ΠΜΣ Εκπαιδευτικές Εφαρμογές με την Επιστημολογία STEM

Πανεπιστήμιο Θεσσαλίας – Ανώτατη Σχολή Παιδαγωγικών και Τεχνολογικών Επιστημών (ΑΣΠΑΙΤΕ)

Μ2: Παραδείγματα Υ.Σ.-STEM με Python (Θ+Ε)

Π.χ. Χάος και δημιουργία σε Python χωρικών χαοτικών συστημάτων(Ε)

ΣΗΜΕΙΩΣΕΙΣ

Στα πλαίσια του μαθήματος αναπτύχθηκαν παραδείγματα σε γλώσσα προγραμματισμού Python. Χρησιμοποιήθηκε το PyCharm LDE, το οποίο μπορείτε να εγκαταστήσετε στον Η/Υ σας από εδώ: https://www.jetbrains.com/pycharm/download/#section=windows

Συστάσεις:

- Δημιουργήστε έναν φάκελο στον Η/Υ σας, όπου θα σώζετε τα .py script files. Σε αυτόν τον φάκελο θα ορίσετε το PyCharm να έχει πρόσβαση.
- Δημιουργήστε ένα αρχείο **main.py** στο οποίο θα δοκιμάζουμε κάθε φορά τον κώδικα της εκάστοτε άσκησης. Μέσω αυτού του αρχείου θα καλούμε είτε έτοιμες είτε δικές μας συναρτήσεις (δομημένος προγραμματισμός)
- Στο μέρος Σύνδεσμοι στο eclass του μαθήματος θα βρείτε χρήσιμο διαδικτυακό υλικό που θα σας βοηθήσει με την Python
- Δημιουργήστε ένα αρχείο **myfunctions.py** στο οποίο θα γράφετε τις δικές σας συναρτήσεις. Τις συναρτήσεις που υπάρχουν στα παρακάτω λυμένα παραδείγματα (δηλαδή οι κώδικες που αρχίζουν με τη λέξη def), μπορείτε να τις γράψετε όλες στο αρχείο myfunctions.py Με τον τρόπο αυτό, μπορείτε να τις καλείτε, ενσωματώνοντας σε κάποιο νέο αρχείο python script (π.χ. το main.py) ως εξής:
  *from myfunctions import **
- Τα παρακάτω παραδείγματα κώδικα χωρίζονται σε δύο μέρη: 1) Το **ΜΕΡΟΣ Α** που αφορά κυρίως δήλωση και κλήση συναρτήσεων, είτε έτοιμες, είτε όσες δημιουργεί ο προγραμματιστής μόνος του και το **ΜΕΡΟΣ Β** που αφορά την κλήση έτοιμων συναρτήσεων γραφικών χελώνας (turtle graphics)

ΜΕΡΟΣ Α - ΣΥΝΑΡΤΗΣΕΙΣ

Άσκηση 1: Γράψτε το παρακάτω πρόγραμμα (στο main.py), το οποίο βρίσκει τη λύση μίας πρωτοβάθμιας εξίσωσης

```
# Πρωτοβάθμια εξίσωση της μορφής ax+b=0
# Η λύση υπολογίζεται σε x=-b/a, εφόσον το a δεν είναι 0
# Εισαγωγή συντελεστών
a = int(input("Δώστε το a:"))
b = int(input("Δώστε το b:"))
print("Λύνω την εξίσωση: ",a,"x +",b,"= 0")
# Έλεγχος για το a
if a!=0:
    x = -b/a
    print("x=", x)
else:
    print("Η εξίσωση είναι αδύνατη")
```

Ζητούμενα:

1. Εκτελέστε το πρόγραμμα, δίνοντας κάθε φορά διαφορετικούς αριθμούς για τους συντελεστές του a,b
2. Δημιουργήστε τη συνάρτηση **prwtovathmia(a,b)** εντός του myfunctions.py

```
#************************************************************
def prwtovathmia(a, b):
    print("Λύνω την εξίσωση: ", a, "x +", b, "= 0")
    # Έλεγχος για το a
    if a != 0:
       x = -b / a
    return x
#************************************************************
```

Για να την καλέσετε στο main.py γράψτε τον κώδικα:

```
print(prwtovathmia(6,3))
```

Δοκιμάστε τις τιμές a,b που δώσατε και στο ζητούμενο 1 και δείτε εάν η λύση είναι η ίδια.

