Πανεπιστήμιο Δυτικής Μακεδονίας

ΕΡΓΑΣΙΑ ΕΞΑΜΗΝΟΥ:

Επιστήμη Δεδομένων Εφαρμογές R-SQL

Μάθημα:

Επιστήμη Δεδομένων Εφαρμογές R-SQL

Γεωργουβιά Ιωάννα Α.Μ.: DN12143

Εργασία Εξαμήνου

(υποβλήθηκε στο Τμήμα Διοικητικής Επιστήμης και Τεχνολογίας, Πανεπιστήμιο Δυτικής Μακεδονίας)

Κοζάνη 2020

Πανεπιστήμιο Δυτικής Μακεδονίας

ΕΡΓΑΣΙΑ ΕΞΑΜΗΝΟΥ:

Επιστήμη Δεδομένων Εφαρμογές R-SQL

Μάθημα:

Επιστήμη Δεδομένων Εφαρμογές R-SQL

Επιβλέπων Καθηγητής:

Δρ. Ζησόπουλος Δημήτριος

Περιεχόμενα

ΠΕΡΙΛΗΨΗ ΒΙΝΤΕΟ

25/05/2020 – Επιχειρηματική Παγκόσμια Πραγματικότητα

Έγινε λόγος :

1. Για τη δημιουργία προγράμματος εξέτασης από την Google
2. Για τη δημιουργία βίντεο ή καναλιού στο Youtube
3. Βίντεο meeting και
4. Πολυεξεταστική φόρμα 2020

16/05/2020 – Matlab programming forms

Διδάχθηκε σε matlab πώς «τρέχει» ένα αρχείο από το web και πως υποβάλλουμε την εργασία στη φόρμα.

04/05/2020 – DAV εργασίες.

Διδάχθηκε:

1. Πως ανεβαίνει ένα αρχείο στο share.uowm.gr
2. Τρόπος εξέτασης εργασίας
3. Διάφοροι editors και browsers

27/04/2020 Ερωτήσεις τηλέτυπου

Έγινε λόγος για το github και την απαλλαγή του μαθήματος με 10 με ανέβασμα της εργασίας στο github.

24/04/2020 Βαθμολόγηση ανάρτησης.

Παρουσιάστηκε πως βγαίνει ο βαθμός του μαθήματος, πως γράφουμε σε κάποιο browser

27/04/2020 DataScience Life Circles

Διδάχθηκε :

1. Νοημοσύνη σμήνους (swarm intelligence)
2. Εφαρμογή σε επιστήμη δεδομένων – Data Science Life Circle for our research
    a. Research understanding
    b. Data Mining
    c. Data Cleaning
    d. Data exploration
    e. Feature engineering
    f. Predictive modeling
    g. Data visualization
    h. Business understanding
3. Data Science
    a. Math and Statistics
    b. Domains/Business Knowledge
    c. Computer Science / IT

13/04/2020 BI vs DS

13/04/2020 Επανεκκίνηση μέρος 1

Πως γράφεται μια εργασία

11/04/2020 Επανεκκίνηση μέρος 2

Πως γράφεται μια εργασία

06/04/2020 Data Science Python

Διδάχθηκαν προγράμματα όπως

1. Ορίστε και εφαρμόστε 3 ακεραίους
2. Ορίστε 4 αριθμούς και επιστρέψτε το άθροισμα
3. Ποια έκδοση python χρησιμοποιείτε

06/04/2020 Data Science Installation

Διδάχθηκε

1. τρόπος εγκατάστασης προγραμμάτων για το συνονθύλευμα της επιστήμης δεδομένων
2. Επιλογής software
3. Github zisopoulos

13/04/2020 Εργασία μία και καλή

Διδάχθηκε ο τρόπος συγγραφής της εργασίας.

