ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ
ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΠΟΛΙΤΙΚΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΟΜΕΑΣ ΔΟΜΟΣΤΑΤΙΚΗΣ

Ακαδ. Έτος: 2022-2023

Πέμπτη, 25/5/2023

Μάθημα: Προγραμματισμός Η/Υ

Διδάσκοντες: Ν.Δ. Λαγαρός (Καθηγητής), Α. Στάμος (ΕΔΙΠ), Χ. Φραγκουδάκης (ΕΔΙΠ)
Αμβ. Σαββίδης (Δρ)

Παραδείγματα για την 9η 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.

Στην περίπτωση της iterby2() αν το seq έχει μόνο ένα στοιχείο, η iterby2 δεν θα επιστρέψει κανένα ζεύγος. Αν το seq δεν έχει κανένα στοιχείο, η next() θα καταθέσει την εξαίρεση StopIteration. Η γεννήτρια συλλαμβάνει την εξαίρεση (με try/except) και στην συνέχεια τερματίζει την εκτέλεση της (με return). Δηλαδή λειτουργεί σαν να είχαμε γράψει:

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

Συνεπώς αν το seq δεν έχει κανένα στοιχείο (ή μόνο 1), η iterby2() δεν θα επιστρέψει κανένα ζεύγος. Ομοίως αν στην iterby3 το seq έχει μέχρι 2 στοιχεία, η iterby3() δεν θα επιστρέψει καμία τριάδα.

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 consecutive 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

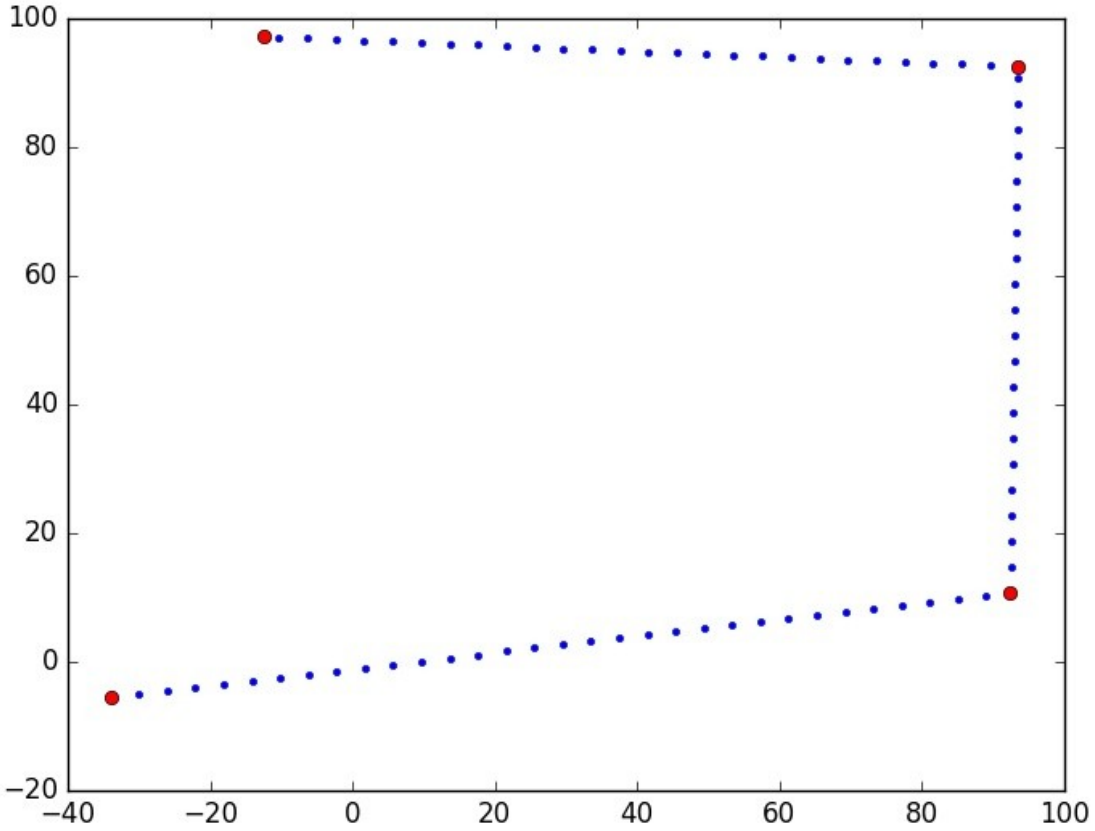
Ένα γραμμικό στοιχείο αυθαίρετης γεωμετρίας (Free Form Linear Feature – FFLF) στην τοπογραφία είναι μία καμπύλη που ορίζεται από κορυφές (nodes). Οι κορυφές ενώνονται με αυθαίρετη συνάρτηση παρεμβολής (πχ γραμμική, κυβική, B-spline κλπ). Θεωρώντας τις κορυφές ως πλειάδες συντεταγμένων x, y και το FFLF 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 β_k 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. Μετονομασία πολλαπλών αρχείων