3. (προαιρετικό αλλά σημαντικό) Δημιουργήστε το **διάγραμμα ροής** που αντιστοιχεί στον κώδικα επίλυσης της πρωτοβάθμιας εξίσωσης.
    a. Για τον ορισμό του διαγράμματος ροής, δείτε εδώ
    b. Για παραδείγματα διαγραμμάτων ροής, δείτε εδώ
    c. Για να δημιουργήσετε διάγραμμα ροής, χρησιμοποιήστε το flowgorithm, το οποίο μπορείτε να εγκαταστήσετε από εδώ

Άσκηση 2: Γράψτε το παρακάτω πρόγραμμα (στο main.py), το οποίο βρίσκει τη λύση μίας δευτεροβάθμιας εξίσωσης

```
# Δευτεροβάθμια εξίσωση της μορφής ax^2+bx+c=0
# Εάν a=0 τότε το πρόβλημα απλοποιείται στη λύση της
# πρωτοβάθμιας εξίσωσης της μορφής bx+c=0
# Εάν a!=0, τότε θα υπολογιστει η διακρίνουσα D=b*b-4*a*c
# και θα ελεγχθούν τρεις περιπτώσεις σύγκρισης της D με το 0

# Εισαγωγή συντελεστών
a = int(input("Δώστε το a: "))
b = int(input("Δώστε το b: "))
c = int(input("Δώστε το c: "))
print("Λύνω την εξίσωση ",a,"x**2+ ",b,"x +",c," = 0")
```

```
# Έλεγχος για το a
if a == 0:  # έχουμε την πρωτοβάθμια bx+c=0
    if b != 0:
        x = -c/b
        print("Λύση πρωτοβάθμιας με ρίζα:", x)
    else:
        print("Η εξίσωση είναι αδύνατη")
else:         # πλήρης δευτεροβάθμια
    D = b**2-4*a*c  # υπολογισμός διακρίνουσας
    print("Διακρίνουσα =",D)
    if D == 0: #μία διπλή ρίζα
        x = -b/(2*a)
        print("Μηδενική διακρίνουσα, διπλή ρίζα x =", x)
    else:  # δύο λύσεις (πραγματικές ή μιγαδκές)
        x1=(-b-D**(1/2))/(2*a)  # υπολογισμός τετραγωνικής ρίζας με
ύψωση εις την 1/2
        x2=(-b+D**(1/2))/(2*a)
        if D>0:
            print("Θετική διακρίνουσα, δύο πραγματικές λύσεις:",
x1, "και", x2)
        else:
            print("Αρνητική διακρίνουσα, δύο Μιγαδικές ρίζες,
συζυγείς:", x1, "και", x2)
```

Ζητούμενα:

1. Εκτελέστε το πρόγραμμα, δίνοντας κάθε φορά διαφορετικούς αριθμούς για τους συντελεστές του a,b,c
2. Δημιουργήστε τη συνάρτηση **deyterovathmia(a,b,c)** εντός του myfunctions.py και καλέστε τη συνάρτηση από τη main.py (π.χ. print(deyterovathmia(6,3,2))
3. (προαιρετικό) Δημιουργήστε το διάγραμμα ροής επίλυσης της δευτεροβάθμιας εξίσωσης, ενσωματώνοντας όλες τις περιπτώσεις που μπορεί κανείς να συναντήσει

Άσκηση 3: Γράψτε στη main.py τον παρακάτω κώδικα, ο οποίος βρίσκει τις Πυθαγόρειες Τριάδες ακεραίων (a,b,c) με **0<a<b<c<30**

Αναζητήστε πληροφορίες σχετικά με τις Πυθαγόρειες Τριάδες από εδώ

```
#Χρησιμοποιούμε τις έτοιμες συναρτήσεις από τη βιβλιοθήκη math
from math import *
for a in range(1,30):
    for b in range(a+1,30):
        for c in range(b+1,30):
            if pow(a,2) + pow(b,2) == pow(c,2):
                print(a,b,c)
```

Ζητούμενα:

1. Εκτελέστε το πρόγραμμα και επιβεβαιώστε ότι εκτυπώνει τις ακόλουθες τριάδες:

```
5 12 13
6 8 10
7 24 25
8 15 17
9 12 15
```

10 24 26
12 16 20
15 20 25
20 21 29

2. Δημιουργήστε τη συνάρτηση PyTriangles(N) και γράψτε την στο myfunctions.py
   a. Καλέστε κατάλληλα την PyTriangles(30) και επιβεβαιώστε ότι εκτυπώνει την παραπάνω λίστα τριάδων
   b. Καλέστε την PyTriangles(100). Παρατηρήστε εάν τα αποτελέσματά σας ανήκουν στα αποτελέσματα του παρακάτω πίνακα. Εξηγήστε για ποιο λόγο υπάρχουν διαφορές

(3, 4, 5)	(5, 12, 13)	(8, 15, 17)	(7, 24, 25)
(20, 21, 29)	(12, 35, 37)	(9, 40, 41)	(28, 45, 53)
(11, 60, 61)	(16, 63, 65)	(33, 56, 65)	(48, 55, 73)
(13, 84, 85)	(36, 77, 85)	(39, 80, 89)	(65, 72, 97)

3. Τροποποιήστε κατάλληλα την PyTriangles(N) ώστε να παράγονται μόνο που δεν είναι πολλαπλάσιες με καμία άλλη, εντός του εύρους $0 < a < b < c < N$

Άσκηση 4: Δημιουργήστε την παρακάτω συνάρτηση findfilioi(x), η οποία χρησιμοποιείται για να ελέγξει εάν δύο φυσικοί αριθμοί (θετικοί ακέραιοι) είναι φίλιοι.

