ΜΕΜ104 Γλώσσα Προγραμματισμού Ι

3ο φυλλάδιο ασκήσεων

2 Νοεμβρίου 2019

1. Γράψτε ένα πρόγραμμα το οποίο ζητάει μια γωνία θ στο διάστημα $(0, \pi/2)$ και τυπώνει τους τριγωνομετρικούς αριθμούς $\sin\theta$, $\cos\theta$ και $\tan\theta$ στη μορφή:

   ```
   theta = 0.5235987755982988
   sin(theta) = 0.8660254037844386
   cos(theta) = 0.5000000000000000
   tan(theta) = 1.7320508075688767
   ```

   Το πρόγραμμα θα πρέπει να ελέγχει ότι η γωνία που έδωσε ο χρήστης είναι στο διάστημα $(0, \pi/2)$ και τότε να τυπώνει τους τρεις τριγωνομετρικούς αριθμούς, διαφορετικά να τυπώνει ένα κατάλληλο μήνυμα σφάλματος.

   ```
   import math
   theta = float(input('Give theta in (0, pi/2): '))
   if 0 < theta < math.pi/2:
     print('theta =', theta)
     print('sin(theta) =', math.sin(theta))
     print('cos(theta) =', math.cos(theta))
     print('tan(theta) =', math.tan(theta))
   else:
     print('Theta not in range (0, pi/2)')
   ```

2. Γράψτε ένα πρόγραμμα το οποίο διαβάζει τις συντεταγμένες ενός σημείου στο επίπεδο και τυπώνει την απόστασή του από την αρχή των αξόνων.

   ```
   x = float(input('Give x: '))
   y = float(input('Give y: '))
   result = (x ** 2 + y ** 2) ** 0.5
   print('Distance of (', x, ', ', y, ') from (0, 0) is ', result, sep = '')
   ```

3. Γράψτε μια συνάρτηση η οποία επιστρέφει το όρισμά της ανεστραμμένο. Για παράδειγμα, αν η συνάρτηση κληθεί με το όρισμα 'maria' θα πρέπει να επιστρέψει 'airam'.

   ```
   def reverse(s):
     return s[::-1]

   s = input('Give name: ')
   print(reverse(s))
   ```

4. Γράψτε μια συνάρτηση η οποία υπολογίζει τον n-οστό όρο της ακολουθίας Fibonacci. Υπενθυμίζουμε ότι $F_0 = 0$, $F_1 = 1$ και $F_n = F_{n-1} + F_{n-2}$ για $n \geq 2$. Φυσικά, το n πρέπει να είναι όρισμα της συνάρτησης.

```
def fib1(n): #using iteration
  prv = 0
  cur = 1
  nxt = n    # set result if n < 2

  for i in range(2, n + 1):
    nxt = cur + prv
    prv = cur
    cur = nxt
  return nxt

def fib2(n): #using recursion
  if n == 0 or n == 1:
    return n
  else:
    return fib2(n-1) + fib2(n-2)

n = int(input('Give n: '))
print(fib1(n))
print(fib2(n))
```

5. Γράψτε μια συνάρτηση η οποία με όρισμα τον ακέραιο n επιστρέφει τη μεγαλύτερη δύναμη του 2 η οποία διαιρεί ακριβώς το n.

```
def largestPower2Divisor(n):
  p = 0
  while n % 2 == 0:
    p += 1
    n //= 2
  return 2**p

n = int(input('Give n: '))
print(largestPower2Divisor(n))
```

6. Γράψτε μια συνάρτηση η οποία μετατρέπει πόδια και ίντσες σε εκατοστά. Θυμίζουμε ότι ένα πόδι έχει 12 ίντσες και μια ίντσα είναι 2.54 εκατοστά.

```
def convert(feet, inches):
  return (feet * 12 + inches) * 2.54

feet = float(input('Give feet: '))
inches = float(input('Give inches: '))
print(convert(feet, inches))
```

7. Γράψτε μία συνάρτηση η οποία να υπολογίζει την τιμή της $f(x) = x^2 - 3x + 1$. Καλέστε την για κάποια τιμή του x και τυπώστε τα $x, f(x)$.

```
def f(x):
  return x**2 - 3*x + 1

x = float(input('Give x: '))
print('x =', x, ', f(x) =', f(x))
```

8. Γράψτε μία συνάρτηση η οποία να ελέγχει αν το όνομά μας τελειώνει σε "-akis", "-idis", "-eas" και να επιστρέφει τον αντίστοιχο πιθανό τόπο καταγωγής (Κρήτη, Πόντος, Μάνη).

```
def origin(name):
  if name[-4:] == 'akis':
    return 'Kriti'
  elif name[-4:] == 'idis':
    return 'Pontos'
  elif name[-3:] == 'eas':
    return 'Mani'
  else:
    return 'Unknown'

name = input('Give name: ')
print('Origin of name \'', name, '\' is', origin(name))
```

9. Ένα σώμα εκτοξεύεται προς τα επάνω με αρχική ταχύτητα $v_0 = 10m/sec$. Γράψτε μια συνάρτηση η οποία θα δίνει τη θέση του σώματος y σε μία χρονική στιγμή t. Τυπώστε τη θέση του σώματος σε διαδοχικές χρονικές στιγμές και προσπαθήστε να εντοπίσετε (κατά προσέγγιση) σε πόσο χρόνο το σώμα θα επιστρέψει στο σημείο από το οποίο ξεκίνησε. Συγκρίνετε με την ακριβή τιμή.

```
def h(t):
  return 10 * t - 0.5 * 9.81 * t ** 2 # x = u0t - (1/2)gt^2

for t in range(5):
  print('Position at t =', t, ':', h(t))

# h(t) = 0 <=> 10t - (1/2) * 9.81t^2 = 0 <=>
# t = 0 or t = 2*10 / 9.81 ~= 2.0387
```

10. Σωμάτιο κινείται κατά μήκος του άξονα x και επιβραδύνεται από δύναμη τριβής με επιτάχυνση $a = -\kappa v^2$, $\kappa > 0$. Αν η αρχική θέση του είναι $x(t = 0) = 0$ και η αρχική του ταχύτητα είναι $v(t = 0) = v_0$, τότε η ταχύτητά του και η θέση του, ως συναρτήσεις του χρόνου είναι

$$v(t) = \frac{v_0}{\kappa v_0 t + 1}, \qquad x(t) = \frac{1}{\kappa} \ln(\kappa v_0 t + 1)$$

Γράψτε συναρτήσεις οι οποίες θα υπολογίζουν τα x, v για διαδοχικούς χρόνους. Τυπώστε στην οθόνη τα t, x, v. Επίσης, βρείτε έναν τρόπο για να παρατηρήσετε ότι η θέση και ταχύτητα συνδέονται με τη σχέση $v(x) = v_0 e^{-\kappa x}$.

```
u0 = float(input('Give u0: '))
k = float(input('Give k: '))

def u(t):
  return u0 / (k*u0*t + 1)

import math
def x(t):
  return math.log(k*u0*t + 1) / k

for t in range(100):
  print('t =', t, ', u =', u(t), ', x =', x(t))

# x(t) = (1/k) * ln(k*u0*t + 1) <=> k*x(t) = ln(k*u0*t + 1) <=>
# e ^ (k * x(t)) = k*u0*t + 1
# u(x) = u0 / e ^ (k * x(t)) = u0 * e ^ (- k * x(t))
```