Να συνταχθεί συνάρτηση που σε δεδομένο φάκελο μετονομάζει αρχεία που το όνομά τους καταλήγει σε suf1, σε αρχεία που αντί για suf1 καταλήγουν σε suf2.

Λύση

Φτιάχνουμε συνάρτηση που βρίσκει όλα τα αρχεία (όχι υποφακέλους) στο δεδομένο φάκελο χρησιμοποιώντας τη βιβλιοθήκη pathlib. Ελέγχουμε αν κάθε αρχείο καταλήγει σε suf2 και αν ναι, το μετονομάζουμε.

```
from pathlib import Path

def rensuf(dn, suf1, suf2):
    "Rename multiple files in directory dn, which end with suf1."
    dn = Path(dn)
    ns = len(suf1)
    for fn in dn.glob("*"):
        if fn.is_dir(): continue
        t = fn.name
        if t[len(t)-ns:] != suf1: continue
        t2 = t[:len(t)-ns] + suf2    #Εδώ δεν λειτουργεί το t[:-ns] γιατί πιθανώς ns==0
        fn2 = fn.with_name(t2)
        fn.rename(fn2)
```

Αρχικά μετατρέπουμε το φάκελο dn σε αντικείμενο Path, για ο χρήστης μπορεί να τον δώσει ως συμβολοσειρά.
Όπως γνωρίζουμε στην Python αντί για `t[len(t)-ns:]` μπορούμε να γράψουμε `t[-ns:]` που σημαίνει το ίδιο πράγμα με την προϋπόθεση ότι το `-ns` είναι αρνητικός αριθμός. Όμως εδώ το ns μπορεί να είναι μηδέν, οπότε πρέπει να γραφεί αναγκαστικά ως `t[len(t)-ns:]`.
Αν suf1=="", τότε θα επιλεγούν όλα τα αρχεία του φακέλου.

Βελτίωση

Η Python έχει μέθοδο που ελέγχει αν μία συμβολοσειρά καταλήγει σε μία άλλη, και έτσι μπορούμε να κάνουμε τον κώδικα πιο ευανάγνωστο:

```
from pathlib import Path

def rensuf(dn, suf1, suf2):
    "Rename multiple files in directory dn, which end with suf1."
    dn = Path(dn)
    ns = len(suf1)
    for fn in dn.glob("*"):
        if fn.is_dir(): continue
        t = fn.name
        if not t.endswith(suf1): continue
        t2 = t[:len(t)-ns] + suf2    #Εδώ δεν λειτουργεί το t[:-ns] πιθανώς ns==0
        fn2 = fn.with_name(t2)
        fn.rename(fn2)
```

Έλεγχος

Μπορούν να γίνουν πολλά λάθη. Καταρχήν αν το τελικό όνομα ενός αρχείου βγει κενό πρέπει να τυπώνεται μήνυμα λάθους. Αν το τελικό όνομα του αρχείου είναι ήδη κατειλημμένο από άλλο αρχείο (ή φάκελο) επίσης πρέπει να τυπώνεται μήνυμα λάθους. Τέλος η μετονομασία μπορεί να αποτύχει (πχ το τελικό όνομα αρχείου έχει χαρακτήρες που δεν υποστηρίζονται από το file system, ή δεν έχουμε δικαιώματα να μετονομάσουμε το αρχείο κλπ). Αν αποτύχει πρέπει να τυπώνεται πάλι μήνυμα λάθους.
Μετά από ένα μήνυμα λάθους συνεχίζουμε με τα επόμενα αρχεία αντί να σταματάμε το πρόγραμμα.

```
from pathlib import Path

def rensuf(dn, suf1, suf2):
    "Rename multiple files in directory dn, which end with suf1."
    dn = Path(dn)
    ns = len(suf1)
    for fn in dn.glob("*"):
```

```
        if fn.is_dir(): continue
        t = fn.name
        if not t.endswith(suf1): continue

        t2 = t[:len(t)-ns] + suf2    #Εδώ δεν λειτουργεί το t[:-ns] πιθανώς ns==0
        if t2.strip() == "":
            print("Can not rename {} to blank".format(t))
            continue

        fn2 = fn.with_name(t2)
        if fn2.exists():
            print("Can not rename {} to {}: already exists".format(t, t2))
            continue

        try:
            fn.rename(fn2)
        except Exception as e:
            print("Could not rename {} to {}: {}".format(t, t2, str(e)))
```

