητικό πληκτρολόγιο.

Ασκηση 2.2.2 *(Στο βιβλίο βρίσκεται στη Σελ. 16) Να εκτελεστούν οι ακόλουθες πράξεις:*
(α) $89 \cdot 7 + 89 \cdot 3$
(β) $23 \cdot 49 + 77 \cdot 49$
(γ) $76 \cdot 13–76 \cdot 3$
(δ) $284 \cdot 99$

```
>>> 89*7+89*3
890
>>> 23*49+77*49
4900
>>> 76*13-76*3
760
>>> 284*99
28116
```

Στις παραπάνω περιπτώσεις η python εκτελεί πρώτα τους πολλαπλασιασμούς και μετά τις προσθέσεις/αφαιρέσεις δίνοντας έτσι το αποτέλεσμα που αναμένεται. Για παράδειγμα 897 + 893 = 623 + 267 = 890, που είναι το σωστό αποτέλεσμα.

Ασκηση 2.2.3 *(Στο βιβλίο βρίσκεται στη Σελ. 18) Υπολογίστε:*

(α) $157 + 33$
(β) $122 + 25 + 78$
(γ) $785 - 323$
(δ) $7.321 - 4.595$
(ε) $60 - (18 - 2)$
(στ) $52 - 11 - 9$
(ζ) $23 \cdot 10$
(η) $97 \cdot 100$
(θ) $879 \cdot 1.000$

Σε python τα παραπάνω υπολογίζονται ως εξής:

```
>>> 157+33
190
>>> 122+25+78
225
>>> 785-323
462
>>> 7321-4595
2726
>>> 60-(18-2)
44
>>> 52-11-9
32
>>> 23*10
230
>>> 97*100
9700
>>> 879*1000
879000
```

Οι παρενθέσεις (SHIFT+9 και SHIFT+0) αλλάζουν τη σειρά των πράξεων. Οι πράξεις που είναι μέσα στην παρένθεση εκτελούνται πρώτες. Γι' αυτό το λόγο 60-(18-2)=60-16=44.

Ασκηση 2.2.4 *(Στο βιβλίο βρίσκεται στη Σελ. 18) Σε ένα αρτοποιείο έφτιαξαν μία μέρα 120 κιλά άσπρο ψωμί, 135 κιλά χωριάτικο, 25 κιλά σικάλεως και 38 κιλά πολύσπορο. Πουλήθηκαν 107 κιλά άσπρο ψωμί, 112 κιλά χωριάτικο, 19 κιλά σικάλεως και 23 κιλά πολύσπορο. Πόσα κιλά ψωμί έμειναν απούλητα;*

Με τις γνώσεις που έχουμε θα πρέπει να μετατρέψουμε το παραπάνω πρόβλημα σε μια αριθμητική παράσταση ώστε η python να μπορεί να την υπολογίσει, στη συγκεκριμένη περίπτωση η σωστή παράσταση είναι

$$(120 - 107) + (135 - 112) + (25 - 19) + (38 - 23)$$

```
>>> (120-107)+(135-112)+(25-19)+(38-23)
57
```

και η απάντηση είναι 57 κιλά ψωμί.

2.3 Δυνάμεις φυσικών αριθμών

Ο τελεστής της python για τις δυνάμεις είναι ο ** (δυο φορές το αστεράκι). Δηλαδή, αν θέλουμε να υπολογίσουμε το 10^2 θα γράψουμε 10**2, με όμοιο τρόπο μπορούμε να υπολογίσουμε και τις υπόλοιπες δυνάμεις. Δοκίμασε τα παρακάτω στο REPL.

```
>>> 10**2
100
>>> 10**3
1000
>>> 10**4
10000
>>> 10**5
100000
>>> 10**6
1000000
```

Στη προτεραιότητα των πράξεων, οι δυνάμεις έχουν μεγλύτερη προτεραιότητα από τον πολλαπλασιασμό και την πρόσθεση. Οπότε όταν έχουμε και δυνάμεις σε μια παράσταση πρώτα γίνονται οι πράξεις στις παρενθέσεις, μετά οι δυνάμεις και μετά οι πολλαπλασιασμοί και οι προσθέσεις. Την ίδια σειρά ακολουθεί και η python για τον υπολογισμό των πράξεων.

Ασκηση 2.3.1 *(Στο βιβλίο βρίσκεται στη Σελ. 21) Να εκτελεστούν οι πράξεις*

1. $(2 \cdot 5)^4 + 4 \cdot (3 + 2)^2$
2. $(2 + 3)^3 - 8 \cdot 3^2$

Οι αντίστοιχες εκφράσεις είναι (2*5)**4+4*(3+2)**2 και (2+3)**3 - 8*3**2.

```
>>> (2*5)**4+4*(3+2)**2
10100
>>> (2+3)**3 - 8*3**2
53
```

Η 8*3**2 υπολογίζεται ως $8 \cdot (3^2)$, δηλαδή $8 \cdot 9 = 72$, αφού πρώτα γίνεται η δύναμη και μετά οι πολλαπλασιασμοί.

Ασκηση 2.3.2 *Κάνε τις πράξεις: (α)* $3 \cdot 5^2$,
(β) $3 \cdot 5^2 + 2$,
(γ) $3 \cdot 5^2 + 2^2$,
(δ) $3 \cdot 5 + 2^2$,
(ε) $3 \cdot (5 + 2)^2$.

Αυτές οι πράξεις μπορούν να γίνουν στο REPL.

```
>>> 3*5**2
75
>>> 3*5**2 + 2
77
>>> 3*5**2 + 2**2
79
>>> 3*5 +2**2
19
>>> 3*(5 + 2)**2
147
```

Ασκηση 2.3.3 *Κάνε τις πράξεις: (α)* $3^2 + 3^3 + 2^3 + 2^4$,
(β) $(13 - 2)^4 + 5 \cdot 3^2$

```
>>> 3**2 +3**3 +2**3 +2**4
60
>>> (13-2)**4 + 5*3**2
14686
```

Ασκηση 2.3.4 *Βρες τις τιμές των παραστάσεων:*
(α) $(6 + 5)^2$ *και* $6^2 + 5^2$,
(β) $(3 + 6)^2$ *και* $3^2 + 6^2$.

```
>>> (6+5)**2
121
>>> 6**2+5**2
61
>>> (3+6)**2
81
>>> 3**2+6**2
45
```

2.4 Συγκρίσεις φυσικών αριθμών

Μπορούμε να συγκρίνουμε αριθμούς στην Python χρησιμοποιώντας τους τελεστές == (πληκτρολογούμε δύο φορές το =) για την *ισότητα*, > για το *μεγαλύτερο* και < για το *μικρότερο*. Επίσης μπορούμε να χρησιμοποιήσουμε >= για το *μεγαλύτερο ή ίσο* και <= για το *μικρότερο ή ίσο*, τέλος υπάρχει το != για το *δεν είναι ίσο*. Μπορείς να δοκιμάσεις τα παρακάτω:

```
>>> 123==123
True
>>> 123>123
False
>>> 123>122
True
>>> 123<123
False
>>> 123<124
True
>>> 123<=123
True
>>> 123<=124
True
>>> 123<=122
False
>>> 123>=123
True
>>> 123>=124
False
>>> 123>=122
True
>>> 122 != 123
True
>>> 122 != 122
False
```

Η Python επιστρέφει True (αληθές) όταν μία πρόταση ισχύει και False (ψευδές) όταν δεν ισχύει.

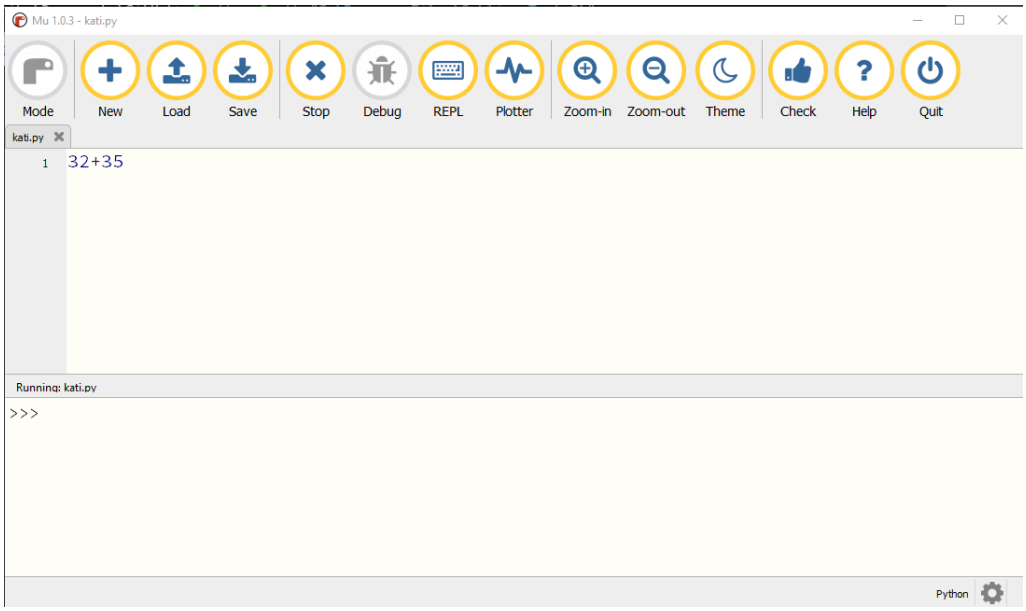
Σκέψου ότι για την Python η σύγκριση είναι και αυτή μια πράξη. Αντί η πράξη αυτή να δίνει σαν αποτέλεσμα έναν αριθμό δίνει σαν αποτέλεσμα το αληθές ή το ψευδές.

Για παράδειγμα:

Ασκηση 2.4.1 *Να συγκρίνετε τα* 3^2 *και* 2^3*.*

Η σύγκριση αυτή μπορεί να γίνει στο REPL. Δοκίμασε:

```
>>> 3**2 > 2**3
True
```



Σχήμα 2.1: Η εκτέλεση δεν δίνει κάποιο αποτέλεσμα

Αρα το 3^2 είναι μεγαλύτερο από το 2^3. Θυμήσου ότι το $3^2 = 9$, ενώ $2^3 = 8$.

Ασκηση 2.4.2

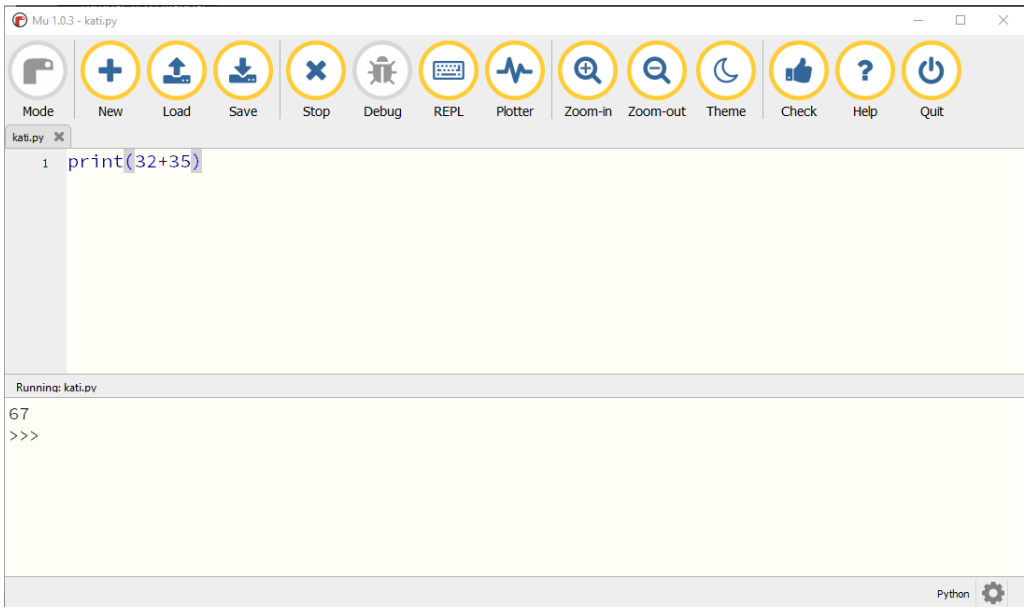
2.5 Η εντολή print

Ηρθε η ώρα να γράψεις εντολές στο πάνω παράθυρο, δηλαδή να γράψεις το πρώτο σου πρόγραμμα. Με βάση όσα ξέρεις προσπάθησε να γράψεις μια πράξη στο πάνω παράθυρο, για παράδειγμα $32 + 35$. Υστερα πάτησε το κουμπί της εκτέλεσης (Run). Μπορείς να δεις το αποτέλεσμα στην εικόνα 2.1.