27/04/2020 CBDC FINTECH STEFANOS reads Dimitrios

Έγινε ανάγνωση του αρχείου Συμπλήρωμα μαθηματικών

25/05/2020 Χωρίς ήχο Data Camp intro

Διδάχθηκε η δημιουργία free account στα :

1. Learn data sciences online
2. Learn R
3. Learn Python
4. Learn SQL

1 ΠΡΟΓΡΑΜΜΑΤΑ ΓΛΩΣΣΑΣ PYTHON

1.1 1 ΓΕΝΙΚΕΣ

1.1.1 ΕΚΔΟΣΗ Python

Πρόγραμμα που εμφανίζει την έκδοση Python που χρησιμοποιείτε

Πρόγραμμα

```
# Ποια έκδοση python χρησιμοποιειτε ;
import sys
print("Ποια έκδοση Python χρησιμοποιείτε;")
print (sys.version)
```

Αποτέλεσμα

```
Ποια έκδοση Python χρησιμοποιείτε;
3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)]
```

Σχόλιο

Είναι σημαντικό να γνωρίζουμε την έκδοση. Πολλές φορές υπάρχουν διαφορές ανάμεσα στις εκδόσεις. Ένα χαρακτηριστικό παράδειγμα είναι στην Python 2 δεν χρησιμοποιούμε παρενθέσεις για την εντολή print, ενώ στην Python 3 είναι απαραίτητες.

1.1.2 Λίστα με Ηλικίες

Πρόγραμμα που δημιουργεί (όχι από τον χρήστη) μια λίστα με ηλικίες και τις εμφανίζει με 4 διαφορετικούς τρόπους

1) Με χρήση εντολής For

2) Χωρίς χρήση βρόχου επανάληψης

3) Χωρίς βρόχο χωρισμένες με κόμμα

4) Χωρίς βρόχο επανάληψης με κάθε ηλικία σε νέα σειρά

Πρόγραμμα

```
#Εμφάνισε τις ηλικίες  με διαφορετικούς τρόπους
#1 με for
```

```
#2 χωρίς λούπα
#3 χωρίς λούπα χωρισμένες με κόμμα
#4 χωρίς λούπα με κάθε ηλικία σε νέα σειρά

age = [31, 42, 65, 24, 67]

# εμφάνισε τις ηλικίες με χρήση for
print("Οι ηλικίες με χρήση for :",end="\n")
for x in range(len(age)):

    print(age[x])
#2 χωρίς λούπα
# εκτύπωση χωρίς λούπα και αγκύλες
print("Οι ηλικίες χωρίς αγκύλες :",*age)

# εκτύπωση που να χωρίζει της ηλικίες με κόμμα
print("Οι ηλικίες χωρισμένες με κόμμα :")
print(*age, sep=", ")
#Κάθε ηλικία σε νέα σειρά
print("Κάθε ηλικία σε νέα σειρά :")
print(*age, sep="\n")
```

Αποτέλεσμα

```
Οι ηλικίες με χρήση for :
31
42
65
24
67
Οι ηλικίες χωρίς αγκύλες : 31 42 65 24 67
Οι ηλικίες χωρισμένες με κόμμα :
31, 42, 65, 24, 67
Κάθε ηλικία σε νέα σειρά :
31
42
65
24
67
```

Σχόλιο

*Το * χρησιμεύει έτσι ώστε να μην φαίνονται οι αγκύλες .Το sep() μέσα στο print* είναι για να διαχωρίζει τα στοιχεία μεταξύ τους.

1.1.3 Πρόγραμμα Κωδικών

Πρόγραμμα που δέχεται κωδικούς από τον χρήστη και εμφανίζει μόνο εκείνους που μπορούν να χρησιμοποιηθούν. (Τα κριτήρια είναι σε μορφή σχολίων στο πρόγραμμα)

