Μαθηματικά Α΄ Γυμνασίου

Μέρος Α - Κεφάλαιο 1ο - Φυσικοί Αριθμοί

Μια πρώτη γνωριμία

Το βιβλίο μαθηματικών της Α΄ Γυμνασίου αρχίζει στη σελίδα 11 με το εξής πρόβλημα:

> Διάλεξε έναν τριψήφιο αριθμό. Βρες όλους τους διαφορετικούς τριψήφιους αριθμούς που προκύπτουν όταν εναλλάξεις τα ψηφία του αριθμού που διάλεξες και γράψε αυτούς με όλους τους δυνατούς τρόπους.
>
> 1. Ποιος είναι ο μικρότερος και ποιος ο μεγαλύτερος;
> 2. Γράψε όλους τους αριθμούς που βρήκες με σειρά αύξουσα, δηλαδή από το μικρότερο προς το μεγαλύτερο.
> 3. Στη συνέχεια, γράψε τους ίδιους αριθμούς με φθίνουσα σειρά.

Η λύση του προβλήματος σε Python είναι η παρακάτω (κατεβάστε τη):

```
import itertools

x = input('Διάλεξε έναν τριψήφιο αριθμό:')
x = int(x)
if x>=100 and x<=999:
    psifio0 = x%10
    x = x - psifio0
    x = x // 10
    psifio1 = x%10
    x = x - psifio1
    x = x // 10
    psifio2 = x

    enallages = []
    for enallagi in itertools.permutations([psifio0,psifio1,psifio2]):
        arithmos = 100*enallagi[2]+10*enallagi[1]+enallagi[0]
        if arithmos not in enallages:
            enallages.append(arithmos)
    print('Ο μεγαλύτερος αριθμός είναι: ' + str(max(enallages)))
    print('Ο μικρότερος αριθμός είναι: ' + str(min(enallages)))
    print('Όλοι οι αριθμοί με αύξουσα σειρά:')
    for arithmos in sorted(enallages):
        print(arithmos)
    print('Όλοι οι αριθμοί με φθίνουσα σειρά:')
    for arithmos in sorted(enallages,reverse = True):
        print(arithmos)
else:
    print('Δεν δώσατε τριψήφιο αριθμό')
```

Δεν χρειάζεται να καταλάβετε από τώρα το πρόγραμμα αυτό, θα μελετήσουμε τα στοιχεία του αργότερα.

Για να εκτελέσετε το πρόγραμμα αυτό θα πρέπει να έχετε εγκαταστήσει τη γλώσσα προγραμματισμού Python στον υπολογιστή σας και συγκεκριμένα την Python 3.

Αν την έχετε εγκαταστήσει τότε μπορείτε να εκτελέσετε το πρόγραμμα με

```
python MathASel11Dras1.py
```

Το αποτέλεσμα αν πληκτρολογήσουμε τον αριθμό 568 θα είναι:

```
Διάλεξε έναν τριψήφιο αριθμό:568
Ο μεγαλύτερος αριθμός είναι: 865
Ο μικρότερος αριθμός είναι: 568
Όλοι οι αριθμοί με αύξουσα σειρά:
568
586
658
685
856
865
Όλοι οι αριθμοί με φθίνουσα σειρά:
865
856
685
658
586
568
```

Μπορείτε να ξαναεκτελέσετε την εντολή και να πληκτρολογήσετε έναν άλλο αριθμό για να πάρετε διαφορετικά αποτελέσματα.

Οι φυσικοί αριθμοί

Η Python μπορεί να χειριστεί φυσικούς αριθμούς. Για παράδειγμα αν γράψουμε έναν αριθμό στην Python, τον επαναλαμβάνει.

```
>>> 123
123
```

Μπορούμε να δούμε και τον τύπο του, δηλαδή τι είδους αντικείμενο είναι ως εξής:

```
>>> type(123)
<class 'int'>
```

Η Python μας λέει πως ο αριθμός 123 ανήκει στην κλάση int. Η λέξη int προκύπτει από το integer που είναι ο αγγλικός όρος για τον ακέραιο αριθμό. Έτσι το 123 για την Python είναι ένας ακέραιος αριθμός.