Η Python εκτελεί την πράξη $32 + 35$, και υπολογίζει το αποτέλεσμα. Αν δεν το έκανε και υπήρχε κάποιο πρόβλημα θα εμφάνιζε κάποιο μήνυμα λάθους στο REPL. Το υπολογισμένο αποτέλεσμα δεν εμφανίζεται. Για να εμφανιστεί το αποτέλεσμα πρέπει να χρησιμοποιήσεις την εντολή print (εκτύπωσε). Η εντολή print εκτελείται ως εξής:

```
print(32+35)
```

Γράφουμε δηλαδή, print ανοίγουμε παρένθεση, γράφουμε αυτό που θέλουμε να εκτυπωθεί και κλείνουμε την παρένθεση. Οταν εκτελέσουμε το πρόγραμμα με την print τότε εμφανίζεται το αποτέλεσμα στο REPL (εικόνα 2.2). Μόλις έγραψες το πρώτο σου πρόγραμμα στην Python. Μάλι-



Σχήμα 2.2: Η εκτέλεση δίνει το αποτέλεσμα της πράξης

στα το πρόγραμμά σου κάνει κάτι. Υπολογίζει το αποτέλεσμα της πράξης $32 + 35$. Μπορείς να αποθηκεύσεις το πρόγραμμά σου στον υπολογιστή σου κάνοντας κλικ στο εικονίδιο Save του Mu (εικόνα 2.3).

2.6 Απαρίθμηση

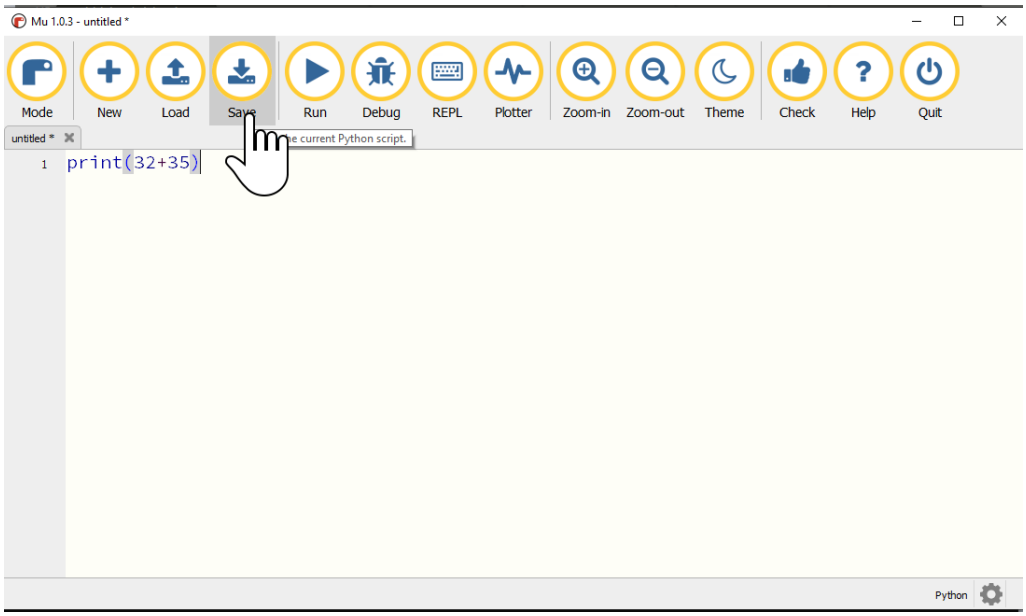
Είδαμε ότι η Python μπορεί να κάνει πολύ γρήγορα, πολύπλοκες πράξεις ακόμη και με δυνάμεις, αλλά δεν είδαμε ακόμη τις απλές ασκήσεις που υπάρχουν στις πρώτες σελίδες του βιβλίου. Οπως για παράδειγμα ποιοι είναι οι τρεις προηγούμενοι αριθμοί του 289 και ποιο οι δύο επόμενοι (Στο βιβλίο βρίσκεται στη Σελ. 13).

Τώρα που μάθαμε να γράφουμε προγράμματα σε Python μπορούμε να αντιμετωπίσουμε αυτό το πρόβλημα με το παρακάτω πρόγραμμα:

```
print(289-3)
print(289-2)
print(289-1)
print(289+1)
print(289+2)
```

που δίνει το αποτέλεσμα

```
286
287
```



Σχήμα 2.3: Αποθήκευση με το Mu

```
288
290
291
```

Πιο σωστό θα ήταν να γράψουμε ποιοι αριθμοί είναι οι προηγούμενοι και ποιοι οι επόμενοι. Σε αυτή την περίπτωση θα γράψουμε τις παρακάτω εντολές.

```
print("Οι  προηγούμενοι αριθμοί είναι:")
print(289-3)
print(289-2)
print(289-1)
print("Οι επόμενοι αριθμοί είναι:")
print(289+1)
print(289+2)
```

Για να εμφανίσει η print τις λέξεις που θέλουμε πρέπει να τις βάλουμε μέσα σε εισαγωγικά. Η Python υποστηρίζει είτε μονά εισαγωγικά, είτε διπλά. Αυτά εισάγονται συνήθως με το ίδιο κουμπί του πληκτρολογίου (κοντά στο ENTER), είτε με SHIFT ή χωρίς. Θυμήσου να κλείνεις τα εισαγωγικά με τον ίδιο τρόπο που τα άνοιξες. Στο πρόγραμμα Mu τα εισαγωγικά αυτά δεν φαίνονται όπως σε άλλα πρόγραμματα σαν ‘Εισαγωγικά’ ή “Εισαγωγικά ” ή «Εισαγωγικά», αλλά φαίνονται κάπως πιο απλά και ίδια στο άνοιγμα και το κλείσιμο `'Εισαγωγικά'` ή `"Εισαγωγικά"`.

Αν θέλουμε να αλλάξουμε το 289 και να βάλουμε έναν άλλο αριθμό,π.χ. το 132 θα πρέπει να αντικαταστήσουμε το 289 μέσα σε όλες τις εντολές print με το 132.

```
print("Οι προηγούμενοι αριθμοί είναι:")
print(132-3)
print(132-2)
print(132-1)
print("Οι επόμενοι αριθμοί είναι:")
print(132+1)
print(132+2)
```

Υπάρχει όμως ένας καλύτερος τρόπος, ο τρόπος αυτός είναι να δώσουμε ένα όνομα στον αριθμό μας. Μπορούμε να πούμε ότι το n είναι το όνομα του αριθμού. Αυτό γίνεται με την εντολή n=132. Τότε το πρόγραμμά μας γίνεται:

```
n = 132
print("Οι προηγούμενοι αριθμοί είναι:")
print(n-3)
print(n-2)
print(n-1)
print("Οι επόμενοι αριθμοί είναι:")
print(n+1)
print(n+2)
```

Μετά την εντολή n=132 η Python ξέρει ότι το n είναι ένα όνομα για το 132 και μπορεί να κάνει πράξεις με αυτό. Για παράδειγμα n+1 κάνει τώρα 133.

Αν θέλουμε να κάνουμε τώρα το ίδιο πρόγραμμα αλλά όχι για το 132 αλλά για το 210, χρειάζεται να αλλάξουμε μόνο μία γραμμή και το πρόγραμμά μας να γίνει ως εξής:

```
n = 210
print("Οι προηγούμενοι αριθμοί είναι:")
print(n-3)
print(n-2)
print(n-1)
print("Οι επόμενοι αριθμοί είναι:")
print(n+1)
print(n+2)
```

Στην Python, όταν δίνουμε ένα όνομα σε έναν αριθμό (με τον τελεστή =) τότε δημιουργούμε μια μεταβλητή. Η μεταβλητή έχει ένα όνομα, στην περίπτωσή μας το n, και μια τιμή, στην περίπτωσή μας το 210.

Αν αντί για τους επόμενους δύο αριθμούς θέλαμε τους επόμενους **δέκα** θα γράφαμε ένα πρόγραμμα όπως το παρακάτω:

```
n = 210
```

```
print(n)
print(n+1)
print(n+2)
print(n+3)
print(n+4)
print(n+5)
print(n+6)
print(n+7)
print(n+8)
print(n+9)
print(n+10)
```

Το παραπάνω πρόγραμμα εμφανίζει και τον αριθμό μας n, δηλαδή το 210.

Για να μην γράφουμε πολλές εντολές όταν κάνουμε το ίδιο πράγμα χρησιμοποιούμε την εντολή for. Το πρόγραμμά μας με την for μπορεί να γίνει:

```
n = 210
for i in 0,1,2,3,4,5,6,7,8,9,10:
    print(n+i)
```

Οταν γράψεις την for στην Python θα πρέπει να δηλώσεις ποιες εντολές θα εκτελεστούν πολλές φορές. Αυτή η δήλωση γίνεται βάζοντας αυτές τις εντολές λίγο πιο μέσα χρησιμοποιώντας το πλήκτρο κενό ή το πλήκτρο tab. Μια καλή πρακτική είναι να βάζεις τέσσερα κενά. Ετσι, πριν την εντολή `print(n+i)` βάζεις τέσσερα κενά δηλαδή `    print(n+i)`. Το πρόγραμμα αυτό σημαίνει πως για το i μέσα στο σύνολο 0, 1, 2, 3, ... 10 και με αυτή τη σειρά εμφάνισε το n+i. Ετσι το αποτέλεσμα είναι το αναμενόμενο

```
210
211
212
213
214
215
216
217
218
219
220
```

Στην Python υπάρχει ένας πιο εύκολος τρόπος να γράψουμε τους αριθμούς από το 0 έως το 10. Αυτός ο τρόπος είναι η εντολή range και συγκεκριμένα η range(11). Η range(11) φτιάχνει τους αριθμούς από το 0 μέχρι το 10 οι οποίοι είναι σε πλήθος 11. Ετσι το πρόγραμμά μας γίνεται:

```
n = 210
for i in range(11):
    print(n+i)
```

Μπορούμε και να μετρήσουμε τους πρώτους 100 αριθμούς ως εξής:

```
for i in range(100):
    print(i)
```

Σκέψου αν θα δεις τον αριθμό 100 στο αποτέλεσμα του παραπάνω προγράμματος.

Μπορούμε να δούμε αριθμούς εύκολα με την Python αλλά θα χρειαστεί ξεχωριστό πρόγραμμα αν θέλουμε να εμφανίζεται το λεκτικό για κάθε αριθμό.

Ενα τέτοιο πρόγραμμα είναι το παρακάτω:

```
print('μηδέν')
print('ένα')
print('δύο')
print('τρία')
print('τέσσερα')
print('πέντε')
print('έξι')
print('εφτά')
print('οχτώ')
print('εννιά')
print('δέκα')
print('έντεκα')
print('δώδεκα')
print('δεκατρία')
print('δεκατέσσερα')
print('δεκαπέντε')
print('δεκαέξι')
print('δεκαεφτά')
print('δεκαοχτώ')
print('δεκαεννιά')
```

Το παραπάνω πρόγραμμα μπορεί να γίνει πιο μαζεμένο με τη χρήση λίστας. Μια λίστα μπορεί να περιέχει τα λεκτικά για κάθε αριθμό. Η λίστα στην Python σημειώνεται με τις τετράγωνες αγκύλες [και]. Τα στοιχεία της χωρίζονται με κόμμα. Ετσι η λίστα που θέλουμε τώρα είναι η εξής:

```
lektika = ['μηδέν','ένα','δύο','τρία','τέσσερα','πέντε','έξι','εφτά','οχτώ','εννιά','
```

Χρησιμοποιούμε τις τετράγωνες αγκύλες και τον αριθμό του στοιχείου που θέλουμε να προσπελάσουμε σε μια λίστα. Η αρίθμηση της λίστας ξεκινάει από το 0. Ετσι, στη λίστα που βλέπουμε παραπάνω το lektika[0] θα είναι η λέξη ’μηδέν’ (θυμηθείτε τα εισαγωγικά), το lektika[1] θα είναι η λέξη ’ένα’ κ.ο.κ.

Αν θέλετε μπορείτε να κάνετε μια μικρή δοκιμή στο REPL.

```
>>>lektika = ['μηδέν','ένα','δύο']
>>>lektika[0]μηδέν

>>>lektika[1]ένα
```

```
>>>lektika[2]δύο
```

Με τη χρήση της λίστας μπορούμε να εμφανίσουμε τους αριθμούς με τη σειρά χρησιμοποιώντας την εντολή for.

```
lektika = ['μηδέν','ένα','δύο','τρία','τέσσερα','πέντε','έξι','εφτά',
'οχτώ','εννιά','δέκα','έντεκα','δώδεκα','δεκατρία',
'δεκατέσσερα','δεκαπέντε','δεκαέξι','δεκαεφτά','δεκαοχτώ',
'δεκαεννιά']
for i in range(20):
    print(lektika[i])
```

Ομως παρότι δεν γράφουμε είκοσι φορές την εντολή print πάλι δίνουμε όλα τα ονόματα στο πρόγραμμά μας βάζοντάς τα σε μια λίστα. Μπορούμε να το αποφύγουμε υπολογίζοντας το λεκτικό. Από το δώδεκα και μέτα το λεκτικό ενός αριθμού i είναι το 'δεκα' και μετά το λεκτικό του αριθμού i-10. Για παράδειγμα, το δεκαοχτώ είναι το 'δεκα' ακολουθούμενο από το λεκτικό του αριθμού που προκύπτει αν αφαιρέσουμε 10 από το 18.