Πρόγραμμα

```
#Τα κριτήρια για κάθε κωδικό είναι:

#1.Τουλαχιστον ένα  γράμμα [a-z]
#2.Τουλάχιστον ένα  γράμμα [A-Z]
#3.Τουλαχιστον έναν αριθμό [0-9]
#4.Τουλάχιστον ένα σύμβολο [@#$]
#5.Το μικρότερο επιτρεπτό μήκος είναι 6
#6.Το μεγαλύτερο επιτρεπτό μήκος είναι 12

#Δίνει ο χρήστης κάποιους κωδικούς (χωρίζονται με κόμμα)

passwords=input("Δώστε κωδικούς ").split(",")

#Για κάθε στοιχειό μέσα στο passwords
for i in passwords:

# αν ο κωδικός ικανοποιεί όλους τους κανόνες ,εκτυπώνεται
    if len(i)>=6 and len(i)<=12 and any(j.isupper() for j in i)==True and any(j.is
digit() for j in i)==True and any(j.islower() for j in i)==True and any(j in ("#",
"@","$") for j in i):
        print("Ο κωδικός {} μπορεί να χρησιμοποιηθει. ".format(i))
```

Αποτέλεσμα

```
Δώστε κωδικούς adfa21, ase22##ASD,ereg
Ο κωδικός  ase22##ASD μπορεί να χρησιμοποιηθεί.
```

Σχόλιο

Δίνει ο χρήστης κωδικούς. Για κάθε κωδικό ελέγχουμε αν ικανοποιεί όλα τα κριτήρια και αν ναι τον εμφανίζουμε με μήνυμα. Το any() επιστρέφει True αν έστω και ένα είναι True.

1.1.4 Έλεγχος Ακεραίου

Πρόγραμμα που ελέγχει αν ένας ακέραιος που δίνεται από το χρήστη είναι μέσα στο εύρος 5 έως 15

Πρόγραμμα

```
#ορίζουμε συνάρτηση
def test(n):
    if n in range(5,15):
        print( " %s είναι στο εύρος "%str(n))
    else :
        print("Ο αριθμός είναι εκτός εύρους.")

#καλούμε την συνάρτηση και ο χρήστης εισάγει έναν αριθμό
test(int(input("Δώστε έναν αριθμό ")))
```

Αποτέλεσμα

Τρέχουμε το πρόγραμμα δύο φορές

```
Δώστε έναν αριθμό 3
Ο αριθμός είναι εκτός εύρους.
Δώστε έναν αριθμό 8
 8 είναι στο εύρος
```

Σχόλιο

Ορίζουμε συνάρτηση .Το %s δείχνει ότι είναι string ,αν ο αριθμός που δίνει ο χρήστης όταν καλεί την συνάρτηση είναι από 4 μέχρι 18 τότε ανήκει στο εύρος ο αριθμός.

1.2 ΛΙΣΤΕΣ

1.2.1 Δημιουργία λίστας με ακεραίους

Δημιουργία λίστας με ακέραιους αριθμούς η οποίοι θα δίνονται από τον χρήστη και εκτύπωση της λίστας

Πρόγραμμα

```
#Αρχικοποιώ την λίστα μου
λιστα = []

# εισάγετε πόσους αριθμούς θα έχει η λίστα σας
n = int(input("Πόσους ακέραιους αριθμούς θέλετε να περιέχει η λίστα ;"))
# τρέχει η λούπα όσες φορές είναι το n
for i in range(0, n):
    στοιχειο = int(input("Δώστε ενα ακέραιο αριθμό : ")) #πατηστε enter μετά απο
κάθε επιλογή

    λιστα.append(στοιχειο) # κάθε στοιχείο μπαίνει στην λίστα που δημιουργήσατε
στην αρχή
#εκτύπωση της λίστας χωρίς αγκύλες
print("Η λίστα περιέχει τους αριθμούς: ",end=" ")
```

```
print(*λιστα,sep=",")
```

Αποτέλεσμα

```
Πόσους ακέραιους αριθμούς θέλετε να περιέχει η λίστα ;4
Δώστε ένα ακέραιο αριθμό : 23
Δώστε ένα ακέραιο αριθμό : 56
Δώστε ένα ακέραιο αριθμό : 76
Δώστε ένα ακέραιο αριθμό : 88
Η λίστα περιέχει τους αριθμούς:  23,56,76,88
```

Σχόλιο

Κάθε στοιχείο προστίθεται στην τελευταία θέση της λίστας «λιστα». Πρέπει πρώτα να την ορίσουμε σαν μια άδεια λίστα. Το end=" " χρησιμοποιείται για να αφήσει ένα κενό ανάμεσα στο πρώτο print και το δεύτερο. Θα μπορούσε να είναι και κάτι άλλο εκτός από κενό πχ νέα σειρά (\n) .