Σύγκριση 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*.

```
>>> 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 όταν μία πρόταση ισχύει και False όταν δεν ισχύει.

Άρτιοι και περιττοί

Άρτιοι λέγονται οι αριθμοί που διαιρούνται ακριβώς με το 2 και περιττοί αυτοί που δεν διαιρούνται ακριβώς με το 2. Το υπόλοιπο της διαίρεσης δύο αριθμών στην Python μπορεί να υπολογιστεί με τον τελεστή %. Για παράδειγμα 3%2 μας κάνει 1 αφού το υπόλοιπο της διαίρεσης του 3 με το 2 είναι 1.

```
>>> 3%2
1
```

Ένας αριθμός x μπορεί να είναι άρτιος ή περιττός. Αν x%2 == 0 τότε ο αριθμός είναι άρτιος και σε αντίθετη περίπτωση είναι περιττός. Για παράδειγμα:

```
>>> 65536%2
0
>>> 65537%2
1
```

Ο αριθμός 65536 είναι άρτιος και ο αριθμός 65537 είναι περιττός. Αν χρησιμοποιήσουμε το όνομα x για τον αριθμό που θέλουμε να ελέγξουμε αν είναι άρτιος τότε η εντολή x%2 == 0 μπορεί να μας υποδείξει αν ο αριθμός είναι άρτιος ή όχι. Δίνουμε όνομα σε έναν αριθμό ως εξής: x = και στη συνέχεια τον αριθμό.

```
>>> x = 65536
>>> x%2 == 0
True
>>> x = 65537
>>> x%2 == 0
False
```

Αφού μια εντολή μας δείχνει αν ο αριθμός είναι άρτιος ή όχι, μπορούμε να δώσουμε ένα καλύτερο όνομα σε αυτή την εντολή, της δίνουμε το όνομα einaiArtios ορίζοντας (def) μια συνάρτηση που επιστρέφει (return) αυτή την εντολή:

```
def einaiArtios(x):
    return(x%2==0)
>>> x = 65536
>>> einaiArtios(x)
True
>>> x = 65537
>>> einaiArtios(x)
False
```

Στρογγυλοποίηση;

Πρόσθεση, αφαίρεση και πολλαπλασιασμός φυσικών αριθμών

Το παρακάτω πρόγραμμα εμφανίζει τον πίνακα πρόσθεσης όπως φαίνεται στη σελίδα 13 του βιβλίου σας (κατεβάστε το).

```
def pinakasProsthesis():
    for i in range(10):
        for j in range(10):
            print(str(i+j).rjust(4),end = '')
        print()

pinakasProsthesis()
```

Παραδείγματα - Εφαρμογές

Να υπολογιστούν τα γινόμενα: (α) $, (β) $ (γ) $ (δ) $

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. Στους μεγάλους αριθμούς δε βάζουμε τελείες αλλά γράφουμε τα ψηφία συνεχόμενα, π.χ. ο αριθμός ένα εκατομμύριο γράφεται 1000000 και όχι ~1.000.000~.

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 89*7 + 89*3 = 623 + 267 = 890, που είναι το σωστό αποτέλεσμα.

Ασκήσεις

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.

> 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)
57
```

και η απάντηση είναι 57 κιλά ψωμί.

> Ένα σχολείο έχει 12 αίθουσες διδασκαλίας. Οι 7 χωράνε από 20 διπλά θρανία και οι υπόλοιπες από 12 διπλά θρανία. Στο σχολείο εγγράφηκαν: στην Α' τάξη 80 παιδιά, στην Β' τάξη 58 παιδιά και στην Γ' τάξη 61 παιδιά. Επαρκούν οι αίθουσες για τα παιδιά αυτού του Γυμνασίου;

Στη συγκεκριμένη περίπτωση οι μαθητές που χωράνε στα θρανία είναι 7*20*2+(12-7)*12*2. Οι μαθητές που εγγράφηκαν είναι 80+58+61. Οπότε η απάντηση της ερώτησης στην python μπορεί να είναι η εξής:

```
>>> 7*20*2+(12-7)*12*2 >= 80+58+61
True
```

Άρα οι μαθητές που χωράνε στα θρανία είναι περισσότεροι από αυτούς που εγγράφηκαν. Όμως αν θέλουμε να δούμε τι ακριβώς συμβαίνει θα μπορούσαμε να κάνουμε το εξής:

```
>>> xwrane = 7*20*2+(12-7)*12*2
>>> xwrane
400
>>> eggr = 80+58+61
>>> eggr
199
>>> xwrane >= eggr
True
```

Θυμηθείτε ότι το = δεν είναι ο τελεστής της ισότητας (==), στην ουσία με τον τελεστή = δίνουμε ένα όνομα σε μια τιμή, έτσι όταν γράφουμε xwrane >= eggr η python θα αντικαταστήσει με τις τιμές και κατά συνέπεια θα επιστρέψει το αποτέλεσμα της πράξης 400>=199 που όπως είδαμε είναι True, αφού η πρόταση ισχύει.