Η Python μπορεί να κάνει πράξεις και με τις λέξεις, η πρόσθεση λέξεων σημαίνει να τις βάλεις δίπλα δίπλα με τη σειρά. Δοκίμασε

```
In [1]: 'δεκα' + 'τρία'
Out[1]: 'δεκατρία'
```

Με αυτή την ευκολία το πρόγραμμά μας γίνεται:

```
lektika = ['μηδέν','ένα','δύο','τρία','τέσσερα','πέντε','έξι','εφτά',
'οχτώ','εννιά','δέκα','έντεκα','δώδεκα','δεκατρία',
'δεκατέσσερα','δεκαπέντε','δεκαέξι','δεκαεφτά','δεκαοχτώ',
'δεκαεννιά']
for i in range(20):
    if i<=12:
        print(lektika[i])
    else:
        print('δεκα'+lektika[i-10])
```

Μάλιστα, το πρόγραμμα υπολογίζει τα λεκτικά από το 13 και μετά και δεν χρειάζεται να τα θυμάται. Μπορούμε να τα διαγράψουμε από τη λίστα.

```
lektika = ['μηδέν','ένα','δύο','τρία','τέσσερα','πέντε','έξι','εφτά',
'οχτώ','εννιά','δέκα','έντεκα','δώδεκα']
for i in range(20):
    if i > 12:
        print('δέκα'+lektika[i-10])
    else:
        print(lektika[i])
```

Τώρα μπορούμε να πάμε μέχρι το 29.

```
lektika = ['μηδέν','ένα','δύο','τρία','τέσσερα','πέντε','έξι','εφτά',
```

```
'οχτώ','εννιά','δέκα','έντεκα','δώδεκα']
for i in range(30):
    if i>20:
        print('είκοσι' + lektika[i-20])
    elif i > 12:
        print('δέκα'+lektika[i-10])
    else:
        print(lektika[i])
```

Το elif είναι συντομογραφία για το else if. Στο σημείο που το έβαλες τώρα σημαίνει αν το i δεν είναι μεγαλύτερο του 20 (else) και είναι μεγαλύτερο από το 12 (if). Αρα το `print('δέκα'+lektika[i-10])` γίνεται μόνο αν το i είναι μικρότερο ή ίσο του 20 και μεγαλύτερο από 12. Το αποτέλεσμα φαίνεται παρακάτω:

```
ένα
δύο
τρία
τέσσερα
πέντε
έξι
εφτά
οχτώ
εννιά
δέκα
έντεκα
δώδεκα
δέκατρία
δέκατέσσερα
δέκαπέντε
δέκαέξι
δέκαεφτά
δέκαοχτώ
δέκαεννιά
 δέκαδέκα
είκοσιένα
είκοσιδύο
είκοσιτρία
είκοσιτέσσερα
είκοσιπέντε
είκοσιέξι
είκοσιεφτά
είκοσιοχτώ
είκοσιεννιά
```

Οπότε καταλαβαίνουμε ότι το είκοσι χρειάζεται ειδικό χειρισμό. Με τη χρήση της elif μπορούμε να βάλουμε και ειδικό χειρισμό για το 20.

```
lektika = ['μηδέν','ένα','δύο','τρία','τέσσερα','πέντε','έξι','εφτά',
'οχτώ','εννιά','δέκα','έντεκα','δώδεκα']
```

```
for i in range(30):
    if i>20:
        print('είκοσι' + lektika[i-20])
    elif i==20:
        print('είκοσι')
    elif i > 12:
        print('δέκα'+lektika[i-10])
    else:
        print(lektika[i])
```

```
ένα
δύο
τρία
 .
 .
 .
έντεκα
δώδεκα
δέκατρία
 .
 .
 .
δέκαεννιά
είκοσι
είκοσιένα
 .
 .
 .
είκοσιεννιά
```

2.7 Στρογγυλοποίηση

Το βιβλίο των Μαθηματικών της Α' Γυμνασίου αναφέρει πως Για να στρογγυλοποιήσουμε έναν φυσικό αριθμό (Στο βιβλίο βρίσκεται στη Σελ. 12):

1. Προσδιορίζουμε την τάξη στην οποία θα γίνει η στρογγυλοποίηση
2. Εξετάζουμε το ψηφίο της αμέσως μικρότερης τάξης
3. Αν αυτό το ψηφίο είναι μικρότερο του 5 (δηλαδή 0, 1, 2, 3 ή 4) το ψηφίο αυτό και όλα τα ψηφία των υπόλοιπων τάξεων μηδενίζονται.
4. Αν είναι μεγαλύτερο ή ίσο του 5 (δηλαδή 5, 6, 7, 8 ή 9) το ψηφίο αυτό και όλα τα ψηφία των υπόλοιπων τάξεων αντικαθίστανται από το 0 και το ψηφίο της τάξης στρογγυλοποίησης αυξάνεται κατά 1.

Ας πούμε ότι θέλουμε να στρογγυλοποιήσουμε τον αριθμό 454.018.512 στα εκατομμύρια. Η απάντηση που περιμένουμε είναι 454 εκατομμύρια. Για να τα καταφέρουμε θα χρησιμοποιήσουμε την διαίρεση. Ομως στην Python υπάρχουν δύο διαιρέσεις μία με το σύμβολο/και μία με το σύμβολο //. Ας δούμε τις διαφορές τους στο REPL.

```
>>> x = 454018512
>>> print(x/1000000)
454.018512
>>> print(x//1000000)
454
```

Η «κανονική» διαίρεση, με τη μία κάθετο /, δίνει το αποτέλεσμα της διαίρεσης με τα δεκαδικά ψηφία. Η «ακέραια» διαίρεση δίνει μόνο τον ακέραιο αριθμό. Δεν μπορούμε να πούμε ότι η ακέραια διαίρεση θα μας δώσει την στρογγυλοποίηση γιατί η ακέραια διαίρεση δεν στρογγυλοποιεί τα δεκαδικά ψηφία αλλά τα απορρίπτει εντελώς. Ετσι, ακόμη και αν είχαμε 454918512 κατοίκους η ακέραια διαίρεση θα δώσει 454 αντί για το στρογγυλοποιημένο που είναι 455.

```
>>> x = 454918512
>>> print(x/1000000)
454.918512
>>> print(x//1000000)
454
```

Χρειάζεται επομένως να δούμε το ψηφίο της αμέσως χαμηλότερης τάξης το οποίο είναι το πρώτο δεκαδικό της κανονικής διαίρεσης. Για να το απομονώσουμε αφαιρούμε από το αποτέλεσμα της κανονικής διαίρεσης το ακέραιο μέρος.

```
>>> x = 454018512
>>> x/1000000 - x//1000000
0.018511999999986983
```

Οπότε τώρα έχουμε δύο ενδεχόμενα αν το αποτέλεσμα αυτής της πράξης είναι μικρότερο από 0.5 όπως παραπάνω τότε το αποτέλεσμα που ψάχνουμε είναι το αποτέλεσμα της ακέραιας διαίρεσης. Αλλιώς πρέπει να προσθέσουμε ένα στο αποτέλεσμα της ακέραιας διαίρεσης. Αυτό γίνεται με την εντολή if, που σημαίνει στα αγγλικά αν. Για ευκολία μπορούμε να ονομάσουμε d την διαφορά των δύο διαιρέσεων με την εντολή:

```
d = x/1000000 - x//1000000
```

Επειδή το πρόγραμμα γίνεται μεγαλύτερο τώρα θα το γράψουμε στο πάνω παράθυρο του Mu.

```
x = 454018512
d = x/1000000 - x//1000000
```

```
if d < 0.5:
    print(x//1000000)
else:
    print(x//1000000 + 1)
```

Την **if** την γράφουμε ως εξής:

```
if συνθήκη:
    εντολές που εκτελούνται
    αν ισχύει η συνθήκη
else:
    εντολές που εκτελούνται
    αν δεν ισχύει η συνθήκη
```

Θυμήσου να βάζεις την άνω κάτω τελεία μετά τη συνθήκη και μετά τη λέξη else που σημαίνει αλλιώς.

Αν στο ίδιο πρόγραμμα και βάλεις αντί για 454.018.512 τον αριθμό 454.918.512 θα δεις ότι θα εμφανιστεί το σωστό αποτέλεσμα (455).

Αν θέλεις στρογγυλοποίηση στις χιλιάδες τότε το πρόγραμμά σου γίνεται:

```
x = 454018512
d = x/1000 - x//1000
if d < 0.5:
    print(x//1000)
else:
    print(x//1000 + 1)
```

και το αποτέλεσμα είναι 454019.

Ασκηση 2.7.1 *Για να γίνει το 454.018.512, 450 εκατομμύρια (Στο βιβλίο βρίσκεται στη Σελ. 12) η στρογγυλοποίηση γίνεται στις δεκάδες των εκατομμυρίων. Μπορείς να γράψεις ένα πρόγραμμα που να στρογγυλοποιεί αριθμούς στις δεκάδες των εκατομμυρίων;*

2.8 Επανάληψη στις πράξεις

Ασκηση 2.8.1 *Συμπλήρωσε τον πίνακα τα τετράγωνα και τους κύβους των αριθμών από το 8 μέχρι το 25 (Στο βιβλίο βρίσκεται στη Σελ. 22).*

```
for a in range(8,26):
    print(a**2,end =" ")
print()
print()
for a in range(8,26):
    print(a**3,end=" ")
```

Το αποτέλεσμα αυτού του προγράμματος είναι:

```
64 81 100 121 144 169 196 225 256 289 324 361 400 441 484 529 576 625

512 729 1000 1331 1728 2197 2744 3375 4096 4913 5832 6859 8000 9261
10648 12167 13824 15625
```

Η εντολή print μπορεί να πάρει περισσότερα από ένα ορίσματα, το πρώτο όρισμα είναι αυτό που θα εμφανίσει. Το δεύτερο όρισμα που δώσαμε είναι το end και το ορίσαμε ίσο με το κενό (`end=" "`) που σημαίνει ότι η print όταν εμφανίσει το πρώτο όρισμα δεν θα αλλάξει γραμμή αλλά θα αφήσει ένα κενό. Η εντολή print() αλλάζει απλά γραμμή.

Ασκηση 2.8.2 *Βρες τα τετράγωνα των αριθμών 10,20,30,40,50,60,70,80 και 90 (Στο βιβλίο βρίσκεται στη Σελ. 22).*

Το πρόγραμμα είναι το εξής:

```
for i in range(10,100,10):
    print(i**2,end=',')
```

και το αποτέλεσμα της εκτέλεσης του προγράμματος είναι

```
100,400,900,1600,2500,3600,4900,6400,8100,
```

Ασκηση 2.8.3 *Βρες τους κύβους των αριθμών 10,20,30,40,50*

```
for i in range(10,60,10):
    print(i**3,end=', ')
```

Το αποτέλεσμα της εκτέλεσης είναι:

```
1000, 8000, 27000, 64000, 125000,
```

2.9 Ανάπτυγμα

Ασκηση 2.9.1 *(Στο βιβλίο βρίσκεται στη Σελ. 21)Να γραφεί το ανάπτυγμα του αριθμού 7.604 με χρήση των δυνάμεων του 10.*

Η απάντηση είναι $7 \cdot 10^3 + 6 \cdot 10^2 + 0 \cdot 10^1 + 4$. Με συμβολισμό της Python η απάντηση που περιμένουμε είναι:

```
7*10**3+6*10**2+0*10+4
```

Ας υποθέσουμε ότι ξέρουμε ότι ο αριθμός είναι τετραψήφιος, πως μπορούμε να βρούμε το ανάπτυγμα του. Ξεκινάμε από το πρώτο ψηφίο. Ποιο είναι το πρώτο ψηφίο; Το πρώτο ψηφίο προκύπτει αν διαιρέσουμε τον αριθμό με το 1000 και κρατήσουμε το ακέραιο μέρος. Δοκίμασε στο REPL:

```
In [1]: 7604//1000
Out[1]:7
```

Βρήκες το πρώτο ψηφίο, πώς μπορείς να βρεις το δεύτερο; Ας διαιρέσουμε με το 100.

```
In [1]:7604//100
Out[1]:76
```

Στην ουσία δεν μπορείς να διαιρέσεις τον αρχικό αριθμό με το 100 αλλά αυτόν που σου μένει αφού αφαιρέσεις το πρώτο ψηφίο που έχεις ήδη βρει δηλαδή το 604.

```
In [2]:604//100
Out[2]:6
```

Ετσι για τα σωστά βήματα είναι:

1. Διαιρείς τον αριθμό με το 1000 και κρατάς το ακέραιο μέρος
2. Αφαιρείς από τον αριθμό τις χιλιάδες που βρήκες

Ας ονομάσουμε τον αριθμό 7604, n (`n=7604`), και το πρώτο ψηφίο, στην περίπτωσή μας τις χιλιάδες, prwto.

```
In [1]: n = 7604

In [2]: prwto = n//1000

In [3]: prwto
Out[3]: 7

In [4]: n = n - prwto*1000

In [5]: n
Out[5]: 604
```

Ας δούμε λίγο αυτή την εντολή:

```
n = n - prwto*1000
```

Θυμηθείτε ότι εκείνη τη στιγμή το n έχει την τιμή 7604 και το prwto την τιμή 7. Η παραπάνω εντολή σημαίνει:

1. Κάνε τις πράξεις που υπάρχουν δεξιά από το σύμβολο ίσον
2. Δώσε σαν τιμή στην μεταβλητή που υπάρχει αριστερά από το σύμβολο ίσον το αποτέλεσμα των πράξεων

Ετσι η Python κάνει πρώτο `n-prwto*1000` δηλαδή $7604 - 7 * 1000 = 604$ και αυτό το αποτέλεσμα το δίνει σαν τιμή στην μεταβλητή που υπάρχει

αριστερά από το = δηλαδή στη μεταβλητή n. Ετσι το n τώρα είνι 604. *Προσοχή!* Η τιμή 7604 δεν υπάρχει στην μεταβλητή n. Με αυτόν τον τρόπο το n έχει πάντα τον αριθμό που χρειάζεσαι για να απομονώσεις το επόμενο ψηφίο του αριθμού.

