ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ
ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΠΟΛΙΤΙΚΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΟΜΕΑΣ ΔΟΜΟΣΤΑΤΙΚΗΣ

Ακαδ. Έτος: 2016-2017

Μάθημα: Εφαρμογές Η/Υ

Δευτέρα, 12/12/2016

Διδάσκοντες:
Ν.Δ. Λαγαρός (Επικ. Καθηγητής), Αθ. Στάμος (ΕΔΙΠ), Χ. Φραγκουδάκης (ΕΔΙΠ)

Παραδείγματα για την 11η παράδοση – Γεννήτριες, διαδρομές

1. Επανάληψη ανά τριάδες και ζεύγη μίας ακολουθίας

Να συνταχθεί γεννήτρια (generator) η οποία να επιστρέφει διαδοχικές τριάδες από άλλη γεννήτρια. Πχ αν μία γεννήτρια επιστρέφει τα στοιχεία 10, 20, 22, 33, 40 η ζητούμενη γεννήτρια να επιστρέφει τις πλειάδες (10,20,22), (20,22,33), (22,33,40). Να συνταχθεί παρόμοια γεννήτρια που να επιστρέφει διαδοχικά ζεύγη.

Λύση

Αρχικά το αντικείμενο της ακολουθίας, το οποίο μπορεί να μην είναι γεννήτρια, θα μετατραπεί σε γεννήτρια (στην πραγματικότητα σε iterator) με τη συνάρτηση iter(), για να μπορεί να χρησιμοποιηθεί η συνάρτηση next(). Παρεμπιπτόντως η συνάρτηση iter(x) όταν το x είναι γεννήτρια επιστρέφει το ίδιο το x. Τα πρώτα 2 στοιχεία της ακολουθίας θα ληφθούν με next() και θα αποθηκευτούν σε μεταβλητές. Στη συνέχεια κάθε ένα από τα υπόλοιπα θα ληφθεί με for και θα επιστρέφεται αυτό και τα δύο προηγούμενα. Παρομοίως για τα ζεύγη.

```
def iterby3(seq):
    "Iterate consequtive triplets."
    it = iter(seq)
    a = next(it)
    b = next(it)
    for c in it:
        yield (a, b, c)
        a, b = b, c

def iterby2(seq):
    "Iterate consequtive pairs."
    it = iter(seq)
    a = next(it)
    for b in it:
        yield (a, b)
        a = b

def gen(): yield 12; yield 20; yield 34; yield 40; yield 100
for i in iterby3(gen()):
    print(i)
```

Στο παραπάνω πρόγραμμα περιλαμβάνεται και παράδειγμα.

2. Υπολογισμός εμβαδού μη κυρτού πολυγώνου

Το εμβαδόν μη κυρτού πολυγώνου n κορυφών δίνεται από:

$$A=\frac{1}{2}\left|\sum_{i=1}^{n}(x_{i+1}-x_i)(y_{i+1}+y_i)\right| \quad , \quad y_i \geq 0$$

όπου x_i, y_i οι συντεταγμένες των κορυφών, και γίνεται σύμβαση ότι $x_{n+1}=x_1$, $y_{n+1}=y_1$ και πρέπει να ισχύει $y_i \geq 0$. Να συνταχθεί συνάρτηση σε python η οποία παίρνει ως όρισμα ακολουθία από ζεύγη x,y και να επιστρέφει το εμβαδόν.

Λύση

Χρειάζεται μία γεννήτρια σαν την iterby2 του προηγούμενου παραδείγματος, με τη διαφορά ότι πρέπει να επιστρέφει και το ζεύγος (n, 1):

```
def iterby2c(seq):
    "Iterate consequtive pairs."
    it = iter(seq)
    a = first = next(it)
    for b in it:
        yield (a, b)
        a = b
    yield (b, first)
```

Ο υπολογισμός εμβαδού γίνεται τώρα πολύ εύκολα:

```
def area(c):
    "Compute area of non-convex polygon."
    a = 0.0
    for (x1,y1), (x2, y2) in iterby2c(c):
        a += (x2-x1)*(y2+y1)
    return abs(a)/2

c = (0,0), (10,0), (10,10), (0, 10)
print(area(c))
```

Ο κώδικας περιέχει και παράδειγμα το οποίο δίνει εμβαδόν 100.

