Σύνδεση με τα προηγούμενα

Στην πρώτη μας συνάντηση είδαμε ότι μαζί με την γλώσσα Python, στον υπολογιστή μας εγκαθίσταται το **προγραμματιστικό περιβάλλον IDLE**. Αυτό περιλαμβάνει έναν εξελιγμένο **συντάκτη** κειμένου (editor), όπου συντάσσουμε τα προγράμματά μας, τον κώδικά μας, καθώς και ένα **κέλυφος** (shell) διερμήνευσης και εκτέλεσης της Python, όπου εκτελούνται τόσο μεμονωμένες εντολές που πληκτρολογούμε απευθείας σε αυτό, όσο και τα προγράμματα που ζητούμε από τον συντάκτη. Παράλληλα είναι ενσωματωμένος ένας **μηχανισμός ελέγχου συντακτικών λαθών** (debugger).

Στο φύλλο εργασίας, θα ξεχωρίζουμε τις δραστηριότητες που θα επιτελείς με τα δύο εικονίδια:

 *Γράψε κώδικα στον συντάκτη του IDLE*, και  *γράψε κώδικα στο κέλυφος*

Τα πρώτα σου προγράμματα διάβασαν δεδομένα με είσοδο από το πληκτρολόγιο, εμφάνισαν τιμές (αριθμητικές και αλφαριθμητικές) στην οθόνη, εκτέλεσαν συγκρίσεις και απλές αριθμητικές πράξεις, ενεργώντας ενίοτε μόνο εφόσον ίσχυε ή όχι (ήταν **True** ή **False**) κάποια συνθήκη. Τα επόμενα προγράμματα εμπλουτίζουν τα παραπάνω.

Θα χρησιμοποιήσεις πρώτα το κέλυφος. Με ποια **προτεραιότητα** εκτελούνται οι πράξεις σε μια σύνθετη έκφραση; Πληκτρολόγησε τις εκφράσεις και συμπλήρωσε τον παρακάτω πίνακα με την τιμή τους (το αποτέλεσμά τους), και με τα ονόματα των πράξεων, όπως στο παράδειγμα:

Έκφραση	Τιμή	1η εκτελείται η πράξη	2η εκτελείται η πράξη
`3+5*2`		*Πολλαπλασιασμός (*)*	*Πρόσθεση (+)*
`3+5**2`			
`(3+5)*2**2`			
`17-8/2`			
`17-8/3`			
`17-8//2`			
`17-8//3`			
`11//3`			
`11%3`			
`47//6%4`			
`47%6//4`			
`47%6//4*3`			
`2*"python"`			
`'py'+'thon'`			

Πληκτρολόγησε επίσης την ανάθεση τιμής στο x και τις εντολές print (που θα δούμε παρακάτω ότι *δεν* είναι εντολή!). Τι συμπεραίνεις για την print; Τι αλλάζει από πριν που γράφαμε σκέτη μια έκφραση στο κέλυφος;

`x=(3+5)*2`	
`print(3+5*2)`	
`print(x, 3*x)`	
`print("py\nthon")`	
`print(2*"python")`	
`print(2,"python")`	
`print('py'+'thon')`	
`print("py"+"thon")`	
`print("py","thon")`	

Τώρα που έμαθες την πράξη της **<u>ακέραιας διαίρεσης</u>** (//) και του **<u>ακέραιου υπολοίπου</u>** (%), ίσως αναρωτιέσαι για τη χρησιμότητά τους. Μπορούν να βοηθήσουν τον υπολογιστή να αναγνωρίσει αν ένας ακέραιος είναι περιττός ή άρτιος! *Μπορείς να σκεφτείς πώς;* Ας πάμε στο πρόγραμμά σου.

Ο μυστικός αριθμός

Θα φτιάξεις ένα «παιχνίδι», από αυτά που μαντεύουν τον αριθμό που σκέφτηκε κάποιος. Ο χρήστης θα είναι αυτός που θα σκέφτεται τον αριθμό· ο υπολογιστής θα τον μαντεύει. Φυσικά ο υπολογιστής θα καθοδηγείται από το πρόγραμμά σου. Με βάση ένα θεώρημα ενός αρχαίου Κινέζου Μαθηματικού, θα χρειαστεί να κάνει στο χρήστη 3 μόνο ερωτήσεις!