Δυνάμεις

> Ο Κωστάκης, η Ρένα και ο Δημήτρης έκαναν τις πράξεις στην αριθμητική παράσταση: $8 \cdot (2 \cdot 3 + 4 \cdot 6) + 5 \cdot (7 + 7 \cdot 9) + 10$ και βρήκαν ο καθένας διαφορετικό αποτέλεσμα. Ο Κωστάκης βρήκε 1.312, η Ρένα 600 και ο Δημήτρης 180.
>
> 1. Βρες ποιο από τα τρία αποτελέσματα είναι το σωστό.
> 2. Μπορείς να μαντέψεις με ποια σειρά έκανε ο καθένας τις πράξεις;
> 3. Διατύπωσε έναν κανόνα για την προτεραιότητα που πρέπει να τηρούμε, όταν κάνουμε πράξεις σε μια αριθμητική παράσταση.

Στην python είναι πολύ εύκολη αυτή η πράξη, με βάση τους κανόνες που μάθαμε:

```
>>> 8*(2*3+4*6)+5*(7+7*9)
590
```

Η προτεραιότητα των πράξεων είναι:

1. Υπολογισμός δυνάμεων
2. Υπολογισμός πολλαπλασιασμών και διαιρέσεων
3. Υπολογισμός προσθέσεων και αφαιρέσεων Αν υπάρχουν παρενθέσεις γίνονται πρώτα οι πράξεις μέσα στις παρενθέσεις, με την παραπάνω σειρά.

Για τις δυνάμεις ο τελεστής στην python είναι ** ,δηλαδή αν θέλουμε να υπολογίσουμε το 10^2 θα γράψουμε 10**2, με όμοιο τρόπο μπορούμε να υπολογίσουμε και τις υπόλοιπες δυνάμεις.

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

> 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 $8 \cdot 9 = 72$.

Το ανάπτυγμα ενός αριθμού σε δυνάμεις του 10 μπορεί να υπολογιστεί με το ακόλουθο πρόγραμμα:

```
def arAnalisi(intx):
  onoma = {}
  #μονάδα,δεκάδα,εκατοντάδα,χιλιάδα,δεκάδα χιλιάδες, εκατοντάδα χιλιάδες (6 θηλυκά)
  #εκατομμύριο (ουδέτερο)

  genos = ['Θηλυκό']*6
  genos.append('Ουδέτερο')
  onoma[0] = ['Μονάδα','Μονάδες']
  onoma[1] = ['Δεκάδα','Δεκάδες']
  onoma[2] = ['Εκατοντάδα','Εκατοντάδες']
  onoma[3] = ['Χιλιάδα','Χιλιάδες']
  onoma[4] = ['Δεκάδα χιλιάδες','Δεκάδες χιλιάδες']
  onoma[5] = ['Εκατοντάδα χιλιάδες','Εκατοντάδες χιλιάδες']
  onoma[6] = ['Εκατομμύριο','Εκατομμύρια']

  analisi = ''
  psifioOn = {'Θηλυκό':['Μηδέν','Μία','Δύο','Τρεις','Τέσσερις','Πέντε','Έξι','Εφτά',
    'Οχτώ','Εννιά'],
    'Αρσενικό':['Μηδέν','Ένας','Δύο','Τρεις','Τέσσερις','Πέντε','Έξι','Εφτά',
    'Οχτώ','Εννιά'],
    'Ουδέτερο':['Μηδέν','Ένα','Δύο','Τρία','Τέσσερα','Πέντε','Έξι','Εφτά',
    'Οχτώ','Εννιά']}
  if intx < 1 or intx >= 1e7:
    return(None)
  else:
    for b in range(6,-1,-1):
      dinami = 10**b
      if intx >= dinami:
        psifio = intx//dinami
        analisi += psifioOn[genos[b]][psifio]
        analisi += ' '
        analisi += onoma[b][psifio>1]
        analisi += ' '
        intx -= psifio*dinami
    return(analisi)


x = input('Δώσε αριθμό:')
x = float(x)
intx = int(x)
print(arAnalisi(intx))
```

> Συμπλήρωσε τον πίνακα τα τετράγωνα και τους κύβους των αριθμών από το 8 μέχρι το 25.