3. Πύκνωση σημείων σε γραμμικό στοιχείο

Ένα γραμμικό στοιχείο αυθαίρετης γεωμετρίας (Free Form Linear Feature – FFLF) στην τοπογραφία είναι μία καμπύλη που ορίζεται από κορυφές (nodes). Οι κορυφές ενώνονται με αυθαίρετη συνάρτηση παρεμβολής (πχ γραμμική, κυβική, B-spline κλπ). Θεωρώντας τις κορυφές ως πλειάδες συντεταγμένων x, y και το FFLF ως λίστα τέτοιων πλειάδων, να συνταχθεί γεννήτρια (generator) στην python που να επιστρέφει σημεία πάνω στο FFLF με γραμμική παρεμβολή. Στα σημεία να περιλαμβάνονται και οι κορυφές, αλλά να μην υπάρχουν διπλά σημεία.

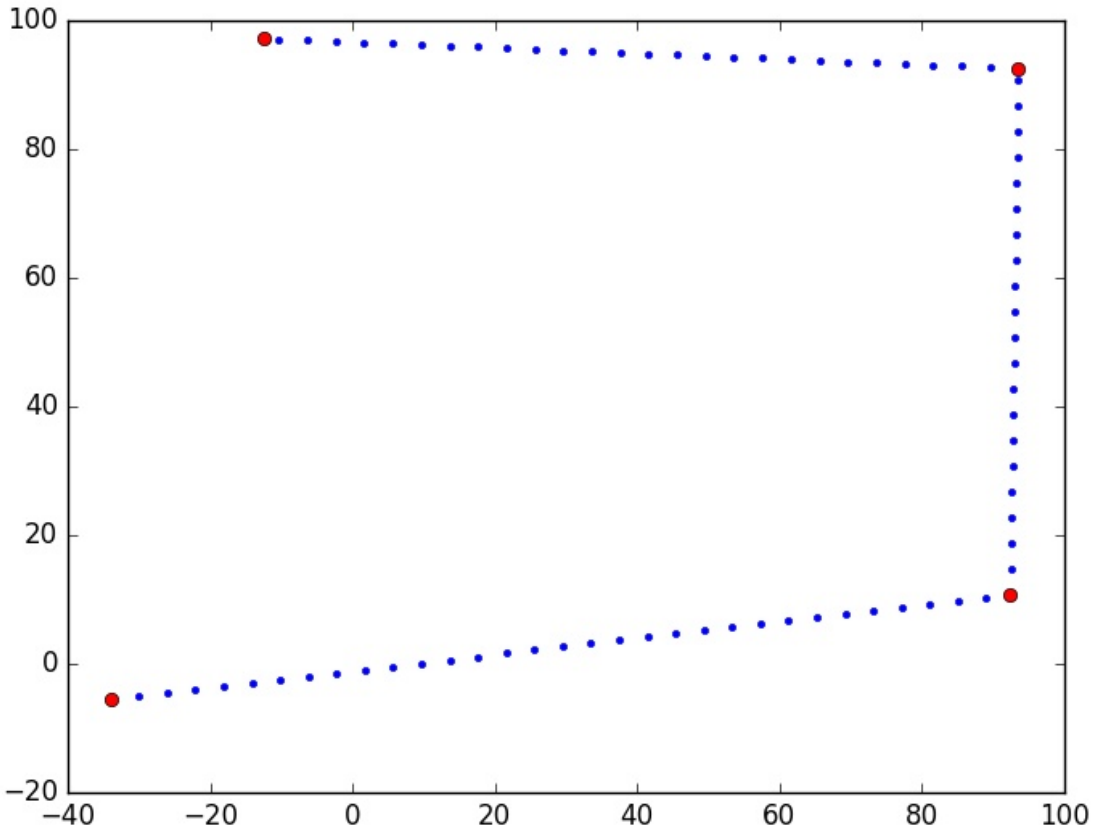
Λύση

Θα χρησιμοποιηθεί οι γεννήτρια iterby2() προηγούμενου παραδείγματος και η γεννήτρια frange() που αναφέρθηκε στη θεωρία. Σε κάθε ευθύγραμμο τμήμα επιστρέφονται η κορυφή αρχής του, τα εσωτερικά σημεία του αλλά όχι η κορυφή τέλους του, η οποία είναι η κορυφή αρχής του επόμενου σημείου. Η κορυφή τέλους του τελευταίου ευθ. τμήματος επιστρέφεται ξεχωριστά. Σε κάθε ευθ. τμήμα μήκους d12, γίνεται βρόχος από d=0 (κορυφή αρχής) έως d=d12 μη συμπεριλαμβανομένου του d12. Οι συντεταγμένες σε απόσταση d βρίσκονται με γραμμική παρεμβολή.

```
from math import hypot

def iterdis2(a, dd):
    "Iterate through polyline a, returning a point every d units distance."
    for (xa,ya), (xb,yb) in iterby2(a):
        d = hypot(xb-xa, yb-ya)
        for d1 in frange(0.0, d, dd):
            x = xa + (xb-xa)/d*d1
```

```
            y = ya + (yb-ya)/d*d1
            yield x, y
    yield xb, yb

def iterby2(seq):
    "Iterate consequtive pairs."
    it = iter(seq)
    a = next(it)
    for b in it:
        yield (a, b)
        a = b

def frange(xa, xb, dx):
    "Like range but reals."
    if dx == 0.0: raise ValueError("Zero increment")
    x = xa
    if dx >= 0:
        while x < xb:
            yield x
            x += dx
    else:
        while x > xb:
            yield x
            x += dx

def example():
    "Test the program."
    import numpy as np, itertools as it
    from matplotlib import pyplot as plt
    seq = np.array([ (-33.962,         -5.516),
            ( 92.433,         10.739),
            ( 93.554,         92.573),
            (-12.382,         97.197),
          ])