1.2.2 Δημιουργία λίστας με ακέραιους από το χρήστη και εκτύπωση της

Πρόγραμμα

```
#Δημιουργηστε μια λιστα με ακεραιους αριθμους .Η λιστα θα δεχεται αριθμους απο τον
 χρηστη και στο τελος θα την εμφανιζει με αποσταση
#δυο κενών μεταξύ των αριθμών
print("Η λίστα περιέχει τους αριθμούς ", *list(map(int,input("Δώστε ακέραιους αριθ
μούς ").split(","))),sep=" ")
```

Αποτέλεσμα

```
Δώστε ακέραιους αριθμούς 34,56,44,3,6
Η λίστα περιέχει τους αριθμούς  34 56 44 3 6
```

Σχόλιο

Η map( ) συνάρτηση εφαρμόζει μια δοσμένη συνάρτηση (εδώ την int) σε κάθε στοιχείο μιας λίστας ,tuple,…και με την εντολή list γίνεται λίστα.

1.2.3 Πρόγραμμα που βρίσκει το μέγιστο αριθμό σε μια Λίστα

Πρόγραμμα

```
#Γραψτε ενα προγραμμα που βρισκει το μεγιστο αριθμο απο την λιστα
[443,435,56,324,65,123,67,54,-546,0]

#Η λίστα μας
lista=[443,435,56,324,65,123,67,54,-546,0]

#sorted : ταξινόμηση της λίστας από το μικρότερο στο μεγαλύτερο
sort_list=sorted(lista)

#Για να βρουμε τον μεγαλυτερο αριφμο αρκει να παρουμε το τελευταιο στοιχειο της τα
ξινομημενης λιστας
print("Ο μέγιστος της λίστας {} ειναι το {}".format(lista,sort_list[-1]))
```

Αποτέλεσμα

```
Ο μέγιστος της λίστας [443, 435, 56, 324, 65, 123, 67, 54, -546, 0] είναι το 443
```

Σχόλιο

Η εντολή sorted(one_list) ταξινομεί την λίστα σε αύξουσα σειρά. Άρα θα εμφανίσουμε το τελευταίο στοιχείο της ταξινομημένης λίστας. Σε περίπτωση που θέλαμε σε φθίνουσα θα χρησιμοποιούσαμε sorted(one_list,reverse=True)

1.2.4 Πρόγραμμα που δείχνει τη διαφορά ανάμεσα στην εντολή insert και append για μια λίστα

Πρόγραμμα

```
#Δημιουργία μιας λίστας
lista=[34,34,24,435,2,4,3,4]

# insert
#βάζουμε την λέξη "aris" στην θέση 3 στην λίστα
lista1=[34,34,24,435,2,4,3,4]
lista.insert(3,"aris")
print(lista)

#append
##βάζουμε την λέξη "aris" στην θέση 3 στην λίστα
lista2=[34,34,24,435,2,4,3,4]
lista2.append("aris")
print(lista2)
```

Αποτέλεσμα

```
[34, 34, 24, 'aris', 435, 2, 4, 3, 4]
[34, 34, 24, 435, 2, 4, 3, 4, 'aris']
```

Σχόλιο

Με το append προσθέτουμε ένα στοιχείο στο τέλος μιας λίστας, ενώ με το insert προσθέτουμε ένα στοιχείο σε θέση που ορίζουμε εμείς.