Ετσι ένα συνολικό πρόγραμμα που μπορείς να γράψεις είναι:

```
n = 7604
prwto = n//1000
n = n - prwto*1000
deutero = n//100
n = n - deutero*100
trito = n //10
n = n - trito*10
tetarto = n
print(prwto,end='')
print('*10**3+',end='')
print(deutero,end='')
print('*10**2+',end='')
print(trito,end='')
print('*10+',end='')
print(tetarto)
```

Οταν το εκτελέσεις δίνει το σωστό αποτέλεσμα:

```
7*10**3+6*10**2+0*10+4
```

Το παραπάνω πρόγραμμα δουλεύει με όλους τους τετραψήφιους αριθμούς, απλά άλλαξε το n σε όποιον αριθμό θέλεις στην αρχή του προγράμματος.

Ομως οι 7 εντολές print δεν είναι ο καλύτερος τρόπος να γράψεις το αποτέλεσμα. Θα ήταν καλύτερα να τυπώσουμε αυτά που πρέπει για κάθε ψηφίο χωριστά ώστε να έχουμε τέσσερις εντολές print ως εξής:

```
n = 7604
prwto = n//1000
n = n - prwto*1000
deutero = n//100
n = n - deutero*100
trito = n //10
n = n - trito*10
tetarto = n
print(prwto + '*10**3+',end='')
print(deutero + '*10**2+',end='')
print(trito + '*10+',end='')
print(tetarto)
```

Εξάλλου όταν έχουμε πρόσθεση με λέξεις η Python τις βάζει δίπλα δίπλα οπότε για το `print(prwto + '*10**3+',end='')` θα περιμέναμε να εμφανιστεί το `7*10**3+`. Αν όμως εκτελέσεις το παραπάνω πρόγραμμα θα προκύψει ένα μήνυμα λάθους.

```
Traceback (most recent call last):
  File "a.py", line 9, in <module>
    print(prwto + '*10**3+',end='')
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Αν προσέξουμε λίγο θα δούμε ότι το λάθος αφορά την ένατη γραμμή (line 9) και το λάθος είναι TypeError: unsupported operand type(s) for +: 'int' and 'str'. Σε μετάφραση από τα αγγλικά το μήνυμα λάθους γράφει:

ΛάθοςΤύπων: μη υποστηριζόμενοι τύποι για το +: 'int' και 'str'

Τι είναι οι τύποι και προκύπτουν λάθη από αυτούς;

Οτιδήποτε χρησιμοποιούμε στην Python έχει τύπο. Μάλιστα μπορούμε να δούμε τον τύπο αυτό με την εντολή type. Ετσι δοκίμασε:

```
In [1]: type(7)
Out[1]:<class 'int'>
In [2]: type('a')
Out[2]:<class 'str'>
In [2]: type('7')
Out[2]:<class 'str'>
```

Βλέπουμε ότι οι αριθμοί έχουν τύπο 'int', θα αγνοήσουμε τη λέξη class προς το παρόν. Ενώ οι λέξεις που έχουν τα εισαγωγικά έχουν τύπο 'str'. Το 'int' προκύπτει από την αγγλική λέξη 'integer' που σημαίνει ακέραιος, και το 'str' προκύπτει από την αγγλική λέξη 'string' που σημαίνει μια σειρά από γράμματα και αριθμούς. Στα ελληνικά το 'string' το μεταφράζουμε ως αλφαριθμητικό.

Το πρόβλημα είναι ότι η Python δεν μπορεί να προσθέσει έναν ακέραιο με ένα αλφαριθμητικό. Γι' αυτό δίνει και το μήνυμα λάθους. Ωστόσο, αυτό το πρόβλημα έχει λύση και είναι η μετατροπή του αριθμού σε αλφαριθμητικό με την εντολή str(). Δοκίμασε:

```
In [1]: type(7)
Out[1]: int

In [2]: str(7)
Out[2]: '7'

In [3]: type('7')
Out[3]: str
```

Το παραπάνω πρόγραμμα γίνεται λοιπόν:

```
n = 7604
prwto = n//1000
n = n - prwto*1000
deutero = n//100
n = n - deutero*100
trito = n //10
```

```
n = n - trito*10
tetarto = n
print(str(prwto) + '*10**3+',end='')
print(str(deutero) + '*10**2+',end='')
print(str(trito) + '*10+',end='')
print(str(tetarto))
```

Που και πάλι δίνει τη σωστή απάντηση.

Καλύτερα είναι να βάλουμε τα στοιχεία prwto, deutero, trito και tetarto σε μια λίστα που θα την ονομάσουμε psifia και θα έχει τέσσερα στοιχεία. Καλό είναι στην αρχή να αρχικοποιούμε τη λίστα με κάποια τιμή ειδικά στην περίπτωση που ξέρουμε πόσο μεγάλη θα είναι όπως τώρα.

```
n = 7604
psifia  = [0,0,0,0]
psifia[0] = n//1000
n = n - psifia[0]*1000
psifia[1] = n//100
n = n - psifia[1]*100
psifia[2] = n //10
n = n - psifia[2]*10
psifia[3] = n
print(str(psifia[0]) + '*10**3+',end='')
print(str(psifia[1]) + '*10**2+',end='')
print(str(psifia[2]) + '*10+',end='')
print(str(psifia[3]))
```

Φαίνεται ότι κάνουμε τα ίδια πράγματα τρεις φορές όπως βλέπεις εδώ:

```
psifia[0] = n//1000
n = n - psifia[0]*1000
psifia[1] = n//100
n = n - psifia[1]*100
psifia[2] = n //10
n = n - psifia[2]*10
```

Θα προσπαθήσουμε να τα κάνουμε με for όπου το i θα μετράει 0,1,2 έτσι το psifia[0] θα γίνει psifia[i]. Ομως θα πρέπει να υπολογίσουμε το 1000 το 1000 είναι 10**3 και στην επόμενη επανάληψη είναι 10**2 κ.ο.κ. οπότε είναι 10**(3-i). Αρα οι τρεις παραπάνω εντολές μπορούν να αντικατασταθούν με μία for

```
for i in range(3):
    psifia[i] = n//10**(3-i)
    n = n - psifia[i]* 10**(3-i)
```

Το ίδιο πρέπει να γίνει και με τις τρεις εντολές print:

```
print(str(psifia[0]) + '*10**3+',end='')
print(str(psifia[1]) + '*10**2+',end='')
print(str(psifia[2]) + '*10+',end='')
```

Θα πρέπει να κάνουμε έναν ιδιαίτερο χειρισμό για τις δυνάμεις όπου θα πρέπει να μειώνονται καθώς συνέχίζουν οι επαναλήψεις οπότε αντί για '*10**3΄ χρειαζόμαστε '*10**(3-i)+' όμως το 3-i είναι αριθμός οπότε για να το γράψουμε στην Python θα χρειαστεί να βάλουμε το str() και να γίνει `'*10**'+str(3-i)+'+'`. Οι παραπάνω τρεις εντολές μπορούν τώρα να γραφούν με μία for ως εξής:

```
for i in range(3):
    print(str(psifia[i]) + '*10**' + str(3-i) + '+',end='')
```

και όλο το πρόγραμμα να γίνει:

```
n = 7604
psifia  = [0,0,0,0]
for i in range(3):
    psifia[i] = n//10**(3-i)
    n = n - psifia[i]* 10**(3-i)
psifia[3] = n
for i in range(3):
    print(str(psifia[i]) + '*10**' + str(3-i) + '+',end='')
print(str(psifia[3]))
```

Πώς μπορεί το πρόγραμμα αυτό να δουλεύει για όλους τους ακέραιους ανεξάρτητα από το μέγεθός τους. Αν μετατρέψουμε τον ακέραιο σε αλφαριθμητικό η Python μπορεί να μας πει πόσο μεγάλος είναι με την εντολή len.

```
In [1]: n = 7604

In [2]:len(str(n))
Out[2]: 4

In [3]:n = 7102234

In [4]:len(str(n))
Out[4]: 7
```

Και η αρχικοποίηση του πίνακα μπορεί να γίνει για όσα ψηφία θέλουμε (plithos) με την εντολή `[0]*plithos`:

```
In [1]: plithos = 7
In [2]: psifia = [0] * plithos

In[2]: psifia
Out[2]: [0, 0, 0, 0, 0, 0, 0]
```

Αν υπολογίσουμε το plithos των ψηφίων με το len(str(n)) θα προκύψει 4, ωστόσο εμείς στο πρόγραμμά μας κάνουμε επαναλήψεις τρεις φορές γιατί χρειαζόμαστε ειδικό χειρισμό στο τελευταίο ψηφίο. Ετσι αντικαθιστούμε το 3 με plithos-1 παντού στο πρόγραμμα.

```
n = 7604
plithos = len(str(n))
psifia  = [0]*plithos
for i in range(plithos-1):
    psifia[i] = n//10**(plithos-1-i)
    n = n - psifia[i]* 10**(plithos-1-i)
psifia[plithos-1] = n
for i in range(plithos-1):
    print(str(psifia[i]) + '*10**' + str(plithos-1-i) + '+',end='')
print(str(psifia[plithos-1]))
```

Η Python έχει διάφορους τρόπους να συμπυκνώνει μεγάλα προγράμματα ακόμη και σε μία γραμμή. Ενας τέτοιος τρόπος να γραφτεί το ανάπτυγμα είναι ο εξής, αλλά τέτοιους τρόπους θα τους δούμε σε επόμενα κεφάλαια:

```
>>> n = 7604
>>> '+'.join([x+'*10**'+str(len(str(n))-1-i)
for (i,x) in enumerate(list(str(n)))])

'7*10**3+6*10**2+0*10**1+4*10**0'
```

2.10 Ιστορικό σημείωμα

(Στο βιβλίο βρίσκεται στη Σελ. 17)

Οταν ο δάσκαλος ζήτησε από τους υπόλοιπους μαθητές να υπολογίσουν το άθροισμα $1+2+3+\ldots+98+99+100$, πριν οι υπόλοιποι αρχίσουν τις πράξεις ο μικρός Γκάους το είχε ήδη υπολογίσει. Ο δάσκαλος ρώτησε έκπληκτος πώς το βρήκε. Τότε εκείνος έγραψε στον πίνακα: Ο Γκάους σκέφτηκε πως το άθροισμα

$$1 + 2 + 3 + \ldots + 99 + 100$$

είναι ίδιο με το

$$100 + 99 + 98 + \ldots + 2 + 1$$

και αν αθροίσουμε τον πρώτο όρο με τον πρώτο όρο, τον δεύτερο με τον δεύτερο κ.ο.κ. Συνολικά αυτό γίνεται:

$$\underbrace{(1+100)+(2+99)+\ldots(99+2)+(100+1)}_{50\text{φορές}} = \tag{2.1}$$

$$101 \cdot 100 \tag{2.2}$$

(2.3)

Αρα το άθροισμα $1 + 2 + 3 + \ldots + 99 + 100$ είναι $\frac{1}{2}101 \times 100 = 5050$.

Μπορείς να υπολογίσεις το άθροισμα $1 + 2 + 3 + \ldots + 999 + 1000$ με τον τρόπο του Γκάους;

Ας δούμε το πρόβλημα από την πλευρά της Python. Ξέρουμε ήδη να εμφανίζουμε τους αριθμούς από το 1 έως το 100 με το παρακάτω πρόγραμμα:

```
for i in range(101):
    print(i)
```

Πώς όμως θα αθροίσουμε τους αριθμούς αυτούς. Θα φτάξουμε μιά νέα μεταβλητή athroisma και σε αυτή θα προσθέτουμε το i κάθε φορά. Το πρόγραμμα γίνεται:

```
athroisma = 0
for i in range(101):
    athroisma = athroisma + i
print(athroisma)
```

Ομως επειδή το άθροισμα είναι χρήσιμο σε πολλές περιπτώσεις η Python έχει έτοιμο το άθροισμα με την εντολή sum. Δοκίμασε:

```
>>> sum([1,2,3])
6
```

Με τον ίδιο τρόπο μπορείς να βρεις το άθροισμα $1+2+3+\ldots+99+100$:

```
>>> sum(range(101))
5050
```

Τέλος ο τρόπος του μικρού Γκάους είναι:

```
>>> 101*100/2
5050
```

Με την Python μπορούμε να υπολογίσουμε το άθροισμα από το 1 έως το 1000 και με τους δύο τρόπους:

```
>>> sum(range(1001))
500500
>>> 1000*1001/2
500500.0
```

Στην Python το 500500.0 σημαίνει πως το δεκαδικό μέρος είναι 0 οπότε το αποτέλεσμα είναι και πάλι 500500.

2.11 Ευκλείδια διαίρεση

Ασκηση 2.11.1 *Για να αποφασίσει ο καθηγητής με ποιο τρόπο θα παρατάξει τους 168 μαθητές για την παρέλαση, πρέπει να διαιρέσει το 168 με τους αριθμούς 3, 4, 5, 6 και 7.*

Διαίρεση με το τρία:

```
>>> print(168/3,168%3)
56.0 0
```

Η διαίρεση γίνεται με τον τελεστή /, οπότε 168/3 είναι το πηλίκο της διαίρεσης του 168 με το 3. Στους υπολογιστές δεν χρησιμοποιούμε την άνω κάτω τελεία για διαίρεση αλλά την κάθετο /, που συνήθως βρίσκεται χαμηλά στο πληκτρολόγιο και στο αριθμητικό πληκτρολόγιο.

Το υπόλοιπο δίνεται με τον τελεστή %, οπότε 168%3 είναι το υπόλοιπο της διαίρεσης του 168 με το 3. Στο συγκεκριμένο παράδειγμα παρατηρούμε ότι το 168 διαιρείται ακριβώς με το 3 και δίνει πηλίκο 56, οπότε ο καθηγητής μπορεί να παρατάξει τους 168 μαθητές σε 56 τριάδες.

Παρόμοια, η διαίρεση του αριθμού 168 με τους αριθμούς 4, 6, και 7 δίνει τα πηλίκα: 42, 28 και 24 αντίστοιχα.