GUI

Η συνάρτηση μπορεί να χρησιμοποιηθεί σε πρόγραμμα που έχει GUI. Σε αυτή την περίπτωση τα μηνύματα λάθος δεν πρέπει να γράφονται σε terminal αλλά πιθανότατα πρέπει να γράφονται σε κάποιο παράθυρο/widget. Αυτό μπορεί να γίνει με μία συνάρτηση prt(t) που τυπώνει το όρισμα εισόδου t σε κάποιο παράθυρο. Τη συνάρτηση αυτή πρέπει να τη δώσει ο χρήστης στη συνάρτηση rensuf() ως όρισμα εισόδου. Το όρισμα εισόδου θα έχει ως προεπιλεγμένη τιμή τη συνάρτηση print() της Python:

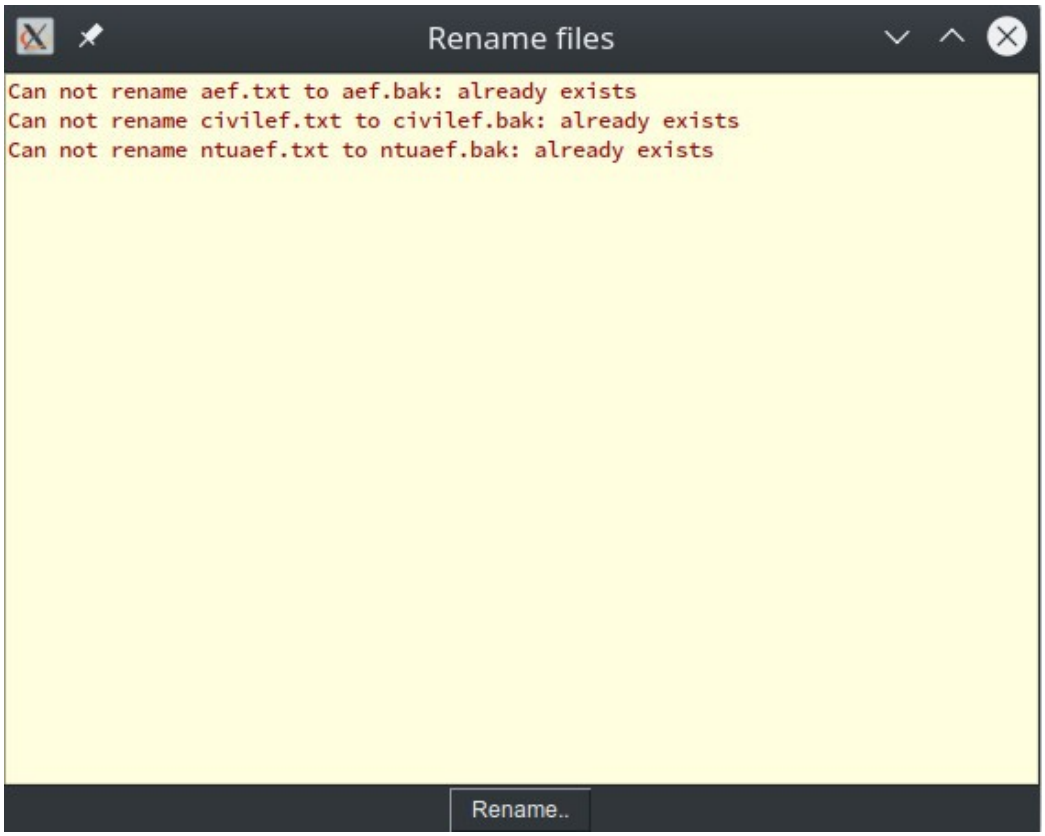
```
from pathlib import Path

def rensuf(dn, suf1, suf2, prt=print):
    "Rename multiple files in directory dn, which end with suf1."
    dn = Path(dn)
    ns = len(suf1)
    for fn in dn.glob("*"):
        if fn.is_dir(): continue
        t = fn.name
        if not t.endswith(suf1): continue

        t2 = t[:len(t)-ns] + suf2    #Εδώ δεν λειτουργεί το t[:-ns] πιθανώς ns==0
        if t2.strip() == "":
            prt("Can not rename {} to blank".format(t))
            continue

        fn2 = fn.with_name(t2)
        if fn2.exists():
            prt("Can not rename {} to {}: already exists".format(t, t2))
            continue

        try:
            fn.rename(fn2)
        except Exception as e:
            prt("Could not rename {} to {}: {}".format(t, t2, str(e)))
```