    t = np.fromiter(it.chain.from_iterable(iterdis2(seq, 4.0)), np.float)
    plt.figure()
    plt.plot(t[0::2], t[1::2], ".")
    plt.plot(seq[:, 0], seq[:, 1], "ro")
    plt.show()

example()
```

Στο παράδειγμα, η συνάρτηση np.fromiter() δημιουργεί μονοδιάστατο μητρώο από γεννήτρια. Έτσι χρησιμοποιείται η it.chain.from_iterable() η οποία μετατρέπει ακολουθία από πλειάδες σε ακολουθία από αριθμούς. Στο μητρώα τα μονά στοιχεία (1, 3, 5...) είναι τα x και τα ζυγά (0, 2, 4...) είναι τα y. Παρακάτω φαίνεται διάγραμμα με τα σημεία. Ας σημειωθεί ότι η iterdis2(), εκτός από λίστα πλειάδων, δέχεται και διδιάστατο μητρώο, λίστα λιστών και πλειάδα πλειάδων.


100
80
60
40
20
0
−20
−40
−20
0
20
40
60
80
100

4. Υπολογισμός γωνίας σε πολυγωνική οδού

Οι συντεταγμένες των κορυφών πολυγωνικής οδού δίνεται σε λίστα από πλειάδες (x, y) ή διδιάστατο μητρώο. Να συνταχθεί πρόγραμμα σε python που να υπολογίζει και να τυπώνει τις δεξιόστροφες γωνίες κάθε καμπύλης, δηλαδή κάθε εσωτερικής κορυφής.

Λύση

Θα χρησιμοποιηθεί η γεννήτρια iterby3() προηγούμενου παραδείγματος. Η δεξιόστροφη γωνία β δίνεται:

$$\beta_k = \alpha_{k,k+1} - \alpha_{k-1,k} - \pi + \lambda\, 2\pi$$

όπου $\alpha_{k,k+1}$ είναι η γωνία διεύθυνσης του προσανατολισμένου ευθ. τμήματος που ενώνει την κορυφή k με την κορυφή k+1 και δίνεται:

$$\alpha_{k,k+1} = \text{atan2}(x_{k+1}-x_k,\ y_{k+1}-y_k)$$

Ο ακέραιος λ έχει την τιμή που χρειάζεται για να ισχύει:

$$0 \le \beta_k < 2\pi$$

Στην python αρκεί να πάρουμε το υπόλοιπο της διαίρεσης της $\beta_κ$ δια 2π.

```
from math import atan2, pi

def polyg(c):
    "Υπολογισμός γωνιών από συντεταγμένες κορυφών πολυγωνικής."
    for (x1,y1), (x2,y2), (x3,y3) in iterby3(c):
        a12 = atan2(x2-x1, y2-y1)
        a23 = atan2(x3-x2, y3-y2)
        b = a23 - a12 - pi
        b %= 2*pi
        print(b*180/pi, "deg")

def iterby3(seq):
    "Iterate consequtive triplets."
    it = iter(seq)
    a = next(it)
    b = next(it)
    for c in it:
        yield (a, b, c)
        a, b = b, c

def example():
    "Test the program."
    seq = [ (-33.962,         -5.516),
            ( 92.433,         10.739),
            ( 93.554,         92.573),
            (-12.382,         97.197),
          ]
    polyg(seq)

example()
```