Αναζητήστε πληροφορίες για το ποιοι είναι «φίλιοι αριθμοί», πως υπολογίζονται και ένα παραδείγματα από εδώ

```
def findfilioi(x):
    #Υπολογίζει το άθροισμα των διαιρετών του x
  sum = 0
  for i in range(1,x):
     if x % i == 0:
         sum+=i
         #print(i)
  return sum
```

Στο αρχείο main.py καλέστε την παραπάνω συνάρτηση ως εξής:

```
x = int(input('Dose ton 1o arithmo: '))
y = int(input('Dose ton 2o arithmo:'))
if x == findfilio(y) and y == findfilio(x):
  print('Einai filioi!')
else:
  print('Den einai filioi')
```

Ζητούμενα:

1. Καλέστε κατάλληλα τη συνάρτηση findfilioi(x), στη main.py σύμφωνα με τον κώδικα όπου διαβάζετε δύο ακεραίους από το πληκτρολόγιο. Δώστε το ζεύγος (220,284) και επιβεβαιώστε ότι είναι φίλιοι. Δώστε έπειτα ένα διαφορετικό ζεύγος τυχαίο και επιβεβαιώστε ότι δεν είναι φίλιοι.
2. Το παρακάτω κομμάτι κώδικα δημιουργεί δύο λίστες αριθμών, οι οποίοι είναι οι φίλιοι αριθμοί μέχρι το 100000. Δοκιμάστε τον κώδικα και επιβεβαιώστε ότι τα ζεύγη είναι φίλιοι

```
list1 = [220,1184,2620,5020,6232,10744,12285,17296,63020,79750]
list2 = [284,1210,2924,5564,6368,10856,14595,18416,76084,88730]

for i in range(len(list1)):
    if list1[i] == findfilioi(list2[i]) and list2[i] ==
findfilioi(list1[i]):
        print('Einai filioi!')
    else:
        print('Den einai filioi')
```

Άσκηση 5: Η άσκηση αυτή αφορά τον χαρακτηρισμό και εύρεση πρώτων αριθμών. Αναζητήστε πληροφορίες σχετικά με τους πρώτους αριθμούς εδώ

Πρώτος λέγεται ένας φυσικός αριθμός (θετικός ακέραιος) αν δεν διαιρείται ακριβώς με κανένα άλλον φυσικό αριθμό, εκτός από τον εαυτό του και τη μονάδα.

Άλλος ορισμός είναι: Ένας αριθμός είναι πρώτος όταν έχει ακριβώς δύο φυσικούς αριθμούς ως διαιρέτες. (Άρα η μονάδα δεν θεωρείται πρώτος)

```
def isPrime(x):
#Ελέγχει εάν ένας φυσικός αριθμός είναι πρώτος αριθμός
    if x<2:
        return 'no'
    if x == 2:
        return 'yes'
    for i in range(2,x-1):
        if x%i == 0:
            return 'no'
    return 'yes'
```

Ζητούμενα: Να γίνει πρόγραμμα το οποίο να υπολογίζει και να εμφανίζει τους πρώτους αριθμούς μέχρι το **100.**

ΜΕΡΟΣ Β – TURTLE CODE

Γεωμετρικά Σχήματα με Turtle

Άσκηση 1

```
# This is a sample Python script.

from myfunctions import *
from turtle import *

reset()
Screen()
speed('fastest')
for i in range(4):
    forward(500)
    goto(0,0)
    left(90)
up()
goto(-300,f(300))

color('red')
for x in range(-300,300):
    goto (x,f(x))
    down()
mainloop()
```

Άσκηση 2: Δημιουργήστε **100 κύκλους** τυχαίας ακτίνας στο διάστημα [10,100] και τυχαίου χρώματος RGB

```
from random import *
reset()
Screen()
speed('fastest')
for i in range(100):
    up()
    x = randint(-100,100)
    y= randint(-100, 100)
    goto(x,y)
    down()
    color(random(),random(),random())
    r = randint(5,50)
    circle(r)
mainloop()
```

Άσκηση 3: Σχεδιασμός Σπείρας

circle(x,y): η οποία δέχεται και δεύτερο όρισμα το οποίο καθορίζει το τόξο σε μοίρες που θα σχεδιαστεί.

```
for i in range(20):
    circle(50+10*i,180)
```

Άσκηση 4: Δημιουργήστε **100 ισόπλευρα τρίγωνα**

```
def triangle(x):
    i = 0
    while i < 3:
        forward(x)
        left(120)
        i = i + 1

#*****************************************************************
reset()
Screen()
speed('fastest')
clear()
for i in range(100):
    up()
    x = randint(-100,100)
    y= randint(-100, 100)
    goto(x,y)
    down()
    color(random(),random(),random())
    r = randint(10,100)
    triangle(r)
    #circle(r) #έτοιμη συνάρτηση της turtle
mainloop()
#*****************************************************************
```