1.3 DICTIONARIES

1.3.1 Πρόγραμμα δημιουργίας Dictionary

Πρόγραμμα

```
#Δημιουργούμε ενα dictionery χωρίς στοιχειά (αδειο)
dict={}
print("Εμφανίζουμε ενα άδειο dictionery ",end="\n")# Το \n ειναι για αλλαγή σειράς
print(dict)

#Γεμίζουμε το dictionery
dict["aris"]=3
dict["paok"]=4
dict["iraklis"]=13
print("Εμφανίζουμε το dictionary  με τα στοιχεία που προσθέσαμε ",dict)

#Αλλάζουμε την τιμή στο dictionary (Ηρακλής απο 13 σε 10)
dict["iraklis"]=10
print("Μετά την αλλαγή το dictonary έγινε : ",dict)

#Διαγράφουμε το ΠΑΟΚ από το dictionery
del dict["paok"]
print("Μετά την διαγραφή ",dict)

#Στο dictionary προσθέτουμε nested δεδομένα
dict["athens"]={"aek":33,"pao":13,"atromitos":31}
dict["eparxia"]={"ofara":4,"ergotelis":34,"kalamaria_pontos":12}
print("Προσθεσαμε στο dictionary nested δεδομένα ",dict)

#Διαγράφουμε απο nested dictionary
del dict["eparxia"]["kalamaria_pontos"]
print(dict)

#Διαγράφουμε ολόκληρο το dictionary
dict.clear()
print("Διαγράφουμε το dictionary",dict)
```

Αποτέλεσμα

```
Εμφανίζουμε ένα άδειο dictionery
{}
```

```
Εμφανίζουμε το dictionary με τα στοιχεία που προσθέσαμε {'aris': 3, 'paok': 4,
'iraklis': 13}
Μετά την αλλαγή το dictonary έγινε :  {'aris': 3, 'paok': 4, 'iraklis': 10}
Μετά την διαγραφή  {'aris': 3, 'iraklis': 10}
Προσθεσαμε στο dictionary nested δεδομένα  {'aris': 3, 'iraklis': 10, 'athens':
{'aek': 33, 'pao': 13, 'atromitos': 31}, 'eparxia': {'ofara': 4, 'ergotelis': 34,
'kalamaria_pontos': 12}}
{'aris': 3, 'iraklis': 10, 'athens': {'aek': 33, 'pao': 13, 'atromitos': 31},
'eparxia': {'ofara': 4, 'ergotelis': 34}}
Διαγράφουμε το dictionary {}
```

Σχόλιο

Αρχικά δημιουργούμε ένα άδειο dictionary .Έπειτα ορίζουμε τιμές σε keys και values. Μετά διαγράφουμε ένα item(ένα ζεύγος από το dictionary πχ paok) και δημιουργούμε nested dictionary.Τέλος διαγράφουμε το dictionary με την εντολή clear.

1.3.2 Πρόγραμμα που επιστρέφει ονόματα από δεδομένο dictionary

Πρόγραμμα

```
#Dictionary
dict={"Papadopoulos":1000,"Georgouvia":1300,"Karipidis":3100,"Kasapis ":900,"Mitra
kas":2200}

#Επιλεγουμε τα ονοματα απο το dictionary και τα αποθηκευουμε στο "names"
names=dict.keys()

#Αν τα επιστέψουμε χωρίς * θα έχουμε
print(names)

#Τα επιστρέφουμε με χρήση * και sep(τα διαχωρίζει με κόμμα)
print(*names,sep=",")
```

Αποτέλεσμα

```
dict_keys(['Papadopoulos', 'Georgouvia', 'Karipidis', 'Kasapis ', 'Mitrakas'])
Papadopoulos,Georgouvia,Karipidis,Kasapis ,Mitrakas
```

Σχόλιο

*Τα ονόματα βρίσκονται στα keys,τα αποθηκεύουμε στην μεταβλητή names.Αν προσπαθήσουμε να εκτυπώσουμε τα ονόματα παρατηρούμε ότι εμφανίζει την λέξη dict_keys που απλά σημάνει ότι είναι το αποτέλεσμα από ένα dictionary και συγκεκριμένα από τα keys του .Με το * μέσα στο print λύνεται το πρόβλημα.*

1.3.3 Πρόγραμμα που ενώνει 3 dictionaries σε ένα

Πρόγραμμα

```
#Δίνονται
dic1={5:15, 6:16}
dic2={7:17, 8:18}
dic3={9:19,10:20}
#Τελικό dictionary ,που περιέχει τα άλλα 3
dict = {}

#Το ι τρέχει για κέθε dictionary  και κάνει update τις τιμές αυτών στο τελικό
for i in (dic1,dic2,dic3): dict.update(i)
print("Ενώνοντας τα 3 dictionaries θα έχουμε : ",dict)
```