Το θεώρημα δουλεύει για κάθε αριθμό, έστω x, από 0 ως 104. Έστω ότι α, β, γ είναι τα υπόλοιπα της διαίρεσης του x με το 3, το 5 και το 7, αντίστοιχα. Υπολογίζεις την παράσταση 70α+21β+15γ. Διαιρείς ακέραια με το 105. Το υπόλοιπο της διαίρεσης είναι το x, ο μυστικός αριθμός! Πάμε!

```
.....
.....
a=int(input("ερώτηση")).....
.....
.....
.....
.....
.....
.....
.....
.....
.....
```

Φτιάξε πρόγραμμα, με το όνομα **guess1.py**, το οποίο:

- ζητάει από το χρήστη να σκεφτεί έναν μυστικό αριθμό από το 0 μέχρι και το 104,
- του ζητάει διαδοχικά το υπόλοιπο της διαίρεσής του με το 3, το 5 και το 7,
- εφαρμόζει το θεώρημα για να τον βρει, και
- τον ανακοινώνει.

Ποιος σε μελετάει;

Φταρνίστηκες κι αμέσως σου λένε έναν αριθμό (π.χ. 136). Αθροίζεις τα ψηφία του (1+3+6=10), και ψάχνεις το γράμμα της αλφαβήτας που αντιστοιχεί στον αριθμό αυτό (Το Κ στο 10). Αν είναι μεγαλύτερος του 24, αθροίζεις ξανά τα ψηφία του και ξαναψάχνεις το γράμμα. Ένας φίλος σου με όνομα από το γράμμα αυτό σε μελετούσε. Γι' αυτό φταρνίστηκες! Θα φτιάξεις ένα πρόγραμμα **letter1.py** που θα προσομοιώνει τη συνήθεια αυτή.

>>> Πριν από αυτό, γύρισε στο κέλυφος. Ας πούμε ότι ο αριθμός είναι το 3067.

1. *Όρισε τη μεταβλητή x με τιμή 3067.*
   `x = .....`
2. *Με ποια πράξη, που έμαθες, μπορείς να πάρεις το τελευταίο ψηφίο του x (το 7) στη μεταβλητή d1;*
   `d1 = x.....`
3. *Δες την d1.*
   `d1`
4. *Αν βρήκες την πράξη, θα μπορούσες να την επαναλάβεις στο τμήμα του αριθμού 306_, για να πάρεις το επόμενο ψηφίο, το 6, στη d2.*
   Όμως, με ποια πράξη θα πάρεις το 306 από το 3067, σε μια μεταβλητή r;
   `r = x.....`
5. *Δες την r.*
   `r`
6. *Τη βρήκες σωστά, μπράβο. Για να της πάρεις στη μεταβλητή d2 το τελευταίο ψηφίο της, κάνε με την r την ίδια πράξη που έκανες με την x στο βήμα 2.*
   `d2 = r.....`
7. *Δες την d2.*
   `d2`
8. *Όπως στο βήμα 4, πάρε πάλι στην r το τμήμα του αριθμού 30__. Δεν θα χρησιμοποιήσεις άλλη μεταβλητή, αλλά το αποτέλεσμα της πράξης θα το αναθέσεις εκ νέου στην r.*
   `r = r.....`
9. *Δες την ανανεωμένη r, που θα έχει τιμή 30.*
   `r`
10. *Επανάλαβε με αυτήν ό,τι έκανες στο βήμα 5, για να πάρεις στο d3 το ψηφίο 0.*
    `d3 = r.....`
11. *Δες την d3.*
    `d3`
12. *Με την ίδια πράξη που έκανες στο βήμα 6, πάρε πάλι στην r το τμήμα του αριθμού 3___. Ανάθεσε το αποτέλεσμα της πράξης πάλι στην r.*
    `r = r.....`
13. *Δες την ανανεωμένη r, που θα έχει τιμή 3.*
    `r`
14. *Επανάλαβε με αυτήν ό,τι έκανες στο βήμα 5, για να πάρεις στο d4 το ψηφίο 0. Δες το d4*
    `d4 = r.....`
15. *Υπολόγισε ξανά το r, όπως πριν, και εμφάνισέ το. Τι τιμή έχει; Μηδέν, ε; Ενδιαφέρον!*
    `r = r.....`
    `r`

Όπως διαπίστωσες, η **ακέραια διαίρεση** και το **ακέραιο πηλίκο** είναι τα εργαλεία για τον υπολογισμό των ψηφίων ενός αριθμού. Καθώς δεν γνωρίζουμε εκ των προτέρων, ως προγραμματιστές, πόσα ψηφία μπορεί να έχει ο αριθμός, πώς θα ξέρουμε πόσες μεταβλητές θα χρειαστούμε, και πότε θα σταματήσουμε; Παρατήρησε τα βήματα παραπάνω. Τι τιμή πήρε η r όταν εξαντλήθηκαν τα ψηφία του αριθμού;

>>>

*Ο τελεστής **!=** δηλώνει τη σχέση ανισότητας «διάφορο», το αντίθετο δηλαδή της ισότητας **==**.*

Δοκίμασε στο κέλυφος τις γραμμές δεξιά. Τι παρατηρείς;

```
bored = "n"
while bored != "y":
    bored = input("Are you bored?)
```

.....