GUI

Για λόγους πληρότητας, δίνεται παρακάτω παράδειγμα συνάρτησης (μεθόδου) prtLog() που τυπώνει σε ένα Text widget:

```
import tkinter as tk
from rensuf4 import rensuf

class Mywin:
    "A window."
    def __init__(self, root):
        "Make widgets."
        root.title("Rename files")
        self.txtLog = tk.Text(root, bg="lightyellow", fg="darkred")
        self.txtLog.grid(row=0, column=0, sticky="wesn")
        but = tk.Button(root, text="Rename..", command=self.rename)
        but.grid(row=1, column=0)
```

```
        root.columnconfigure(0, weight=1)
        root.rowconfigure(0, weight=1)

    def prtLog(self, t):
        "Print something to the txtLog widget."
        self.txtLog.insert(tk.END, t+"\n")

    def rename(self):
        "Rename example."
        rensuf(".", "ef.txt", "ef.bak", self.prtLog)

root = tk.Tk()
win = Mywin(root)
root.mainloop()
```



6. Υπολογισμός συνολικού μήκους αγωγών από διαφορετικά αρχεία

Σε φάκελλο που δίνεται από το χρήστη υπάρχουν πολλοί υποφάκελοι 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 στην ιστοσελίδα helios.

Λύση

Φτιάχνουμε συνάρτηση που διαβάζει το όνομα του φακέλου με GUI από το χρήστη. Στην συνάρτηση 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
```

Για την εύρεση όλων των υποφακέλων και όλων των αρχείων .sad στους υποφακέλους, θα χρησιμοποιήσουμε την pathlib.

```
def sadFiles(dnmain):
    "Search all subdirectories."
    fns = []
    for dn in dnmain.glob("*"):       #Find all objects (files/directories) of dnmain
        if not dn.is_dir(): continue  #Ignore files
        print(dn.name)
        for fn in dn.glob("*.sad"):        #Find all objects of dn, with ".sad" suffix
            if not fn.is_file(): continue  #Ignore directories
            fns.append(fn)
    return fns
```

Επειδή το πλήθος αρχείων είναι αυθαίρετα μεγάλο, και επειδή δεν χρειάζονται όλα τα αρχεία την ίδια στιγμή, θα αντικαταστήσουμε την (επιστρεφόμενη) λίστα με γεννήτρια:

```
def sadFiles(dnmain):
    "Search all subdirectories."
    for dn in dnmain.glob("*"):       #Find all objects (files/directories) of dnmain
        if not dn.is_dir(): continue  #Ignore files
        print(dn.name)
        for fn in dn.glob("*.sad"):        #Find all objects of dn, with ".sad" suffix
            if not fn.is_file(): continue  #Ignore directories
            yield fn
```

Η ανάγνωση ενός αρχείου αγωγού γίνεται με μια συνάρτηση. Ο έλεγχος των δεδομένων αφήνεται ως άσκηση στον αναγνώστη. Η μεταβλητή fn είναι αντικείμενο Path.

```
def readAgog(fn):
    "Read pipe from file."
    fr = fn.open()
    nam = next(fr).strip()
    x = []
    y = []
    for dline in fr:
        if dline.strip() == "$": break
        dl = dline.split()
        x.append(float(dl[1]))
        y.append(float(dl[2]))
    return nam, x, y
```

Κάθε αρχείο που επιστρέφει η sadFiles(), το διαβάζουμε, υπολογίζουμε το μήκος του αγωγού και σχεδιάζουμε τον αγωγό. Ταυτόχρονα υπολογίζουμε το συνολικό μήκος των αγωγών. Η συνάρτηση zip() είναι γεννήτρια και επιστρέφει ζεύγη c=(x, y), δηλαδή (x[0], y[0]), (x[1], y[1]), (x[2], y[2]), κλπ. Η iterby2() είναι η γεννήτρια της 1ης άσκησης που επιστρέφει διαδοχικά ζεύγη δηλαδή (c[0], c[1]), (c[1], c[2]), (c[2], c[3]), κλπ. Ήτοι: (x[0], y[0]), (x[1], y[1])), (x[1], y[1]), (x[2], y[2])), (x[2], y[2]), (x[3], y[3])), κλπ.