5. Υπολογισμός συνολικού μήκους αγωγών από διαφορετικά αρχεία

Σε φάκελλο που δίνεται από το χρήστη υπάρχουν πολλοί υποφάκελλοι ο καθένας από τους οποίους περιέχει αρχείο με κατάληξη .sad που περιέχει συντεταγμένες αγωγού αποχέτευσης με τη μορφή:

```
Σ-ΔΗΛ1
Κ0              16022.185      29616.203          3.070
Π1              16031.033      29608.640          3.000
Π2              16035.525      29604.800          2.830
...
Π87             16428.977      28312.580          0.470
Κ35             16433.710      28301.852          0.370
ΚΤ              16440.221      28305.981          0.800
$
```

Να συνταχθεί πρόγραμμα σε Python το οποίο:

α. Να διαβάζει με GUI το όνομα του φακέλου από το χρήστη.

β. Να διαβάζει το όνομα και τις συντεταγμένες κάθε αγωγού.

γ. Να υπολογίζει και να τυπώνει το μήκος του αγωγού.

δ. Να υπολογίζει και να τυπώνει το συνολικό μήκος των αγωγών.

ε. Να σχεδιάζει τους αγωγούς σε γράφημα, με διαφορετικό χρώμα τους συλλεκτήριους (η ονομασία τους αρχίζει από Σ) και με διαφορετικό χρώμα τους δευτερεύοντες (όλοι οι άλλοι).
Δοκιμάστε το πρόγραμμα με δεδομένα που υπάρχουν στη διεύθυνση:
http://users.ntua.gr/stamthan/efarmogeshy/paradeigmata11/

Λύση
Για κάθε αγωγό θα συνταχθεί ένα αντικείμενο που θα αποθηκεύει τις συντεταγμένες και το όνομά του, θα υπολογίζει μήκος και θα σχεδιάζει τον εαυτό του. Για κάθε ένα από τα ερωτήματα θα συνταχθεί μία ξεχωριστή συνάρτηση.
Μία μέθοδος του αντικειμένου δέχεται ένα όνομα αρχείου και διαβάζει όνομα και συντεταγμένες από αυτό. Άλλες μέθοδοι υπολογίζουν το μήκος και σχεδιάζουν τον αγωγό.

```
import sys
from pathlib import Path
import tkinter as tk
import tkinter.filedialog
import numpy as np
from matplotlib import pyplot as plt

class Agogos:
    def readFile(self, fn):
        "Read the name and coordinates of a pipe."
        fr = fn.open()
        self.name = next(fr).strip()
        coor = []
        for dline in fr:
            if dline.strip() == "$": break
            dl = dline.split()
            coor.append((float(dl[1]), float(dl[2])))
        self.coor = np.array(coor)
        fr.close()

    def length(self):
        "Compute the length of the pipe."
        dxy = self.coor[1:, :] - self.coor[:-1, :]
        return self.name, sum(np.hypot(dxy[:, 0], dxy[:, 1]))

    def plot(self):
        "Plot the pipe."
        if self.name.startswith("Σ"):
            col = "r"
        else:
            col = "k"
        plt.plot(self.coor[:, 0], self.coor[:, 1], col)
        n = len(self.coor)//2
        plt.arrow(self.coor[n, 0], self.coor[n, 1], self.coor[n+1, 0]-self.coor[n, 0],
            self.coor[n+1, 1]-self.coor[n, 1],
            shape='full', lw=0, length_includes_head=True, head_width=50.05)
```

Στη συνάρτηση που διαβάζει το όνομα του φακέλλου με GUI από το χρήστη, η μέθοδος update() χρειάζεται διότι δεν καλείται η mainloop() και έτσι δεν προλαβαίνει το GUI να αρχικοποιηθεί πριν του ζητήσουμε το διάλογο επιλογής φακέλλου.

```
def getMaindir():
    "Get the name of the directory from the user."
    root = tk.Tk()
    root.update()
    dn = tk.filedialog.askdirectory(initialdir=Path.cwd(), mustexist=True,
        title="Choose directory")
    root.destroy()
    return dn
```

Στη συνέχεια μία συνάρτηση βρίσκεις όλους τους υποφακέλλους και σε κάθε φάκελλο ψάχνει για αρχεία με κατάληξη .sad και αν βρει περισσότερα από ένα, διαβάζει μόνο το πρώτο.

```
def readAgogs(dnmain):
    "Read all the pipes."
```

```
    agogs = []
    for dn in dnmain.glob("*"):
        if not dn.is_dir(): continue
        for fn in dn.glob("*.sad"):
            if not fn.is_file(): continue
            print(fn)
            ag = Agogos()
            ag.readFile(fn)
            agogs.append(ag)
            break
    return agogs
```

