ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ
ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΠΟΛΙΤΙΚΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΟΜΕΑΣ ΔΟΜΟΣΤΑΤΙΚΗΣ

Ακαδ. Έτος: 2016-2017

Μάθημα: Εφαρμογές Η/Υ

Δευτέρα, 31/10/2016

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

Παραδείγματα για την 5η παράδοση – Αντικειμενοστραφής προγραμματισμός

1. Υπολογισμός ολικού μητρώου δυσκαμψίας συνδεδεμένων υποστυλωμάτων

Να συνταχθεί πρόγραμμα σε Python το οποίο:
1. Να διαβάζει από το αρχείο k.txt το μονοδιάστατο μητρώο [K] (δυσκαμψίες υποστυλωμάτων) το οποίο έχει αυθαίρετο πλήθος στοιχείων N.
2. Να δημιουργεί το ολικό μητρώο δυσκαμψίας ως φαίνεται στο παρακάτω παράδειγμα (για N=5):

$$[Kol]=\begin{bmatrix} K_1 & -K_1 & 0 & 0 & 0 & 0 \\ -K_1 & K_1+K_2 & -K_2 & 0 & 0 & 0 \\ 0 & -K_2 & K_2+K_3 & -K_3 & 0 & 0 \\ 0 & 0 & -K_3 & K_3+K_4 & -K_4 & 0 \\ 0 & 0 & 0 & -K_4 & K_4+K_5 & -K_5 \\ 0 & 0 & 0 & 0 & -K_5 & K_5 \end{bmatrix}$$

3. Να γράφει το ολικό μητρώο δυσκαμψίας στο αρχείο kol.txt.
4. Να δημιουργεί το μειωμένο μητρώο δυσκαμψίας [Km] το οποίο είναι το μητρώο [Kol] χωρίς την δεύτερη και προτελευταία γραμμή και χωρίς τη δεύτερη και προτελευταία στήλη.
5. Να τυπώνει το το [Km] στην οθόνη με τα νούμερα σε εκθετική μορφή.

Δοκιμάστε το πρόγραμμά σας με το αρχείο k.txt:

```
1e8
0.5e8
0.2e8
1.2e8
0.5e8
1e8
```

Λύση 1 με λίστες

Παρατηρούμε ότι το Kol έχει N+1 γραμμές και στήλες. Η πρώτη και τελευταία γραμμή είναι διαφορετικές απο τις ενιδάμεσες. Οι N-1 ενδιάμεσες γραμμές έχουν στη διαγώνιο του μητρώου Kol(i, i)=K_{i-1}+K_i και εκατέρωθεν της διαγωνίου τα -K_{i-1} και -K_i.

```
# -*- coding: iso-8859-7 -*-
def readK():
    "Διάβασε τις δυσκαμψίες."
    K = []
    fr = open("k.txt")
    for dline in fr:
        k1 = float(dline)
        K.append(k1)
    fr.close()
    return K

def ypolKol(K):
    "Υπολόγισε ολικό μητρώο δυσκαμψίας."
```

```
    Kol = []
    N = len(K)
    temp = [0.0 for i in range(N+1)]
    temp[0] = K[0]
    temp[1] = -K[0]
    Kol.append(temp)
    for i in range(1, N):
        temp = [0.0 for i in range(N+1)]
        temp[i-1] = -K[i-1]
        temp[i]   = K[i-1]+K[i]
        temp[i+1] = -K[i]
        Kol.append(temp)
    temp = [0.0 for i in range(N+1)]
    i = N
    temp[i-1] = -K[i-1]
    temp[i]   = K[i-1]
    Kol.append(temp)
    return Kol

def writeKol(Kol):
    "Εγγραφή Kol σε αρχείο."
    fw = open("kol.txt", "w")
    N = len(Kol)     #Τι μετράει το N εδώ;
    for i in range(N):
        for j in range(N): fw.write(" {}".format(Kol[i][j]))
        fw.write("\n")
    fw.close()

def reduceKol(Kol):
    "Δημιουργία μειωμένου μητρώου."
    Km = []
    N = len(Kol)
    for row in Kol:
        rown = row[0:1] + row[2:N-2] + row[N-1:N]
        Km.append(rown)
    del Km[N-2]
    del Km[1]
    return Km

def writeKm(Km):
    "Εκτύπωση Km στην οθόνη σε εκθετική μορφή."
    f = ""
    for i in range(len(Km)): f += "{:14.6e}"
    for row in Km:
        print(f.format(row[0], row[1], row[2], row[3], row[4]))

def pyMain():
    "Start here."
    K = readK()
    Kol = ypolKol(K)
    writeKol(Kol)
    Km = reduceKol(Kol)
    writeKm(Km)

pyMain()
```