```
def agog(dnmain):
    "Find pipe lengths and plot."
    plt.figure()
    #plt.hold(True)  #Matplotlib newer versions do not have hold() (hold is always True)
    ss = 0.0
    for fn in sadFiles(dnmain):
        nam, x, y = readAgog(fn)
        itxy = iterby2(zip(x, y))
        s = sum(hypot(x2-x1, y2-y1) for (x1,y1),(x2,y2) in itxy)
        print("    ", fn.name, ":", nam, s)
        ss += s
        if nam.startswith("Σ"): c = "r"
        else:                   c = "b"
        plt.plot(x, y, c)
    print("Συνολικό μήκος:", ss)
    plt.axis("equal")
    plt.show()
```

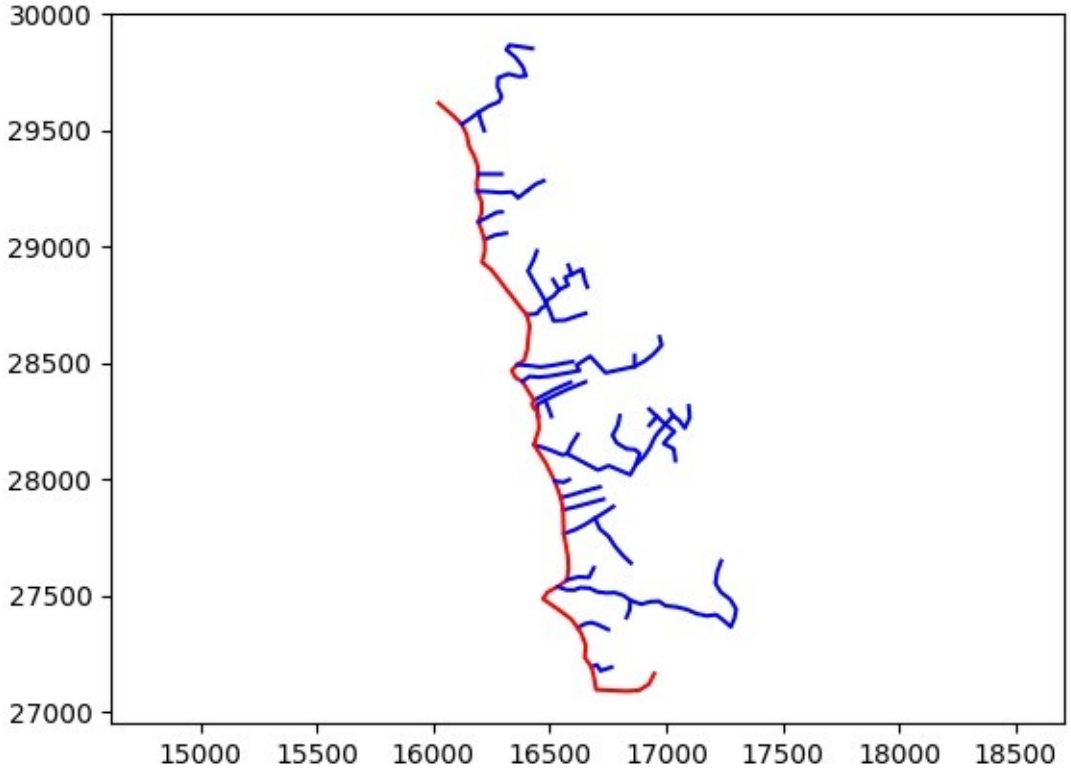
Τέλος η pyMain() καλεί όλες τις άλλες συναρτήσεις. Αν η μεταβλητή dnmain περιέχει None ή μία κενή συμβολοσειρά, ο χρήστης ακύρωσε το πρόγραμμα, και το πρόγραμμα τερματίζεται καλώντας τη συνάρτηση exit() της βιβλιοθήκης sys.

```
def pyMain():
    "Start here."
    dnmain = getMaindir()
    if dnmain is None or dnmain == "": sys.exit()  #user cancelled
    agog(Path(dnmain))

pyMain()
```

Το πρόγραμμα χρησιμοποιεί πολλές βιβλιοθήκες τις οποίες πρέπει να φορτώσουμε στο πρόγραμμα:

```
import sys
from math import hypot
from pathlib import Path
import tkinter as tk
import tkinter.filedialog
from matplotlib import pyplot as plt
```

Το πρόγραμμα κάνει το επόμενο γράφημα:


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