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

> Βρες τα τετράγωνα των αριθμών 10,20,30,40,50,60,70,80 και 90.

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,
```

> 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
```

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) $3 \cdot (5 + 2)^2$.

```
>>> 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
```

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

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

> Βρες τις τιμές των παραστάσεων: (α) $(6 + 5)^2$ και $6^2 + 5^2$, (β) $(3 + 6)^2$ και $3^2 + 6^2$. Τι παρατηρείς;

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

> Γράψε τους αριθμούς: (α) 34.720, (β) 123.654, (γ) 890.650 σε αναπτυγμένη μορφή με χρήση των δυνάμεων του 10.

Η άσκηση αυτή μπορεί να λυθεί με τη χρήση της συνάρτηση arAnalisi που γράφηκε παραπάνω. Θυμηθείτε ότι στην python δεν γράφουμε την τελεία στις χιλιάδες οπότε:

```
>>> print(arAnalisi(34720))
Τρεις Δεκάδες χιλιάδες Τέσσερις Χιλιάδες Εφτά Εκατοντάδες Δύο Δεκάδες
>>> print(arAnalisi(123654)
Μία Εκατοντάδα χιλιάδες Δύο Δεκάδες χιλιάδες Τρεις Χιλιάδες Έξι Εκατοντάδες Πέντε Δεκάδες Τέσσερις Μονάδες
>>> print(arAnalisi(890650))
Οχτώ Εκατοντάδες χιλιάδες Εννιά Δεκάδες χιλιάδες Έξι Εκατοντάδες Πέντε Δεκάδες
```

> Υπολογίστε τις πράξεις $(1 + 2) \cdot (3 + 4), 1 \cdot (2 + 3 \cdot 4), (1 \cdot 2 + 3) \cdot 4, 1 + (2 + 3) \cdot 4$:

```
>>> (1+2)*(3+4)
21
>>> 1*(2+3*4)
14
>>> (1*2+3)*4
20
>>> 1+(2+3)*4
21
```

> Υπολογίστε τις πράξεις $2 + 2 \cdot 2, 3 + 3 \cdot 3, 4 + 4 \cdot 4 \cdot 4, 5 + 5 \cdot 5 + 5 \cdot 5, 5 \cdot 5 + 5 \cdot 5 \cdot 5, 4 + 4 \cdot 4 - 4$

```
>>> 2+2*2
6
>>> 3+3*3
12
>>> 4+4*4*4
68
>>> 5+5*5+5*5
55
>>> 5*5+5*5*5
150
>>> 4+4*4-4
16
```

> Βρείτε τους αριθμούς που είναι ταυτόχρονα τρίγωνοι και τετράγωνοι:

```
import math
def trigwnos(n):
  return(n*(n+1)/2)

def triwnoitetragwnoi():
  n = 2
  while True:
    if math.floor(math.sqrt(trigwnos(n))) == math.sqrt(trigwnos(n)):
      print(n,trigwnos(n))
    n += 1

triwnoitetragwnoi()
```

Το αποτέλεσμα του παραπάνω προγράμματος είναι:

```
1
36
1225
41616
1413721
48024900
1631432881
55420693056
1882672131025
```