```
>>> print(168/4,168%4)
42.0 0
>>> print(168/6,168%6)
28.0 0
>>> print(168/7,168%7)
24.0 0
```

Επομένως, μπορούν να παραταχθούν οι μαθητές σε 42 τετράδες ή 28 εξάδες ή σε 24 επτάδες.

Τέλος, η διαίρεση του 168 με το 5 δίνει πηλίκο 33 και αφήνει υπόλοιπο 3.

```
>>> print(168/5,168%5)
33.6 3
```

Αρα, δεν μπορεί ο καθηγητής να παρατάξει τους μαθητές σε πλήρεις πεντάδες.

Εκτός από τον τελεστή / για την διαίρεση υπάρχει και ο τελεστής // ο οποίος όμως δίνει το ακέραιο μέρος του πηλίκου. Οπότε αν γράψουμε 168//5 το αποτέλεσμα θα είναι ο ακέραιος αριθμός 33, που όμως είναι μόνο το ακέραιο μέρος του πηλίκου.

Χρησιμοποιώντας αυτούς τους τελεστές μπορεί να γραφεί μια συνάρτηση που να εμφανίζει την ευκλείδια διαίρεση μεταξύ δύο αριθμών:

```
def eykleidia(D,d):
  print(str(D)+':' + str(d) + '=' + str(D//d) + '*' + str(d) + '+' + str(D%d))
```

Για παράδειγμα (Ασκηση 1):

```
>>> eykleidia(4002,69)
4002:69=58*69+0
>>> eykleidia(1445,17)
1445:17=85*17+0
>>> eykleidia(925,37)
```

```
925:37=25*37+0
>>> eykleidia(3621,213)
3621:213=17*213+0
>>> eykleidia(35280,2940)
35280:2940=12*2940+0
>>> eykleidia(5082,77)
5082:77=66*77+0
```

Ασκηση 2:

```
>>> eykleidia(65,5)
65:5=13*5+0
>>> eykleidia(30,3)
30:3=10*3+0
>>> eykleidia(46592,52)
46592:52=896*52+0
```

Ασκηση 3:

```
>>> eykleidia(125,3)
125:3=41*3+2
>>> eykleidia(762,19)
762:19=40*19+2
>>> eykleidia(1500,35)
1500:35=42*35+30
>>> eykleidia(300,18)
300:18=16*18+12
```

Ασκηση 2.11.2 *Ποια μπορεί να είναι τα υπόλοιπα της διαίρεσης ν:8.*

```
>>> for i in range(10):
        print(i%8, end = ', ')
0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3,
```

Φυσικά τα υπόλοιπα μπορεί να είναι από 0 έως 7.

Ασκηση 2.11.3 *Αν ένας αριθμός διαιρεθεί δια 9 δίνει πηλίκο 73 και υπόλοιπο 4. Ποιος είναι ο αριθμός;*

```
>>> print(9*73+4)
661
>>> eykleidia(661,73)
661:73=9*73+4
```

Ο αριθμός αυτός είναι $9 \cdot 73 + 4 = 661$.

Ασκηση 2.11.4 *Αν σήμερα είναι Τρίτη, τι μέρα θα είναι μετά από 247 ημέρες;*

Για να λυθεί αυτό το πρόβλημα θα κατασκευάσουμε έναν πίνακα meres:

```
meres = ['ΤΡ','ΤΕ','ΠΕ','ΠΑ','ΣΑ','ΚΥ','ΔΕ']
```

Σε αυτόν meres[0] = 'ΤΡ', meres[1]='ΤΕ' κ.λπ. Αυτό το πρόβλημα προσεγγίζεται ως εξής: 1. Μετά από μία(1) ημέρα θα είναι Τετάρτη(meres[1]) 2. Μετά από δυό(2) ημέρες θα είναι Πέμπτη(meres[2]) 3. Μετά από τρεις(3) ημέρες θα είναι Παρασκευή(meres[3]) 4. Μετά από τέσσερις(4) ημέρες θα είναι Σάββατο(meres[4]) 5. Μετά από πέντε(5) ημέρες θα είναι Κυριακή(meres[5]) 6. Μετά από έξι(6) ημέρες θα είναι Δευτέρα(meres[6]) 7. Μετά από επτά(7) ημέρες θα είναι Τρίτη(meres[0] = meres[7 % 7]), αφού το υπόλοιπο της διαίρεσης του 7 με το 7 είναι 0. 8. Μετά από οκτώ(8) ημέρες θα είναι Τετάρτη(meres[1] = meres[8 % 7]), αφού το υπόλοιπο της διαίρεσης του 8 με το 7 είναι 1.

κατά συνέπεια μετά από 247 ημέρες θα είναι Πέμπτη:

```
>>> meres = ['ΤΡ','ΤΕ','ΠΕ','ΠΑ','ΣΑ','ΚΥ','ΔΕ']
>>> print(meres[247%7])ΠΕ
```

2.12 Χαρακτήρες διαιρετότητας - ΜΚΔ - ΕΚΠ - Ανάλυση αριθμού σε γινόμενο πρώτων παραγόντων

Ασκηση 2.12.1 *Το τοπικό γραφείο της UNICEF θα μοιράσει 150 τετράδια, 90 στυλό και 60 γόμες σε πακέτα δώρων, ώστε τα πακέτα να είναι τα ίδια και να περιέχουν και τα τρία είδη.*

Ασκηση 2.12.2 *1. Μπορεί να γίνουν 10 πακέτα δώρων; Αν ναι, πόσα από κάθε είδος θα έχει κάθε πακέτο;*

Ασκηση 2.12.3 *2. Πόσα πακέτα δώρων μπορεί να γίνουν με όλα τα διαθέσιμα είδη;*

Ασκηση 2.12.4 *3. Πόσα πακέτα δώρων μπορεί να γίνουν με τα λιγότερα δυνατά από κάθε είδος;*

1. Μπορούν να γίνουν 10 πακέτα δώρων, με 15 τετράδια, 9 στυλό και 6 γόμες.

```
>>> 150%10
0
>>> 90%10
0
>>> 60%10
0
```

2. Ακόμη και δύο πακέτα μπορούν να γίνουν με 75 τετράδια, 45 στυλό και 30 γόμες. 3. Για να έχουμε μέσα τα λιγότερα δυνατά είδη θα έχουμε περισσότερα πακέτα και θα πρέπει και οι τρεις αριθμοί να διαιρούν ακριβώς το πλήθος των πακέτων:

```
>>> for i in range(1,60):
>>>     if (150 % i == 0 and 90 % i == 0 and 60 % i == 0):
>>>         print(i)
1
2
3
5
6
10
15
30
```

Ετσι η απάντηση είναι 30 πακέτα με 150:30 = 5 τετράδια, 3 στυλό και 2 γόμες το καθένα.

Το μέγιστο πλήθος των πακέτων που διαιρεί και τους τρεις αριθμούς είναι ο μέγιστος κοινός διαιρέτης τους, ή αλλιώς Μ.Κ.Δ., για να βρεθεί ο Μ.Κ.Δ. δύο αριθμών στην Python 3 μπορεί να χρησιμοποιηθεί το παρακάτω πρόγραμμα:

```
>>> from fractions import gcd
>>> print(gcd(150,90))
30
```

Η συνάρτηση gcd υπολογίζει τον Μ.Κ.Δ. δύο αριθμών.

Για τους τρεις αριθμούς, υπάρχει το εξής πρόβλημα αν τους τοποθετήσετε σαν ορίσματα της συνάρτησης gcd

```
>>> print(gcd(150,90,60))
Traceback (most recent call last):
  File "python", line 1, in <module\begin{exercise}\end{exercise}
TypeError: gcd() takes 2 positional arguments but 3 were given
```

Η φράση *gcd() takes 2 positional arguments but 3 were given* σημαίνει πως δεν μπορούμε να υπολογίσουμε τον Μ.Κ.Δ. βάζοντας όλους τους αριθμούς σαν ορίσματα της συνάρτησης. Ευτυχώς, ο Μ.Κ.Δ. των τριών αριθμών μπορεί να υπολογιστεί ως εξής:

```
>>> mkd150_90 = gcd(150,90)
>>> print(gcd(mkd150_90,60))
30
```

που είναι το ίδιο με τον παρακάτω κώδικα:

```
>>> print(gcd(gcd(150,90),60))
30
```

Ανάλυση σε γινόμενο πρώτων παραγόντων

Το παρακάτω πρόγραμμα αναλύει αριθμούς σε γινόμενο παραγόντων.

```
class ginomenoparagontwn():
  def __init__(self,n):
    self.paragontes = []
    self.dinameis = {}
    if type(n) == int:
      self.arithmos = n
      while n <1:
        for i in range(2,n+1):
          if n%i == 0:
            n = n // i
            self.paragontes.append(i)
            break
      self.paragontes = sorted(self.paragontes)
      for i in self.paragontes:
        if i not in self.dinameis:
          self.dinameis[i] = self.paragontes.count(i)
    elif type(n) == dict:
      self.arithmos = 1
      self.dinameis = n
      for i in n:
        self.arithmos *= i**n[i]
      for i in n:
        self.paragontes.extend([i]*n[i])
      self.paragontes = sorted(self.paragontes)
    if len(self.paragontes) < 1:
      self.einaiPrwtos = False
    else:
      self.einaiPrwtos = True

  def mkd(self,other):
    if type(other) == int:
      other = ginomenoparagontwn(other)
    dinameismkd = {}
    for i in self.dinameis:
      if i in other.dinameis:
        dinameismkd[i] = min(self.dinameis[i],other.dinameis[i])
    if dinameismkd == {}:
      return(ginomenoparagontwn({1:1}))
    else:
      return(ginomenoparagontwn(dinameismkd))

  def ekp(self,other):
    if type(other) == int:
      other = ginomenoparagontwn(other)
    dinameisekp = self.dinameis
    for i in other.dinameis:
```

```
      if i in dinameisekp:
        dinameisekp[i] = max(dinameisekp[i],other.dinameis[i])
      else:
        dinameisekp[i] = other.dinameis[i]
    return(ginomenoparagontwn(dinameisekp))

  def __repr__(self):
    return(str(self.arithmos) + ':' + '*'.join([str(x) + '^' + str(self.dinameis[x])
      for x in self.dinameis]) + ':' + '*'.join([str(x) for x in self.paragontes]) +
      ('πρώτος,' if self.einaiPrwtos else ""))
```

Ασκηση 2.12.5 *Να αναλυθούν οι αριθμοί 2520, 2940, 3780 σε γινόμενο πρώτων παραγόντων.*

```
>>> print(ginomenoparagontwn(2520))
2520:2^3*3^2*5^1*7^1:2*2*2*3*3*5*7
>>> print(ginomenoparagontwn(2940))
2940:2^2*3^1*5^1*7^2:2*2*3*5*7*7
>>> print(ginomenoparagontwn(3780))
3780:2^2*3^3*5^1*7^1:2*2*3*3*3*5*7
```

Επομένως, η ανάλυση σε γινόμενο είναι:

$$2520 = 2^3 \cdot 3^2 \cdot 5 \cdot 7$$

$$2940 = 2^2 \cdot 3 \cdot 5 \cdot 7^2$$

$$3780 = 2^2 \cdot 3^3 \cdot 5 \cdot 7$$

Το ελάχιστο κοινό πολλαπλάσιο (Ε.Κ.Π.) υπολογίζεται ως εξής:

```
>>> ginomenoparagontwn(2520).ekp(2940).ekp(3780)
52920:2^3*3^3*5^1*7^2:2*2*2*3*3*3*5*7*7
```

επομένως Ε.Κ.Π.(2520,2940,3780) = 52920. Ομοίως, ο μέγιστος κοινός διαιρέτης (Μ.Κ.Δ.) υπολογίζεται ως εξής:

```
>>> ginomenoparagontwn(2520).mkd(2940).mkd(3780)
420:2^2*3^1*5^1*7^1:2*2*3*5*7
```

και Μ.Κ.Δ.(2520,2940,3780) = 420.

Ασκηση 2.12.6 *Να βρεθεί αν διαιρούνται οι αριθμοί 12510, 772, 225, 13600 με 2, 3, 4, 5, 8, 9, 10, 25, 100.*

Η φιλοσοφία της άσκησης είναι να λυθεί με τα κριτήρια διαιρετότητας ωστόσο, στον υπολογιστή ο υπολογισμός του υπολοίπου είναι εύκολος με τον τελεστή

```
diairetaios = [12510,772,224,13600]
diairetis = [2,3,4,5,8,9,10,25,100]
print(" "*5,end=',')
for d in diairetis:
    print(str(d).rjust(3),end = ',')
print()
for D in diairetaios:
    print(str(D).rjust(5),end=',')
    for d in diairetis:
        if (D%d==0):
            print('Ναι'.rjust(3),end = ',')
        else:
            print('Όχι'.rjust(3),end = ',')
    print()
```

Το αποτέλεσμα είναι:

```
     ,  2,  3,  4,  5,  8,  9, 10, 25,100,ΝαιΝαίΟχιΝαίΟχιΝαιΝαίΟχίΟχι
12510,,,,,,,,,,
  ΝαίΟχιΝαιΟχιΟχίΟχίΟχιΟχίΟχι772,,,,,,,,,,
  ΝαίΟχιΝαιΟχιΝαίΟχιΟχίΟχίΟχι224,,,,,,,,,,ΝαίΟχιΝαιΝαιΝαίΟχιΝαιΝαιΝαι
13600,,,,,,,,,,
```

Το άθροισμα των ψηφίων ενός αριθμού μπορεί να υπολογιστεί με την εξής εντολή:

```
sum([int(i) for i in str(num)])
```

Για παράδειγμα

```
>>> num = 123
>>> sum([int(i) for i in str(num)])
6
```

Αυτή η εντολή ανήκει στην κατηγορία του list comprehension. Η εναλλακτική είναι το εξής πρόγραμμα:

```
athroisma = 0
for i in str(num):
    athroisma = athroisma + i
```

Το αποτέλεσμά του είναι:

```
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Ο λόγος είναι σε αυτό το i είναι το κάθε ψηφίο αλλά σαν γράμμα της αλφαβήτου του υπολογιστή (αλφαριθμητικό / string). Ετσι το σωστό πρόγραμμα είναι:

```
athroisma = 0
for i in str(num):
    athroisma = athroisma + int(i)
```

που υπολογίζει το άθροισμα των ψηφίων.

Στην python ο τελεστής sum, υπολογίζει το άθροισμα των στοιχείων μιας λίστας όταν αυτή αποτελείται από αριθμούς για παράδειγμα

```
>>> sum([1,2,3])
6
```

Επομένως πρέπει να κατασκευάσουμε μια λίστα στην οποία οι αριθμοί θα είναι τα ψηφία του αρχικού αριθμού num. Η κατασκευή αυτής της λίστας μπορεί να γίνει πάλι με μία for και στη συνέχεια να υπολογιστεί το άθροισμα :

```
lista = []
for i in str(num):
    lista.append(int(i))
sum(lista)
```

Με το list comprehension όμως η δημιουργία της λίστας απλοποιείται, γράφοντας το for μέσα στον πίνακα και γίνεται:

```
lista = [int(i) for i in str(num)]
sum(lista)
```

και τέλος

```
sum([int(i) for i in str(num)])
```

Το άθροισμα των ψηφίων των παραπάνω αριθμών μπορεί να υπολογιστεί με το παρακάτω πρόγραμμα:

```
diairetaios = [12510,772,224,13600]
for num in diairetaios:
    print(num,sum([int(i) for i in str(num)]))
```

και είναι

```
12510 9
772 16
224 8
13600 10
```

Ασκήσεις

Ασκηση 2.12.7 *1. Ισχύει ότι: (100 - 30) - 10 = 100 - (30 - 10)*

```
>>> (100 - 30) - 10 == 100 - (30 - 10)
False
```

Ασκηση 2.12.8 *2. Για να πολλαπλασιάσουμε έναν αριθμό με το 11 τον πολλαπλασιάζουμε με το 10 και προσθέτουμε 1.*

Ισχύει

Ασκηση 2.12.9 *3. Το γινόμενο* $3 \cdot 3 \cdot 3$ *γράφεται* 3^3*.*

```
>>> 3*3*3 == 3**3
True
```

Ασκηση 2.12.10 *4. Το* 2^5 *ισούται με 10.*

```
>>> 2**5 == 10
False
```

Ασκηση 2.12.11 *5.* $\alpha + \alpha + \alpha + \alpha = 4 \cdot \alpha$*.*

Ισχύει

Ασκηση 2.12.12 *6.* $\alpha \cdot \alpha \cdot \alpha \cdot \alpha \cdot \alpha \cdot \alpha = \alpha^5$*.*

Δεν ισχύει

Ασκηση 2.12.13 *7.* $2^3 + 3 = 11$*.*

```
>>> 2**3 + 3 == 11
True
```

Ασκηση 2.12.14 *8.* $3 \cdot 10^2 + 2 \cdot 10^1 + 2 \cdot 10^0 = 322$*.*

```
>>> 3*10**2 + 2*10**1 + 2*10**0 == 322
True
```

Ασκηση 2.12.15 *9.* $20 - 12 : 4 = 2$*.*

```
>>> 20-12/4 == 2
False
```

Ασκηση 2.12.16 *10.* $9 \cdot 3 - 2 + 5 = 30$*.*

```
>>> 9*3-2+5 == 11
False
```

Ασκηση 2.12.17 *11.* $(3 \cdot 1 - 3) : 3 = 0$*.*

```
>>> (3*1-3)/3 == 0
True
```

Ασκηση 2.12.18 *12. Στη σειρά των πράξεων:* $7 + (6 \cdot 5) + 4$*, οι παρενθέσεις δεν χρειάζονται.*

```
>>> 7+(6*5)+4 == 7+6*5+4
True
```

Ασκηση 2.12.19 *13. Η διαφορά δύο περιττών αριθμών είναι πάντα περιττός αριθμός.*

Δεν ισχύει

Ασκηση 2.12.20 *14. Αν ο αριθμός α είναι πολλαπλάσιο του αριθμού β, τότε ο α διαιρείται με το β.*

Ισχύει

Ασκηση 2.12.21 *15. Το 38 είναι πολλαπλάσιο του 2 και του 3.*

```
>>> 38%2
0
>>> 38%3
2
```

Το 38 δεν είναι πολλαπλάσιο του 3

Ασκηση 2.12.22 *16. Ο αριθμός 450 διαιρείται με το 3 και το 9.*

```
>>> 450%3
0
>>> 450%9
0
```

Αρα, ο αριθμός 450 διαιρείται και με το 3 και με το 9.

Ασκηση 2.12.23 *17. Ο 35 και ο 210 έχουν μέγιστο κοινό διαιρέτη τον αριθμό 5.*

```
>>> ginomenoparagontwn(35).mkd(210)
35:5^1*7^1:5*7
```

Δεν ισχύει, έχουν το 35.

Ασκηση 2.12.24 *18. Το ΕΚΠ των 2 και 24 είναι ο αριθμός 48.*

```
ginomenoparagontwn(2).mkd(24)
24:2^3*3^1:2*2*2*3
```

Ασκηση 2.12.25 *19. Η διαίρεση 420 : 15 δίνει υπόλοιπο 18.*

```
>>> 420 % 15
0
```

Δεν ισχύει

Ασκηση 2.12.26 *20. Η σχέση 177 = 5 · 35 + 2 είναι μια ευκλείδια διαίρεση.*

```
>>> 177 == 5 * 35 + 2
True
```

και επίσης, το υπόλοιπο (2) είναι μικρότερο του διαιρέτη 5.

Ασκηση 2.12.27 *21. Ο αριθμός 3 · α + 9 διαιρείται με το 3.*

$$\frac{3 \cdot \alpha + 9}{3} = \alpha + 3$$

Οπότε αν ο α είναι ακέραιος τότε το $3 \cdot \alpha + 9$ διαρείται με το 3.

Ασκηση 2.12.28 *22. Ο αριθμός 300 αναλύεται σε γινόμενο πρώτων παραγόντων ως $3 \cdot 10^2$.*

```
>>> ginomenoparagontwn(300)
300:2^2*3^1*5^2:2*2*3*5*5
```

Ασκηση 2.12.29 *23. Ο αριθμός 224 διαιρείται με το 4 και το 8.*

```
>>> 224%4
0
>>> 224%8
0
```

Ισχύει!

Κεφάλαιο 3

Η έννοια του κλάσματος

3.1 Εισαγωγή

Ασκηση 3.1.1 *Ενα βράδυ τρεις φίλοι αγοράζουν μια πίτσα και την χωρίζουν σε οκτώ κομμάτια. Ο ένας έφαγε το ένα, ο δεύτερος τα τρία και ο τρίτος δύο κομμάτια από αυτά που περίσσεψαν.*

1. Μπορείς να βρεις το μέρος της πίτσας που έφαγε ο καθένας;

2. Τι μέρος της πίτσας περίσσεψε;

Ο πρώτος έφαγε το $\frac{1}{8}$ ο δεύτερος τα $\frac{3}{8}$ και ο τρίτος τα $\frac{2}{8}$. Επομένως και οι τρεις μαζί έφαγαν $1 + 3 + 2 = 6$ κομμάτια, δηλαδή τα $\frac{6}{8}$ της πίτσας. Αρα περίσσεψαν τα υπόλοιπα δύο κομμάτια από τα οκτώ, δηλαδή τα $\frac{2}{8}$ της πίτσας.

Πώς μπορούν να γίνουν πράξεις με κλάσματα στην python; Με τις γνώσεις που ήδη έχουμε εκτελούμε τις παρακάτω εντολές:

```
>>> 1/8
0.125
>>> 3/8
0.375
>>> 2/8
0.25
```

Θυμηθείτε ότι η python χρησιμοποιεί την τελεία για τους δεκαδικούς! Τι μέρος της πίτσας περίσσεψε;

```
>>> 1 - 1/8 - 3/8 - 2/8
0.25
```

Ομως, υπάρχει τρόπος η python να υπολογίζει κλάσματα. Απλά θα πρέπει να εισαχθεί το κατάλληλο module. Στην python υπάρχει διαθέσιμο τέτοιο module και ονομάζεται fractions. Για το δεύτερο ερώτημα λοιπόν μπορούμε να κάνουμε το εξής:

```
>>> import fractions
>>> prwtos = fractions.Fraction(1,8)
>>> deuteros = fractions.Fraction(3,8)
>>> tritos = fractions.Fraction(2,8)
>>> 1 - prwtos - deuteros - tritos
Fraction(1, 4)
```

Ετσι η python υπολογίζει το αποτέλεσμα με τη μορφή κλάσματος και Fraction(1,4) σημαίνει $\frac{1}{4}$.

Παρατηρείτε ότι γράφουμε το όνομα του module στη συνέχεια την τελεία "." και τέλος το Fraction. Με αυτόν τον τρόπο καλούμε κάτι που υπάρχει μέσα στο module. Υπάρχει όμως και ένας πιο εύκολος τρόπος για να εισάγουμε μόνο τις λειτουργίες που θέλουμε από ένα module και να τις καλούμε.

```
>>> from fractions import Fraction
>>> prwtos = Fraction(1,8)
>>> deuteros = Fraction(3,8)
>>> tritos = Fraction(2,8)
>>> 1 - prwtos - deuteros - tritos
Fraction(1, 4)
```

Χρειάζεται προσοχή μόνο αν υπάρχουν περισσότερα από ένα module που πιθανόν να έχουν την ίδια λειτουργία κάτι που δεν ισχύει σε αυτήν την περίπτωση.

Ασκηση 3.1.2 *Μια σοκολάτα ζυγίζει 120 gr και έχει 6 ίσα κομμάτια. (α) Ποιο μέρος της σοκολάτας είναι το κάθε κομμάτι; (β) Πόσα κομμάτια πρέπει να κόψουμε για να πάρουμε 40 gr;*

Το μέρος της σοκολάτας είναι $\frac{1}{6}$ για να βρούμε πόσα κομμάτια χρειαζόμαστε για 40gr θα πρέπει να βρούμε το βάρος του κομματιού που είναι $\frac{1}{6} \cdot 120$. Τέλος, τα κομμάτια που πρέπει να κόψουμε προκύπτουν από τη διαίρεση των 40 γραμμαρίων με το βάρος του κάθε κομματιού.

```
from fractions import Fraction

sok = Fraction(1,6)
print(sok)
baroskommatiou = sok * 120
kommatia = 40 / baroskommatiou
print('θα κόψω ' + str(kommatia) + ' κομμάτια!')
```

το αποτέλεσμα θα είναι το εξής:

```
1/60α
 κόψω 2 κομμάτια!
```

Ασκηση 3.1.3 *Το καμπαναριό μιας εκκλησίας έχει ύψος 20 m, ενώ η εκκλησία έχει ύψος τα Εικόνα του ύψους του καμπαναριού. Ποιο είναι το ύψος της εκκλησίας;*

```
from fractions import Fraction

>>> kampanario = Fraction(3,5)*20
>>> print(kampanario)
12
```

Αρα το ύψος της εκκλησίας είναι 12m.