.....

.....

.....

*Η **εντολή while** (όπως και η if) ελέγχει μια συνθήκη, και όσο αυτή ισχύει, **επαναλαμβάνει** τις εντολές στην εσοχή της. (αντίθετα με την if, που τις εκτελεί μόνο μία φορά).*

*Αν χρειάζεται, μπορεί να ακολουθεί και ο **κλάδος else:** (όπως στην if) που θα εκτελεστεί όταν πάψει να ισχύσει η συνθήκη.*

Πάμε στο πρόγραμμα που θα φτιάξεις.

.....	**Ζήτησε** και **διάβασε** έναν αριθμό **number**.
.....	
s=	Μηδένισε τη μεταβλητή **s**. Η μεταβλητή **s θα κρατήσει άθροισμα των ψηφίων του number**.
r=	Δώσε στην **r** την αρχική τιμή **number**. Η **r θα κρατάει τα διαδοχικά κομμάτια του number**.
`while r >0:`	Όσο το **r** δεν είναι (ακόμη) μηδέν,
d=	βάλε στο **d** το τελευταίο ψηφίο του **r**
s=	πρόσθεσε το **d** στο **s** (ή, αλλιώς, αύξησε το **s** κατά **d**)
r=	ανανέωσε το **r**, χωρίς το τελευταίο του ψηφίο
`print(s)`	Εμφάνισε το **s**

Στο σημείο αυτό, έχεις υπολογίσει (και εμφανίσει) το άθροισμα των ψηφίων του αριθμού.

>>>

*Το x είναι ένα **αλφαριθμητικό (string)**, δηλαδή μια **ακολουθία** από χαρακτήρες. Με **δείκτες** μπορούμε να πάρουμε συγκεκριμένα τμήματα οποιασδήποτε ακολουθίας.*

Δοκίμασε στο κέλυφος (σε χωριστές γραμμές) τα εξής. Σημείωσε τι παρατηρείς:

`x="python"  x[1]  x[0]  x[-1]  x[5]  x[6]  x[2:4]  x[0:5:2]`

.....

.....

.....

.....

Γύρισε στο πρόγραμμά σου, για να το ολοκληρώσεις.

Ως προγραμματιστής ή προγραμματίστρια, όταν υλοποιείς τη λύση ενός προβλήματος, ή μια προσομοίωση στον υπολογιστή, έχεις την ευκαιρία να σκεφτείς καλύτερα τη δομή τους. Στη συνήθεια με το φτάρνισμα, επειδή κάνουμε τις πράξεις στο μυαλό μας, για λόγους ευκολίας παραβλέπουμε ένα λάθος. Όταν αθροίζουμε εκ νέου τα ψηφία του αθροίσματος, ευνοούμε τα πρώτα γράμματα του αλφαβήτου. Στο πρόγραμμά σου, θα είσαι ακριβοδίκαιος, και θα παίρνεις το υπόλοιπο της διαίρεσης του αθροίσματος με το 24 (άρα έναν αριθμό από το 0 έως το 23). Έτσι όλα τα γράμματα από το 1ο ως το 24ο θα αντιστοιχιστούν δίκαια και ισοπίθανα από το άθροισμα των ψηφίων του αριθμού.

`s= .....`	Ανανέωσε το s με το υπόλοιπο της διαίρεσής του με το 24. 
`alphabet="ΑΒΓ....."`	Στο αλφαριθμητικό alphabet βάλε όλα τα (κεφαλαία) γράμματα του αλφάβητου.
`print(number,alphabet[.....])`	Εμφάνισε τον αριθμό και το γράμμα από τη θέση s-1 (γιατί όχι την s;) του alphabet. 

Συναρτήσεις

Δοκίμασε στο κέλυφος (σε χωριστές γραμμές) τα εξής, και σημείωσε τα αποτελέσματα:

```
abs(-3.14)   abs(7)   max(3, -2, 5)   min(2, -1)   license()
.....
```

Μόλις χρησιμοποίησες 4 από τις **ενσωματωμένες συναρτήσεις** της Python. Έχεις ήδη πιο πριν χρησιμοποιήσει ακόμη 2-3: την **input()**, την **int()** και την **print()**. Συναρτήσεις θα φτιάχνεις κι εσύ, όπως και κάθε προγραμματιστής.