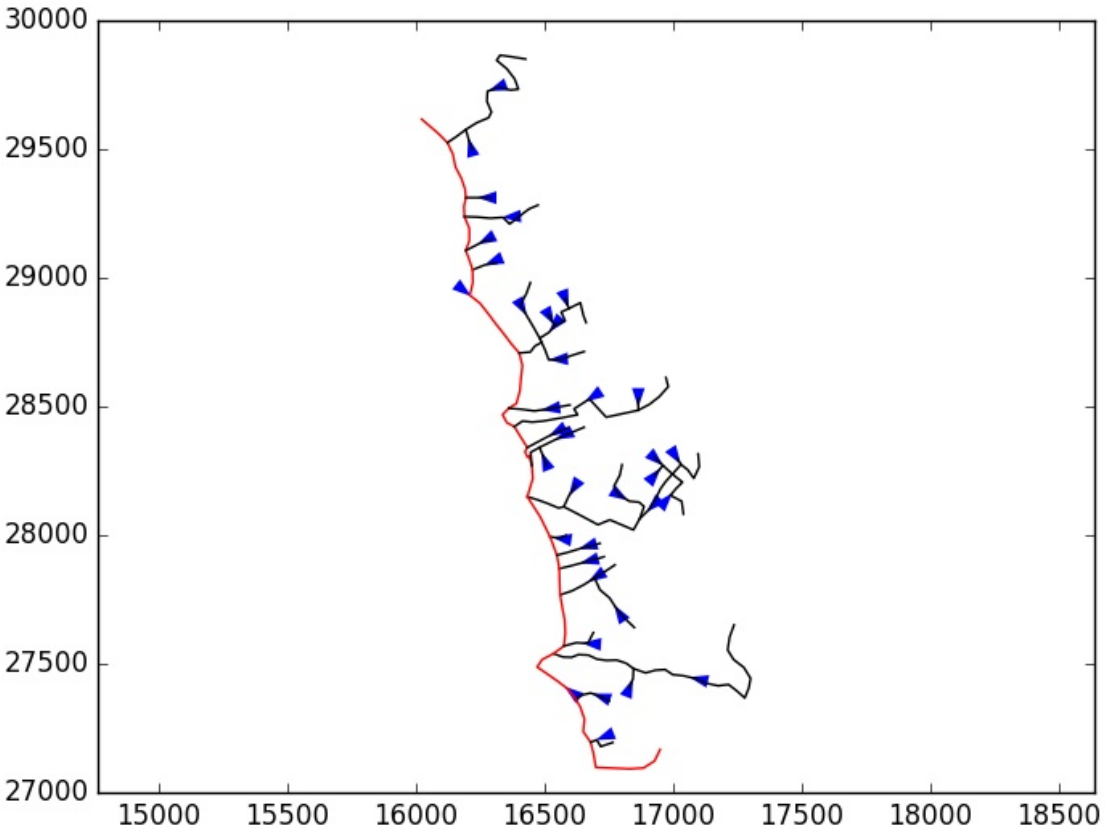
Οι επόμενες συναρτήσεις υπολογίζουν τα μήκη και κάνουν τη σχεδίαση

```
def lengths(agogs):
    "Compute lengths:"
    print("Αγωγός                        Μήκος")
    sL = 0.0
    for ag in agogs:
        name, L = ag.length()
        print("{:20s}{:15.3f}".format(name, L))
        sL += L
    print("{:20s}{:15.3f}".format("ΑΘΡΟΙΣΜΑ:", sL))

def plot(agogs):
    "Plot the pipes."
    plt.figure()
    plt.hold(True)
    for ag in agogs:
        ag.plot()
    plt.axis("equal")
    plt.show()
```

Τέλος η pyMain() καλεί όλες τις άλλες συναρτήσεις. Αν η μεταβλητή dnmain περιέχει None, ο χρήστης ακύρωσε το πρόγραμμα, και το πρόγραμμα τερματίζεται καλώντας τη συνάρτηση exit() της βιβλιοθήκης sys.

```
def pyMain():
    "Start here."
    dnmain = getMaindir()
    if dnmain is None: sys.exit()
    agogs = readAgogs(Path(dnmain))
    lengths(agogs)
    plot(agogs)

pyMain()
```

Το πρόγραμμα κάνει το επόμενο γράφημα:


30000
29500
29000
28500
28000
27500
27000
15000
15500
16000
16500
17000
17500
18000
18500