Οι Ρωμαίοι χρησιμοποιούσαν το λατινικό αλφάβητο και για τους αριθμούς: M = 1000 D = 500 C = 100 L = 50 X = 10 V = 5 I = 1 Οι Ρωμαίοι στη γραφή των αριθμών τους χρησιμοποιούσαν την προσθετική αρχή από τα αριστερά προς τα δεξιά αλλά και την αφαιρετική αρχή. Το 1 γράφεται I, το 2 γράφεται II, το 3 γράφεται III. Το 4 γράφεται IV (5-1), το 9 γράφεται IX (10-1), το 40 γράφεται XL (50-10), το 900 γράφεται CM (1.000-100), κ.λπ. Ένα πρόγραμμα που διαβάζει έναν ρωμαϊκό αριθμό και τον μετατρέπει σε δεκαδικό μπορεί να είναι το εξής:

```
def romanToDec(romStr):
  romNum = {}
  romNum[1000] = 'M'
  romNum[900]  = 'CM'
```

```
  romNum[500]  = 'D'
  romNum[400]  = 'CD'
  romNum[100]  = 'C'
  romNum[90]   = 'XC'
  romNum[50]   = 'L'
  romNum[40]   = 'XL'
  romNum[10]   = 'X'
  romNum[9]    = 'IX'
  romNum[5]    = 'V'
  romNum[4]    = 'IV'
  romNum[1]    = 'I'

  result = 0

  v = list(sorted(romNum.keys()))
  value = v.pop()
  while romStr != '':
    chars = len(romNum[value])
    if (romStr[:chars] == romNum[value]):
      result += value
      romStr = romStr[chars:]
    else:
      if v == []:
        #there are symbols left that are
        #not parsed
        return(None)
      else:
        value = v.pop()
  return(result)

print(romanToDec('MCDXXIV'))
```

Ευκλείδια διαίρεση

> 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
```

Άρα, 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 (Άσκηση 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
```

Άσκηση 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
```

Άσκηση 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
```

> Ποια μπορεί να είναι τα υπόλοιπα της διαίρεσης ν:8.
>
> ```
> >> for i in range(100):
>         print(i%8, end = ', ')
> 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4,
> ```
>
> Φυσικά τα υπόλοιπα μπορεί να είναι από 0 έως 7.
>
> 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 $9 \cdot 73 + 4 = 661$.

> 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])
```

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

> 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.
>
> 1. Μπορεί να γίνουν 10 πακέτα δώρων; Αν ναι, πόσα από κάθε είδος θα έχει κάθε πακέτο;
> 2. Πόσα πακέτα δώρων μπορεί να γίνουν με όλα τα διαθέσιμα είδη;
> 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
```

Έτσι 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>
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))
```

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 ""))
```

> 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.

> 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,
12510,Ναι,Ναι,Όχι,Ναι,Όχι,Ναι,Ναι,Όχι,Όχι,
  772,Ναι,Όχι,Ναι,Όχι,Όχι,Όχι,Όχι,Όχι,Όχι,
  224,Ναι,Όχι,Ναι,Όχι,Ναι,Όχι,Όχι,Όχι,Όχι,
13600,Ναι,Όχι,Ναι,Ναι,Ναι,Όχι,Ναι,Ναι,Ναι,
```

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). Έτσι 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

> 1. Ισχύει ότι: (100 - 30) - 10 = 100 - (30 - 10)

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

> 1. Για να πολλαπλασιάσουμε έναν αριθμό με το 11 τον πολλαπλασιάζουμε με το 10 και προσθέτουμε 1.

Ισχύει

1. Το γινόμενο $3 \cdot 3 \cdot 3$ γράφεται 3^3.

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

1. Το 2^5 ισούται με 10.

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

1. $\alpha + \alpha + \alpha + \alpha = 4 \cdot \alpha$.

Ισχύει

1. $\alpha \cdot \alpha \cdot \alpha \cdot \alpha \cdot \alpha \cdot \alpha = \alpha^5$.

Δεν ισχύει

1. $2^3 + 3 = 11$.

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

1. $3 \cdot 10^2 + 2 \cdot 10^1 + 2 \cdot 10^0 = 322$.

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

1. $20 - 12 : 4 = 2$.

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

1. $9 \cdot 3 - 2 + 5 = 30$.

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

1. $(3 \cdot 1 - 3) : 3 = 0$.

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

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

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

1. Η διαφορά δύο περιττών αριθμών είναι πάντα περιττός αριθμός.

Δεν ισχύει

1. Αν ο αριθμός α είναι πολλαπλάσιο του αριθμού β, τότε ο α διαιρείται με το β.

Ισχύει

1. Το 38 είναι πολλαπλάσιο του 2 και του 3.

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

Το 38 δεν είναι πολλαπλάσιο του 3

1. Ο αριθμός 450 διαιρείται με το 3 και το 9.

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

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

1. 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.

1. Το ΕΚΠ των 2 και 24 είναι ο αριθμός 48.

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

1. Η διαίρεση 420 : 15 δίνει υπόλοιπο 18.

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

Δεν ισχύει

1. 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.

1. 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.

1. 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
```

1. Ο αριθμός 224 διαιρείται με το 4 και το 8.

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

Ισχύει!