*Μια **συνάρτηση** είναι ένα τμήμα κώδικα με ένα όνομα. Εκτελείται μόνο αν **κληθεί** κατάλληλα μέσα. Μέσα στην παρένθεση περιλαμβάνονται οι **παράμετροι** της συνάρτησης. Αυτές είναι τα δεδομένα της. Μια συνάρτηση μπορεί να **επιστρέφει αποτελέσματα**.*

Ποιος σε μελετάει; (με συναρτήσεις)

Θα τροποποιήσεις το πρόγραμμά σου σε **letter2.py**, ώστε να περιλαμβάνει δύο συναρτήσεις:

- την digitSum(x), που θα υπολογίζει και θα επιστρέφει το άθροισμα των ψηφίων της παραμέτρου x, και
- την letter(n) που θα υπολογίζει και θα επιστρέφει το γράμμα του αλφαβήτου που αντιστοιχεί στην παράμετρο n, βάσει της λογικής που αναπτύξαμε.

Οι δύο αυτές συναρτήσεις θα καλούνται κατάλληλα μέσα στο πρόγραμμά σου.

*Μια συνάρτηση ξεκινά με την **εντολή def** για την **επικεφαλίδα** της, που περιέχει το όνομά της, τις **τυπικές παραμέτρους της** σε παρένθεση, και την «:». Ακολουθεί το **σώμα** της σε εσοχή.*
*Επιστρέφει αποτελέσματα με την **εντολή return**.*
*Οι μεταβλητές μέσα σε μια συνάρτηση είναι **τοπικές**. Δεν είναι γνωστές έξω από αυτήν, ακόμα και αν υπάρχουν συνωνυμίες.*
Αντίστροφα, μεταβλητές που προϋπάρχουν έξω από τη συνάρτηση είναι γνωστές και σε αυτήν.
Σε περίπτωση συνωνυμίας, προτεραιότητα έχουν οι τοπικές.
*Κατά την κλήση της συνάρτησης, οι **τυπικές παράμετροι** αντικαθίστανται από τις **πραγματικές παραμέτρους**.*

`def digitSum(x):`	Η **επικεφαλίδα** με το όνομα της συνάρτησης και τις **τυπικές παραμέτρους** της (η x εδώ).
`s=0` `r=x`	Η s και η r είναι **τοπικές** μεταβλητές, ασχέτως συνωνυμιών.

`while r>0:`
`  d=r%10`
`  s+=d`
`  r//=10`
`return s%24`
	Το σώμα επιτελεί ό,τι έκανες και πριν απευθείας μέσα στο πρόγραμμα.
	Εναλλακτική μορφή της εντολής `s=s+d`
	Εναλλακτική μορφή της εντολής `r=r//10`
	Η συνάρτηση μέσω του υπολοίπου-24 επιστρέφει πάντα έναν αριθμό μεταξύ 0 και 23.

`def letter(n):`
`ab="ΑΒΓ……"`
`return ab[……]`
	Επικεφαλίδα με όνομα και μία παράμετρο.
	Η αλφαβήτα σε μία αλφαριθμητική μεταβλητή.
	Συμπλήρωσε το περιεχόμενο της αγκύλης.

`print("Δώσε…… )` `number=……`	Εδώ ξεκινά το κύριο πρόγραμμά σου. Ζητάει και διαβάζει έναν αριθμό number. Με τις συναρτήσεις σου έχει εύκολη δουλειά!
`print(……)`	Συμπλήρωσε την παρένθεση και τέλειωσες!

Μπραβο! Ήρθες σε επαφή σήμερα με πάρα πολλές έννοιες και χαρακτηριστικά της γλώσσας Python.

Θυμάσαι ότι την έχουμε χαρακτηρίσει όμορφη και εκφραστική; Το πρόγραμμα που έφτιαξες, με μια άλλη μεθοδολογία, και με εντολές που έχεις να μάθεις, θα μπορούσε να συντομευτεί σε μία ακριβώς γραμμή:

```
print("ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩ"[sum([int(d) for d in input("Δώσε έναν φυσικό αριθμό: ")])%24-1])
```

Αν τα ολοκλήρωσες όλα, κι έχεις όρεξη και χρόνο, τώρα ή στο σπίτι, δοκίμασε την αντίστροφη παραλλαγή του μυστικού αριθμού.

Επέκταση: Μπορείς να φτιάξεις το αντίστροφο πρόγραμμα (**guess2.py**);

```
import random.....
x=random.randint(0,104).....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
```

Ο υπολογιστής θα «σκέφτεται» έναν αριθμό από το 0 έως το 104, θα ανακοινώνει διαδοχικά στο χρήστη το υπόλοιπο της διαίρεσής του με το 3, το 5 και το 7, και θα του ζητάει να τον μαντέψει. Ανάλογα με το αποτέλεσμα, θα του εμφανίζει ένα κατάλληλο μήνυμα.
Η συνάρτηση randint() της βιβλιοθήκης random παράγει τον τυχαίο αριθμό.