Στη συνέχεια η εντολή from fractions import Fraction θα υποννοείται ώστε να μην επαναλαμβάνεται συνεχώς. Αν τυχόν την ξεχάσετε το αποτέλεσμα θα έχει το εξής σφάλμα:

```
NameError: name 'Fraction' is not defined
```

Ασκηση 3.1.4 *Μια δεξαμενή πετρελαίου σε μια πολυκατοικία, χωράει 2000 lt. Ο διαχειριστής σε μια μέτρηση βρήκε ότι ήταν γεμάτη κατά τα $\frac{3}{4}$. Πόσα λίτρα πετρέλαιο είχε η δεξαμενή;*

```
>>> dexameni = Fraction(3,4)*2000
>>> print(dexameni)
1500
```

Η δεξαμενή έχει 1500 lt.

Ασκηση 3.1.5 *Τα $\frac{3}{5}$ του κιλού τυρί κοστίζουν 27 €.*
Πόσο κοστίζουν τα $\frac{8}{9}$ του κιλού;

```
>>> enaPempto = Fraction(27,3)
>>> tyri = 5*enaPempto
>>> oktwEnata = Fraction(8,9)*tyri
>>> print(oktwEnata)
40
```

Τα $\frac{8}{9}$ του τυριού κοστίζουν 40 ευρώ.

3.2 Ασκήσεις

Ασκηση 3.2.1 *Είναι τα κλάσματα $\frac{3}{4}$, $\frac{2}{3}$, $\frac{7}{9}$, $\frac{10}{9}$, $\frac{18}{20}$ όλα μικρότερα της μονάδας:*

```
>>> print(Fraction(3,4)<1)
True
>>> print(Fraction(2,3)<1)
True
>>> print(Fraction(7,9)<1)
True
>>> print(Fraction(10,9)<1)
False
>>> print(Fraction(18,20)<1)
True
```

Ασκηση 3.2.2 *Τι κλάσμα των μαθητών της τάξης 28 μαθητών είναι οι 4 απόντες;*

```
>>> print(Fraction(4,28))
1/7
```

Παρατηρούμε ότι η εντολή Fraction(4,28) κάνει απλοποίηση κλάσματος.

> Αν το $\frac{1}{5}$ ενός κιλού καρύδια είναι 14 καρύδια, το κιλό περιέχει 70 καρύδια;

```
>>> print(14*5 == 70)
```

Ναι.

Ασκηση 3.2.3 *Βρες ποιο μέρος του κιλού είναι τα: (α) 100, (β) 250, (γ) 500, (δ) 600 γραμμάρια.*

```
>>> print(Fraction(100,1000))
1/10
>>> print(Fraction(250,1000))
1/4
>>> print(Fraction(500,1000))
1/2
>>> print(Fraction(600,1000)).
3/5
```

Ασκηση 3.2.4 *Ποιο μέρος: (α) του μήνα, (β) του εξαμήνου, (γ) του έτους είναι οι 15 ημέρες;*

```
>>> print(Fraction(15,30))
1/2
>>> print(Fraction(15,180))
1/12
>>> print(Fraction(15,365))
3/73
```

Ασκηση 3.2.5 *Ενα κατάστημα κάνει έκπτωση στα είδη του ίση με τα $\frac{2}{5}$ της αρχικής τιμής τους. Ενα φόρεμα κόστιζε 90 € πριν την έκπτωση. Υπολόγισε πόσα ευρώ έκπτωση έγινε στο φόρεμα και πόσο θα πληρώσουμε για να το αγοράσουμε.*

```
>>> ekpt = Fraction(2,5)*90
>>> print("Η έκπτωση είναι: " + str(ekpt) + " ευρώ!")Η
 έκπτωση είναι: 36 ευρώ!
>>> plir = 90 - ekpt
>>> print("Θα πληρώσουμε: " + str(plir) + " ευρώ!")Θα
 πληρώσουμε: 54 ευρώ!
```

Ασκηση 3.2.6 *Σε μία τάξη τα $\frac{3}{8}$ των μαθητών μαθαίνουν αγγλικά. Να βρεις πόσους μαθητές έχει η τάξη, αν γνωρίζεις ότι αυτοί που μαθαίνουν αγγλικά είναι 12 μαθητές.*

```
>>> print(Fraction(8,3)*12)
32
```

Ασκηση 3.2.7 *Σε ένα ορθογώνιο παραλληλόγραμμο η μια πλευρά του είναι 33 εκατοστά και η άλλη τα $\frac{3}{11}$ της πρώτης. Να βρεις την περίμετρο του ορθογωνίου.*

```
>>> plevra1 = 33
>>> plevra2 = Fraction(3,11)*plevra1
>>> perimetros = 2*(plevra1 + plevra2)
>>> print(perimetros)
84
```

3.3 Ισοδύναμα κλάσματα

Ασκηση 3.3.1 *Να εξετάσετε αν τα κλάσματα: α) $\frac{3}{5}$ και $\frac{10}{14}$ β) $\frac{3}{8}$ και $\frac{18}{48}$ είναι ισοδύναμα.*

```
print(Fraction(3,5) == Fraction(10,14))
False
print(Fraction(3,8) == Fraction(18,48))
True
```

Ετσι, τα $\frac{3}{5}$ και $\frac{10}{14}$ δεν είναι ισοδύναμα ενώ τα $\frac{3}{8}$ και $\frac{18}{48}$ είναι.

Ασκηση 3.3.2 *Να απλοποιηθεί το κλάσμα $\frac{30}{66}$*

```
>>> print(Fraction(30,66))
5/11
```

Ασκηση 3.3.3 *Να μετατραπούν σε ομώνυμα τα κλάσματα* $\frac{3}{5}$, $\frac{2}{3}$ *και* $\frac{5}{20}$*:*

Επειδή η Fraction κάνει απλοποίηση σε ανάγωγο κλάσμα δεν μπορούμε να επιλέξουμε παρονομαστή, γι' αυτό θα κατασκευάσετε μια συνάρτηση η οποία θα τυπώνει το κλάσμα επιλέγοντας τον παρονομαστή.

```
def tiposemeparonomasti(k,p):
  """
  tiposemeparonomasti(k,p)
  Τύπωσε το κλάσμα k με παρονομαστή p
  """
  if p % k.denominator == 0:#αν το p είναι πολλαπλάσιο
                            #του τρέχοντος παρονομαστή (denominator)
                            #τότε μπορούμε να πολλαπλασιάσουμε
                            #όλο το κλάσμα με έναν συντελεστή
    synt = int(p // k.denominator)
    print(str(k.numerator * synt) + '/' + str(k.denominator * synt))
  else:#αν το p δεν είναι πολλαπλάσιο του τρέχοντος παρονομαστή
       #τότε τυπώνουμε το κλάσμα ως έχει
    print(k)
```

Το k.denominator είναι ο παρονομαστής του κλάσματος k.

Αφού φτιάξετε τη συνάρτηση tiposemeparonomasti δοκιμάστε:

```
>>> tiposemeparonomasti(Fraction(3,4),12)
9/12
>>> tiposemeparonomasti(Fraction(1,2),20)
10/20
```

Για να κάνουμε ομώνυμα τα κλάσματα βρίσκουμε το Ελάχιστο Κοινό Πολλαπλάσιο (Ε.Κ.Π.) των παρονομαστών. Η συνάρτηση για το Ε.Κ.Π. είναι η παρακάτω, αφού το Ε.Κ.Π. δύο αριθμών προκύπτει από το γινόμενο τους αφού το διαιρέσουμε με τον μέγιστο κοινό διαιρέτη (Μ.Κ.Δ. - G.C.D.). Μάλιστα η βιβλιοθήκη fractions περιέχει τη συνάρτηση gcd που υπολογίζει το Μ.Κ.Δ. οπότε:

```
from fractions import gcd
def ekp(a,b):
  return(a*b/gcd(a,b))
```

Τέλος συνδυάζοντας τα προηγούμενα το συνολικό πρόγραμμα για να κάνουμε ομώνυμα τα κλάσματα $\frac{3}{5}$, $\frac{2}{3}$ και $\frac{5}{20}$ είναι:

```
from fractions import Fraction,gcd
def tiposemeparonomasti(k,p):
  """
```

```
  tiposemeparonomasti(k,p)
  Τύπωσε το κλάσμα k με παρονομαστή p
  """
  if p % k.denominator == 0:#αν το p είναι πολλαπλάσιο
                            #του τρέχοντος παρονομαστή (denominator)
                            #τότε μπορούμε να πολλαπλασιάσουμε
                            #όλο το κλάσμα με έναν συντελεστή
    synt = int(p // k.denominator)
    print(str(k.numerator * synt) + '/' + str(k.denominator * synt))
  else:#αν το p δεν είναι πολλαπλάσιο του τρέχοντος παρονομαστή
       #τότε τυπώνουμε το κλάσμα ως έχει
    print(k)

def ekp(a,b):
  return(a*b/gcd(a,b))

a = Fraction(3,5)
b = Fraction(2,3)
c = Fraction(5,20)

koinos = ekp(a.denominator,ekp(b.denominator,c.denominator))
tiposemeparonomasti(a,koinos)
tiposemeparonomasti(b,koinos)
tiposemeparonomasti(c,koinos)
```

Μπορούμε να εισάγουμε δύο λειτουργίες από την ίδια βιβλιοθήκη χωρίζοντάς τες με κόμμα ”,”. Θυμηθείτε ότι σε ένα κλάσμα a το a.denominator είναι ο παρονομαστής.

Το αποτέλεσμα του προγράμματος είναι:

```
36/60
40/60
15/60
```

Ασκηση 3.3.4 *Να εξετάσετε ποια από τα παρακάτω κλάσματα είναι ισοδύναμα:*

(α) $\frac{2}{3}, \frac{18}{27}$,
(β) $\frac{3}{4}, \frac{1}{2}$,
(γ) $\frac{7}{8}, \frac{30}{40}$,
(δ) $\frac{13}{14}, \frac{26}{28}$.

```
>>> print(Fraction(2,3)==Fraction(18,27))
True
>>> print(Fraction(3,4)==Fraction(1,2))
False
>>> print(Fraction(7,8)==Fraction(30,40))
False
```

```
>>> print(Fraction(13,14)==Fraction(26,28))
True
```

Ασκηση 3.3.5 *Να μετατρέψεις καθένα από τα παρακάτω κλάσματα σε ισοδύναμο κλάσμα με παρονομαστή τον αριθμό 100:*

(α) $\frac{3}{4}$
(β) $\frac{8}{5}$
(γ) $\frac{4}{20}$
(δ) $\frac{5}{2}$
(ε) $\frac{60}{75}$

Μπορούμε να χρησιμοποιήσουμε την συνάρτηση tiposemeparonomasti

```
>>> tiposemeparonomasti(Fraction(3,4),100)
75/100
>>> tiposemeparonomasti(Fraction(8,5),100)
160/100
>>> tiposemeparonomasti(Fraction(4,20),100)
20/100
>>> tiposemeparonomasti(Fraction(5,2),100)
250/100
>>> tiposemeparonomasti(Fraction(60,75),100)
80/100
```

Ασκηση 3.3.6 *Να μετατρέψεις τα παρακάτω κλάσματα σε ισοδύναμα με παρονομαστή τον αριθμό 3:*

```
>>> tiposemeparonomasti(Fraction(10,6),3)
5/3
>>> tiposemeparonomasti(Fraction(50,30),3)
5/3
>>> tiposemeparonomasti(Fraction(18,27),3)
2/3
```

Ασκηση 3.3.7 *Να μετατρέψεις το κλάσμα* $\frac{2}{3}$ *σε ισοδύναμο κλάσμα με παρονομαστή:(α) 6, και (β) 15.*

```
>>> tiposemeparonomasti(Fraction(2,3),6)
4/6
>>> tiposemeparonomasti(Fraction(2,3),15)
10/15
```

Ασκηση 3.3.8 *Να απλοποιήσεις τα κλάσματα: (α)* $\frac{25}{30}$ *(β)* $\frac{12}{9}$ *(γ)* $\frac{32}{56}$

```
>>> print(Fraction(25,30))
5/6
>>> print(Fraction(12,9))
4/3
>>> print(Fraction(32,56))
4/7
```

3.4 Πρόσθεση και αφαίρεση κλασμάτων

Ασκηση 3.4.1 *Το συνεργείο του Δήμου φύτεψε σε μια μέρα τα $\frac{4}{12}$ μιας πλατείας με λουλούδια. Την επόμενη ήμερα που ο καιρός δεν ήταν καλός φύτεψε μόνο τα $\frac{3}{12}$ της πλατείας. Ποιο τμήμα της πλατείας είχε φυτέψει, συνολικά, στο τέλος της δεύτερης ημέρας;*

```
>>> print(Fraction(4,12) + Fraction(3,12)(
7/12
```