Αποτέλεσμα

```
Ενώνοντας τα 3 dictionaries θα έχουμε :  {5: 15, 6: 16, 7: 17, 8: 18, 9: 19, 10:
20}
```

Σχόλιο

Το update() προσθέτει στοιχεία στο dictionary αν τα keys δεν υπάρχουν ήδη στο dictionary. Αν τα keys υπάρχουν ανανεώνει τα key με νέα τιμή

1.3.4 Dictionary που θα περιέχει τα στοιχεία (x,x*x) με το χρήστη να επιλέγει τα ποσά

Πρόγραμμα

```
#Πόσα ζεύγη θέλει ο χρήστης
n=int(input("Δώστε ένα αριθμό "))

#Άδειο dictionary
dict={}

#Για κάθε  i εισάγει στο dictionary  σαν key το i και σαν value το i^2
for i in range(1,n+1): dict[i]=i**2

#Το αποτέλεσμα
print(dict)
```

Αποτέλεσμα

```
Δώστε ένα αριθμό 3
{1: 1, 2: 4, 3: 9}
```

Σχόλιο

Στην πρώτη γραμμή με την εντολή int διασφαλίζουμε ότι το πρόγραμμα δέχεται μόνο ακέραιες τιμές από τον χρήστη. Άλλο ένα πράγμα που πρέπει να προσέξουμε είναι το range.Για παράδειγμα άμα θέλουμε τιμές από 0 μέχρι 10 πρέπει να ορίσουμε range(0,11), για αυτό το range στο πρόγραμμα είναι από 0 μέχρι n+1.

1.4 ΓΡΑΜΜΑΤΑ – ΨΗΦΙΑ

1.4.1 Πρόγραμμα που δέχεται λέξεις ή προτάσεις και τις επιστρέφει με κεφαλαία γράμματα

Πρόγραμμα

```
#Ο χρήστης δίνει λέξεις ή προτάσεις
word=input("Δώστε κάποιες λέξεις ή προτάσεις ").split(",")

#Στο result θα έχουμε το τελικό αποτέλεσμα
result=[]

#Για κάθε γράμμα που δέχεται το μετατρέπει σε κεφαλαίο χρειάζεται)
for i in range(len(word)):

#το append προσθετει οτι υπαρχει μεσα στην παρενθεση του στο result
    result.append(word[i].upper()) #το upper μετατρεπει σε κεφαλαια

#Εμφανίζουμε το αποτέλεσμα
print(" ".join(result))
```

Αποτέλεσμα

```
Δώστε κάποιες λέξεις ή προτάσεις Hello how are you?
HELLO HOW ARE YOU?
```

Σχόλιο

Η εντολή split(“,”) σημαίνει ότι ο χρήστης δίνει δεδομένα και τα χωρίζει μεταξύ τους με κόμμα. Άμα αντί για κόμμα είχαμε βάλει παύλα θα χωρίζονταν με παύλα. Η εντολή sth.upper() μετατρέπει το sth σε κεφάλαια γράμματα .Σε περίπτωση που είναι ήδη κεφάλαιο ή ακόμα και αριθμός το αφήνει όπως είναι. Το λίστα.append() προσθέτει ένα στοιχείο στο τέλος της λίστας. Τέλος το join() είναι μια μέθοδος που χρησιμοποιείται με strings στην οποία κάθε στοιχείο της λίστας ενώνεται με κάτι (εδώ στην άσκηση με κενό).

1.4.2 Πρόγραμμα που ο χρήστης δίνει μια πρόταση και επιστρέφει τον αριθμό των γραμμάτων και των ψηφίων που περιέχονται σε αυτήν.