Λύση με list comprehensions

Όπου μπορούμε χρησιμοποιούμε list comprehensions για να κάνουμε τον κώδικα πιο ευανάγνωστο (και πιο γρήγορο) και χρησιμοποιούμε τον πολλαπλασιασμό στις λίστες και στις συμβολοσειρές:

3*“ab” -> “abababab” και 4*[1, 2] -> [1, 2, 1, 2, 1, 2, 1, 2]

Επίσης κατά την κλήση μίας συνάρτησης χρησιμοποιούμε το σύμβολο * για να μετατρέψουμε μία λίστα σε ορίσματα. Για παράδειγμα αν a[1, 5, 7] τότε:

fun(*a) -> fun(1, 5, 7)

Τέλος η μέθοδος writelines() των αρχείων, γράφει στο αρχείο μία λίστα από συμβολοσειρές.

```
# -*- coding: iso-8859-7 -*-
def ypolKol(K):
    "Υπολόγισε ολικό μητρώο δυσκαμψίας."
    Kol = []
    N = len(K)
    temp = (N+1)*[0.0]
    temp[:2] = K[0], -K[0]
    Kol.append(temp)
    for i in range(1, N):
        temp = (N+1)*[0.0]
        temp[i-1:i+2] = -K[i-1], K[i-1]+K[i], -K[i]
        Kol.append(temp)
    temp = (N+1)*[0.0]
    i = N
    temp[i-1:] = -K[i-1], K[i-1]
    Kol.append(temp)
    return Kol

def writeKol(Kol):
    "Εγγραφή Kol σε αρχείο."
    fw = open("kol.txt", "w")
    f = len(Kol)*" {}" + "\n"
    fw.writelines([f.format(*row) for row in Kol])
    fw.close()

def reduceKol(Kol):
    "Δημιουργία μειωμένου μητρώου."
    N = len(Kol)
    Km = [row[0:1] + row[2:N-2] + row[N-1:N] for row in Kol]
    del Km[N-2]
    del Km[1]
    return Km

def writeKm(Km):
    "Εκτύπωση Km στην οθόνη σε εκθετική μορφή."
    f = len(Km)*"{:14.6e}"
    for row in Km:
        print(f.format(*row))

def pyMain():
    "Start here."
    K = [float(dline) for dline in open("k.txt")]
    Kol = ypolKol(K)
    writeKol(Kol)
    Km = reduceKol(Kol)
    writeKm(Km)

pyMain()
```

Λύση με μητρώα

Οι λίστες επειδή επειδή είναι πολύ γενικά αντικείμενα (πχ λίστες από λίστες, εισαγωγή/διαγραφή κλπ) είναι σχετικά αργές. Αν σε κάποιο πρόβλημα χρειαζόμαστε λίστες από αριθμούς με δυνατότητα μαθηματικών πράξεων, τότε η χρήση μητρώων είναι πιο γρήγορη και απαιτεί λιγότερη μνήμη. Η χρήση μητρών γίνεται περίπου όπως το Matlab.

```
# -*- coding: iso-8859-7 -*-
from numpy import loadtxt, savetxt, zeros, concatenate
def ypolKol(K):
    "Υπολόγισε ολικό μητρώο δυσκαμψίας."
```

```
    N = len(K)
    Kol = zeros((N+1, N+1))
    Kol[0, :2] = K[0], -K[0]
    for i in range(1, N):
        Kol[i, i-1:i+2] = -K[i-1], K[i-1]+K[i], -K[i]
    i = N
    Kol[i, i-1:] = -K[i-1], K[i-1]
    return Kol

def writeKol(Kol):
    "Εγγραφή Kol σε αρχείο."
    fw = open("kol.txt", "w")
    f = len(Kol)*" {}" + "\n"
    fw.writelines([f.format(*row) for row in Kol])
    fw.close()

def reduceKol(Kol):
    "Δημιουργία μειωμένου μητρώου."
    N = len(Kol)
     indexes = concatenate(([0], range(2,N-2), [N-1]))  #Ποιες στήλες/γραμμές
κρατάμε
    i = indexes.reshape(1, N-2)   #Μητρώο με 1 γραμμή και N-2 στήλες, είναι το
ίδιο μητρώο
    j = indexes.reshape(N-2, 1)   #Μητρώο με N-1 γραμμές και 1 στήλη, είναι το
ίδιο μητρώο
    return Kol[i, j]              #είναι αντίγραφο του μητρώου

def writeKm(Km):
    "Εκτύπωση Km στην οθόνη σε εκθετική μορφή."
    f = len(Km)*"{:14.6e}"
    for row in Km:
        print(f.format(*row))

def pyMain():
    "Start here."
    K = loadtxt("k.txt")
    Kol = ypolKol(K)
    savetxt("kol.txt", Kol)
    Km = reduceKol(Kol)
    writeKm(Km)

pyMain()
```