Ασκηση 3.4.2 *Ενα φορτηγό κάλυψε σε μία ώρα τα 2/5 της διαδρομής Πάτρα - Τρίπολη. Ποιο μέρος της διαδρομής του μένει να καλύψει ακόμη;*

```
>>> print(1-Fraction(2,5))
3/5
```

Ασκηση 3.4.3 *Μια βρύση γεμίζει, σε 1 ώρα, τα $\frac{2}{5}$ της δεξαμενής. Μια άλλη βρύση γεμίζει το $\frac{1}{3}$ της ίδιας δεξαμενής, επίσης σε 1 ώρα. Αν και οι δύο βρύσες τρέχουν ταυτόχρονα μέσα στη δεξαμενή, τι μέρος της δεξαμενής θα γεμίσουν σε 1 ώρα;*

```
>>> print(Fraction(2,5)+Fraction(1,3))
11/15
```

Ασκηση 3.4.4 *Να υπολογισθεί το άθροισμα*

$$\frac{1}{4} + \frac{2}{4} + 3$$

```
>>> print(Fraction(1,4)+Fraction(2,4) + 3)
15/4
```

Ασκηση 3.4.5 *Να υπολογισθεί η διαφορά και το άθροισμα των κλασμάτων $\frac{3}{12}$ και $\frac{7}{20}$.*

```
>>> print(Fraction(3,12) + Fraction(7,20))
3/5
>>> print(Fraction(7,20) – Fraction(3,12))
1/10
```

Για να τυπώσουμε ένα κλάσμα ως μεικτό θα εφαρμόσουμε τα εξής βήματα: 1. Βρίσκουμε το ακέραιο μέρος της διαίρεσης του αριθμητή με τον παρονομαστή έστω μ. 2. Αν το μ είναι ο τυπώνουμε το κλάσμα ως έχει (είναι μικρότερο της μονάδας), αλλιώς τυπώνουμε το μ και στη συνέχεια το κλάσμα που προκύπτει αν από το αρχικό κλάσμα αφαίρεσουμε το μ.

```
def tiposemikto(k):
  m = k.numerator // k. denominator
  if m == 0:
        print(k)
  else:
        print(str(m) + " " + str(k–m))
```

Για παράδειγμα

```
>>> tiposemikto(Fraction(15,4))
3 3/4
>>> tiposemikto(Fraction(5,2))
2 1/2
>>> tiposemikto(Fraction(38,12))
```

3.5 Σύγκριση κλασμάτων

Ασκηση 3.5.1 *Η Μαρία είπε πως το ροζ χρώμα καταλαμβάνει τα $\frac{9}{48}$, το γαλάζιο τα $\frac{10}{48}$ και το πράσινο τα $\frac{7}{48}$. Ενώ ο Γιάννης είπε ότι το ροζ είναι τα $\frac{3}{16}$, το γαλάζιο τα $\frac{5}{24}$ και το πράσινο το $\frac{1}{8}$ του τετραγώνου. Ποιος έχει δίκιο και ποιος όχι;*

```
>>> roz = Fraction(9,48)
>>> print(roz)
3/16
```

Αρα και η Μαρία και ο Γιάννης έχουν δίκο όσον αφορά το ροζ χρώμα.

```
>>> galazio = Fraction(10,48)
>>> print(galazio)
5/24
```

Αρα και η Μαρία και ο Γιάννης έχουν δίκο όσον αφορά το γαλάζιο χρώμα.

Τέλος, όσον αφορά το πράσινο προκύπτει ότι:

```
>>> prasino = Fraction(7,48)
>>> print(prasino)
```

```
7/48
>>> Fraction(7,48) == Fraction(1,8)
False
>>> Fraction(7,48) > Fraction(1,8)
True
```

Δηλαδή, το $\frac{7}{48}$ είναι μεγαλύτερο από το $\frac{1}{8}$.

Παραδείγματα

Ασκηση 3.5.2 *Να συγκριθούν τα κλάσματα* $\frac{7}{10}$ *και* $\frac{7}{15}$

```
>>> Fraction(7,10) > Fraction(7,15)
True
```

Αρα, $\frac{7}{10} > \frac{7}{15}$.

Ασκηση 3.5.3 *Να συγκριθούν τα κλάσματα:* $\frac{5}{8}$ *και* $\frac{4}{9}$.

```
>>> Fraction(5,8) > Fraction(4,9)
True
```

Δεν χρειάζεται να τα μετατρέψουμε σε ομώνυμα, η python λύνει το πρόβλημα της σύγκρισης με τον δικό της τρόπο.

Ασκηση 3.5.4 *Σύγκρινε τα κλάσματα (α)* $\frac{3}{7}$ *και* $\frac{5}{7}$, *(β)* $\frac{3}{5}$ *και* $\frac{3}{9}$ *και (γ)* $\frac{4}{5}$ *και* $\frac{8}{12}$.

```
>>> Fraction(3,7) < Fraction(5,7)
True
>>> Fraction(3,5) > Fraction(3,9)
True
>>> Fraction(4,5) > Fraction(8,12)
True
```

Ασκηση 3.5.5 *Βάλε σε σειρά τα κλάσματα* $\frac{3}{5}$, $\frac{8}{15}$, $\frac{5}{10}$, $\frac{20}{15}$, $\frac{7}{5}$

```
>>> lista = [Fraction(31,10),Fraction(8,15),Fraction(5,10),
Fraction(20,15),Fraction(7,5)]
>>> print(",".join([str(x) for x in sorted(lista)]))
1/2,8/15,4/3,7/5,31/10
```

3.6 Πολλαπλασιασμός κλασμάτων

Ασκηση 3.6.1 *Να βρεθεί το γινόμενο* $\frac{3}{7} \cdot \frac{70}{6} \cdot \frac{8}{5}$

```
>>> Fraction(3,7)*Fraction(70,6)*Fraction(8,5)
8
```

Ασκηση 3.6.2 *Σε ένα σχολείο με 252 μαθητές τα* $\frac{5}{9}$ *είναι αγόρια. Πόσα είναι τα αγόρια και πόσα είναι τα κορίτσια;*

```
>>> agoria,koritsia = 252*Fraction(5,9),252-252*Fraction(5,9)
>>> print(agoria)
140
>>> print(koritsia)
112
```

Ασκηση 3.6.3 *Υπολόγισε τα γινόμενα* $3 \cdot \frac{3}{4}$, $7 \cdot \frac{10}{14}$, $\frac{4}{2} \cdot 2$, $\frac{5}{100} \cdot 10$

```
>>> x = 3*Fraction(3,4)
>>> print(x)
9/4
>>> x = 7*Fraction(10,14)
>>> print(x)
5
>>> x = Fraction(4,2)*2
>>> print(x)
4
>>> x = Fraction(5,100)*10
>>> print(x)
1/2
```

Ασκηση 3.6.4 *Βρες τα γινόμενα* $\frac{2}{5} \cdot \frac{7}{8}$, $\frac{8}{10} \cdot \frac{100}{5}$, $\frac{4}{9} \cdot \frac{5}{9}$, $\frac{3}{2} \cdot \frac{2}{15}$

```
>>> x = Fraction(2,5)*Fraction(7,8)
>>> print(x)
7/20
>>> x = Fraction(8,10)*Fraction(100,5)
>>> print(x)
16
>>> x = Fraction(4,9)*Fraction(5,9)
>>> print(x)
20/81
>>> x = Fraction(3,2)*Fraction(2,15)
>>> print(x)
1/5
```

	$\frac{5}{7}$	$\frac{3}{2}$	1	$\frac{3}{4}$
$\frac{7}{5}$				
$\frac{2}{3}$				
1				
$\frac{4}{3}$				

	$\frac{5}{7}$	$\frac{3}{2}$	1	$\frac{3}{4}$
$\frac{7}{5}$	1	$\frac{21}{10}$	$\frac{7}{5}$	$\frac{21}{20}$
$\frac{2}{3}$	$\frac{10}{21}$	1	$\frac{2}{3}$	$\frac{1}{2}$
1	$\frac{5}{7}$	$\frac{3}{2}$	1	$\frac{3}{4}$
$\frac{4}{3}$	$\frac{20}{21}$	2	$\frac{4}{3}$	1

Ασκηση 3.6.5 *Συμπλήρωσε τον πίνακα:*

Ο πίνακας αυτός μπορεί να εμφανιστεί με το παρακάτω πρόγραμμα:

```
from fractions import Fraction

ori = [Fraction(5,7),Fraction(3,2),1,Fraction(3,4)]
kat = [Fraction(7,5),Fraction(2,3),1,Fraction(4,3)]
print(" "*5+", ",end=" ")
for orizontio in ori:
    print(str(orizontio).rjust(5),end=", ")
print()
print("-"*35)
for katheto in kat:
    print(str(katheto).rjust(5),end=", |")
    for orizontio in ori:
        print(str(orizontio*katheto).rjust(5),end=", ")
    print()
```

και το αποτέλεσμα του προγράμματος είναι:

Ασκηση 3.6.6 *Υπολόγισε τα γινόμενα* $2\frac{1}{3} \cdot \frac{3}{21}$, $4\frac{1}{5} \cdot 2\frac{1}{2}$, $3\frac{1}{8} \cdot 10$, $1\frac{2}{3} \cdot \frac{3}{2}$

```
x = (2+Fraction(1,3))*Fraction(3,21)
print(x)

x = (4+Fraction(1,5))*(2+Fraction(1,2))
print(x)

x = (3+Fraction(1,8))*10
print(x)

x = (1+Fraction(2,3))*Fraction(3,2)
print(x)
```

και το αποτέλεσμα είναι

```
1/3
21/2
125/4
5/2
```

Ασκηση 3.6.7 *Να βρεις τους αντίστροφους των αριθμών* $\frac{4}{7}$, 72, $\frac{5}{8}$, $\frac{1}{3}$, $\frac{739}{8}$, 1

```
print(1/Fraction(4,7))
print(1/Fraction(72))
print(1/Fraction(5,8))
print(1/Fraction(1,3))
print(1/Fraction(739,8))
print(1/Fraction(1))
```

και το αποτέλεσμα είναι:

```
7/4
1/72
8/53
8/739
1
```

Ασκηση 3.6.8 *Ο Κώστας ήπιε τα* $\frac{2}{3}$ *από ένα μπουκάλι, που περιείχε αναψυκτικό όγκου* $1\frac{1}{2}$ *του λίτρου. Πόσα λίτρα αναψυκτικού ήπιε;*

```
print(Fraction(2,3)*(1+Fraction(1,2)))
```

που δίνει αποτέλεσμα

```
1
```

Ο Κώστας ήπιε 1 λίτρο αναψυκτικού.

Ασκηση 3.6.9 *Υπολόγισε τα εξαγόμενα των πράξεων* $\frac{6}{5} + \frac{3}{5} \cdot \frac{1}{4}$, $(\frac{6}{5} + \frac{3}{5}) \cdot \frac{1}{4}$, $\frac{6}{5} - \frac{3}{5} \cdot \frac{1}{4}$

```
x = Fraction(6,5) + Fraction(3,5)*Fraction(1,4)
print(x)

x = (Fraction(6,5)+Fraction(3,5))*Fraction(1,4)
print(x)

x = (Fraction(6,5)-Fraction(3,5))*Fraction(1,4)
print(x)
```

Το αποτέλεσμα είναι το εξής:

```
27/20
9/20
3/20
```

Ασκηση 3.6.10 *Ομοια* $(\frac{7}{3}+\frac{2}{15})\cdot\frac{3}{8}$, $(\frac{7}{3}+\frac{2}{15})\cdot\frac{3}{8}$, $\frac{7}{3}-\frac{2}{15}\cdot\frac{3}{8}$

```
x = (Fraction(7,3)+Fraction(2,15))*Fraction(3,8)
print(x)

x = (Fraction(7,3)-Fraction(2,15))*Fraction(3,8)
print(x)

x = Fraction(7,3)-Fraction(2,15)*Fraction(3,8)
print(x)
```

και το αποτέλεσμα είναι:

```
37/40
33/40
137/60
```

Κεφάλαιο 4

Δεκαδικοί αριθμοί

4.1 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 4.1.1 *(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 4.1.2 *(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 4.1.3 *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 4.1.4 *(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 4.1.5 *(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 4.1.6 *(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 4.1.7 *(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 4.1.8 *(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 4.1.9 *(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 4.1.10 *(Στο βιβλίο βρίσκεται στη Σελ. 61) Η περίμετρος ενός τετραγώνου είναι 20,2. Να υπολογίσεις την πλευρά του.*

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

Ασκηση 4.1.11 *(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 4.1.12 *(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 4.1.13 *(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 4.1.14 *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.

4.2 Τυποποιημένη μορφή μεγάλων αριθμών

Η μάζα του Ηλιου είναι 1983000000000000000000000000000 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 4.2.1 *(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 4.2.2 *(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 4.2.3 *(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
print(str(telikiWra ) + ':' + str(telikaLepta))
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
print(str(telikiWra ) + ':' + str(telikaLepta))
if telikiWra > 12:
    print(str(telikiWra ) + ':' + str(telikaLepta –12)+ ' μμ..')
else:
    print(str(telikiWra)+':'+str(telikaLepta) + 'πμ..')
```