2. Υπολογισμός ολικού μητρώου δυσκαψίας από επί μέρους μητρώα δυσκαμψίας

Το μητρώο δυσκαμψίας του υποστυλώματος ij που δρα σαν μονοδιάστατη ράβδος και συνδέει τον κόμβο i με τον κόμβο j δίνεται:

$$[K_{ij}]=\frac{A_{ij}E_{ij}}{L_{ij}}\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}$$

όπου L, A, E το μήκος, το εμβαδόν διατομής και το μέτρο ελαστικότητας αντίστοιχα. Αν υπάρχουν N κόμβοι το ολικό μητρώο δυσκαμψίας [Kol] έχει διαστάσεις είναι (N+1)×(N+1). Το ολικό μητρώο δυσκαμψίας [Kol] αρχικά δημιουργείται με μηδενικά στοιχεία, και στη συνέχεια τα 4 στοιχεία του κάθε επί μέρους μητρώου K_{11}, K_{12}, K_{21}, K_{22} προστίθενται αντίστοιχα στα 4 στοιχεία του ολικού μητρώου Kol_{ii}, Kol_{ij}, Kol_{ji}, Kol_{jj}. Να συνταχθεί πρόγραμμα σε το οποίο:

1. Να διαβάζει από το αρχείο k.txt τα στοιχεία L, A, E, i, j των υποστυλωμάτων (ένα υποστύλωμα ανά σειρά αρχείου).
2. Να υπολογίζει το πλήθος N των κόμβων.
3. Να δημιουργεί και να τυπώνει με εκθετική μορφή και στοιχισμένα στο αρχείο k.txt το ολικό μητρώο δυσκαμψίας.

Δοκιμάστε το πρόγραμμά σας με το αρχείο k.txt:

```
1e8
```

```
0.5e8
0.2e8
1.2e8
0.5e8
1e8
```

Λύση

```
# -*- coding: iso-8859-7 -*-
from numpy import loadtxt, savetxt, zeros, ones, max

def ypolKol(s):
    "Υπολόγισε ολικό μητρώο δυσκαμψίας."
    N = max(s[:, 3:5]) + 1
    Kol = zeros((N, N))
    temp = ones((2, 2))
    temp[0, 1] = temp[1, 0] = -1
    for L, A, E, i, j in s:
        K = A*E/L * temp
        Kol[i, i] += K[0, 0]
        Kol[i, j] += K[0, 1]
        Kol[j, i] += K[1, 0]
        Kol[j, j] += K[1, 1]
    return Kol

def pyMain():
    "Start here."
    s = loadtxt("k.txt")
    Kol = ypolKol(s)
    savetxt("kol.txt", Kol, fmt="%14.6e")

pyMain()
```

Προσοχή: η συνάρτηση max() που έρχεται μαζί με την python (builtin) δεν μπορεί να χειριστεί πολυδιάστατα μητρώα και για αυτό πρέπει να χρησιμοποιηθεί η συνάρτηση max() που είναι ορισμένη στη βιβλιοθήκη numpy.

2. Υπολογισμός ολικού μητρώου δυσκαμψίας επιπέδου δικτυώματος

Το μητρώο δυσκαμψίας του ράβδου ij επιπέδου δικτυώματος που σχηματίζει γωνία θ με το οριζόντιο άξονα και συνδέει τον κόμβο i με τον κόμβο j δίνεται:

$$[K_{ij}]=\frac{A_{ij}E_{ij}}{L_{ij}}\begin{bmatrix} c^2 & & \text{συμ} & \\ cs & s^2 & & \\ -c^2 & -cs & c^2 & \\ -cs & -s^2 & cs & s^2 \end{bmatrix}$$

όπου L, A, E το μήκος, το εμβαδόν διατομής και το μέτρο ελαστικότητας αντίστοιχα και c=cos(θ), s=sin(θ). Αν υπάρχουν N κόμβοι το ολικό μητρώο δυσκαμψίας [Kol] έχει διαστάσεις είναι 2N×2N. Οι σειρές 0, 1 και οι στήλες 0, 1 του επί μέρους μητρώου αντιστοιχούν στον κόμβο i και οι γραμμές 2, 3 και οι στήλες 2, 3 στον κόμβο j. Στο ολικό μητρώο οι σειρές 2i και 2i+1 και οι στήλες 2i και 2i+1 αντιστοιχούν στον κόμβο i και οι γραμμές 2j, 2j+1 και οι στήλες 2j, 2j+1 στον κόμβο j.

Το επί μέρους μητρώο κάθε ράβδου χωρίζεται συμμετρικά σε 4 υπομητρώα διαστάσεων 2×2, πάνω αριστερά, πάνω δεξιά, κάτω αριστερά και κάτω δεξιά:

Το πάνω αριστερά	αντιστοιχεί στις γραμμές 2i και 2i+1 και στις στήλες 2i και 2i+1 του ολικού
Το πάνω δεξιά	αντιστοιχεί στις γραμμές 2i και 2i+1 και στις στήλες 2j και 2j+1 του ολικού
Το κάτω αριστερά	αντιστοιχεί στις γραμμές 2j και 2j+1 και στις στήλες 2i και 2i+1 του ολικού
Το κάτω δεξιά	αντιστοιχεί στις γραμμές 2j και 2j+1 και στις στήλες 2j και 2j+1 του ολικού

Έτσι το ολικό μητρώο δυσκαμψίας [Kol] αρχικά δημιουργείται με μηδενικά στοιχεία. Στη συνέχεια το επί μέρους μητρώο κάθε ράβδου χωρίζεται συμμετρικά σε 4 υπομητρώα διαστάσεων 2×2, και το κάθε

επιμέρους μητρώο προστίθεται στο αντίστοιχο υπομητρώο του ολικού μητρώου.
Να συνταχθεί πρόγραμμα σε το οποίο:
1. Να διαβάζει από το αρχείο k.txt τα στοιχεία L, A, E, θ (μοίρες), i, j των ράβδων (μία ράβδος ανά σειρά αρχείου).
2. Να υπολογίζει το πλήθος N των κόμβων.
3. Να δημιουργεί και να τυπώνει με εκθετική μορφή και στοιχισμένα στο αρχείο k.txt το ολικό μητρώο δυσκαμψίας.
Δοκιμάστε το πρόγραμμά σας με το αρχείο k.txt:

```
3 0.09 200e9 0 0 1
4 0.16 200e9 90 0 2
5 0.12 200e9 143.13 1 2
```

Λύση

Το επί μέρους μητρώο δυσκαμψίας έχει αρκετούς υπολογισμούς και έτσι συνταχθεί συνάρτηση που το υπολογίζει. Αρχικά θα υπολογιστεί το κάτω τριγωνικό μέρος του και στην συνέχεια αντιγραφεί συμμετρικά στο άνω τριγωνικό.

```
# -*- coding: iso-8859-7 -*-
from numpy import loadtxt, savetxt, zeros, ones, max, deg2rad, cos, sin


def barMat(L, A, E, theta):
    "Computes the axial stiffness matrix of a bar in 2D."
    theta = deg2rad(theta)
    c = cos(theta)
    s = sin(theta)
    c2 = c*c
    s2 = s*s
    cs = c*s
    k = E*A/L
    K = zeros((4, 4))

    K[0, 0] = k * c2

    K[1, 0] = k * cs
    K[1, 1] = k * s2

    K[2, 0] = -K[0, 0]
    K[2, 1] = -K[1, 0]
    K[2, 2] =  K[0, 0]

    K[3, 0] = -K[1, 0]
    K[3, 1] = -K[1, 1]
    K[3, 2] =  K[1, 0]
    K[3, 3] =  K[1, 1]

    for i in range(3):
        K[i, i+1:] = K[i+1:, i]
    return K


def ypolKol(s):
    "Υπολόγισε ολικό μητρώο δυσκαμψίας."
    N = max(s[:, 4:6]) + 1
    Kol = zeros((2*N, 2*N))
    for L, A, E, theta, i, j in s:
        K = barMat(L, A, E, theta)
        i *= 2
        j *= 2
        Kol[i:i+2, i:i+2] += K[0:2, 0:2]
        Kol[i:i+2, j:j+2] += K[0:2, 2:4]
        Kol[j:j+2, i:i+2] += K[2:4, 0:2]
```

```
        Kol[j:j+2, j:j+2] += K[2:4, 2:4]
    return Kol


def pyMain():
    "Start here."
    s = loadtxt("k.txt")
    Kol = ypolKol(s)
    savetxt("kol.txt", Kol, fmt="%14.6e")


pyMain()
```

Από αυτό το παράδειγμα μπορεί κάποιος να εκτιμήσει τον τελεστή +=. Αν δεν υπήρχε ο κώδικας για το ολικό μητρώο θα γινόταν:

```
        Kol[i:i+2, i:i+2] = Kol[i:i+2, i:i+2] + K[0:2, 0:2]
        Kol[i:i+2, j:j+2] = Kol[i:i+2, j:j+2] + K[0:2, 2:4]
        Kol[j:j+2, i:i+2] = Kol[j:j+2, i:i+2] + K[2:4, 0:2]
        Kol[j:j+2, j:j+2] = Kol[j:j+2, j:j+2] + K[2:4, 2:4]
```

Αυτό ο κώδικας είναι πιο επίπονος στην πληκτρολόγηση, δημιουργεί πιθανότητες λάθους και δυσχεραίνει την αναγνωσιμότητα.