Πρόγραμμα

```
#Ο χρήστης γραφεί κατι
n=input("Γράψε κάτι ")

#Αρχικοποίηση λιστών
#Για τα ψηφιά
digits=[]

#Για τα γράμματα
letters=[]

#για κάθε στοιχειό του n
for i in n:
#Αν είναι ψηφίο τότε προστίθεται στη  λίστα digit
    if i.isdigit()==True: digits.append(i)

#Αν είναι γράμμα τότε προστίθεται στην λίστα letters
    if i.isalpha()==True: letters.append(i)

print("Η πρόταση σου περιέχει {} νούμερα ! ".format(len(digits)))
print("Η πρόταση σου περιέχει {} γράμματα ! ".format(len(letters)))
```

```
Γράψε κάτι kalimera what as3456 wwwo
Η πρόταση σου περιέχει 4 νούμερα !
Η πρόταση σου περιέχει 18 γράμματα !
```

Σχόλιο

Η εντολή isdigit() είναι True σε περίπτωση το i είναι ψηφίο, αλλιώς False.Η εντολή isalpha() είναι True σε περίπτωση το i είναι γράμμα.Το format είναι για να εμφανίζεται ότι είναι μέσα του σε συγκεκριμένη θέση, η θέση ορίζετε με {}

1.4.3 Πρόγραμμα που δέχεται λέξεις και τις εμφανίζει σε αλφαβητική σειρά

Πρόγραμμα

```
#Δίνει ο χρήστης τις λέξεις.(χαρακτήρες)
words=input("Δώστε κάποιες λέξεις ").split(",")
#αν η λίστα περιέχει λιγότερα από 4 εμφανίζει μήνυμα και AssertionError
assert len(words)>3 ,"Το λιγότερο 4 στοιχεία "
#Αλφαβητική σειρά
result1=sorted(words)

#Εξάλειψη των [] και " "
result=" ".join(result1)

#Εμφανίζουμε το αποτέλεσμα
print(result)
```

Αποτέλεσμα

Τρέχουμε το πρόγραμμα δυο φορές

```
Δώστε κάποιες λέξεις Megalos, Zisis, Kostas
Traceback (most recent call last):
  File "d:/Έγγραφα/repos/zis/4.3.grammata.py", line 6, in <module>
    assert len(words)>3 ,"Το λιγότερο 4 στοιχεία "
AssertionError: Το λιγότερο 4 στοιχεία
```

```
Δώστε κάποιες λέξεις Megalos, Zisi, Kostas, Panos
 Kostas  Megalos Panos  Zisi
```

Σχόλιο

Την πρώτη φορά που τρέξαμε το πρόγραμμα ο χρήστης έδωσε λίγα στοιχεία και εμφάνισε μήνυμα και AssertationError.Την δεύτερη φορά ο χρήστης έδωσε 4 λέξεις και επέστρεψε αυτές σε αλφαβητική σειρά.

1.5 ΗΜΕΡΟΜΗΝΙΕΣ

1.5.1 Πρόγραμμα που επιστρέφει τη διαφορά σε μέρες ανάμεσα σε δυο ημερομηνίες

Πρόγραμμα

```
# εισάγουμε το date
from datetime import date

#πρώτη ημερομηνία
date1 = date(2012, 12, 31)

#δεύτερη ημερομηνία
date2 = date(2010, 1, 1)

#η διαφορά τους
diff = abs(date2 -
 date1) # πήραμε την απόλυτη τιμή ,σε αντίθετη περίπτωση μπορεί να έχουμε
αρνητικούς
print(diff.days,"μέρες")
```

Αποτέλεσμα

```
1095 μέρες
```

Σχόλιο

Το abs() επιστρέφει την απόλυτη τιμή ,είναι χρήσιμο σε περίπτωση που αφαιρέσουμε από παλαιότερη ημερομηνία τη νεότερη.Με το date() ορίζουμε ημερομηνία.

1.5.2 Πρόγραμμα που τυπώνει την τωρινή ημερομηνία σε διάφορες μορφές

Πρόγραμμα

```
#ημερομηνία αυτήν τη στιγμή
from datetime import date
ημερομηνια = date.today()
print("Ημερομηνία σήμερα : ",ημερομηνια)
#τύπος
print("Ο τύπος της ημερομηνίας ειναι : ",type(ημερομηνια))

d = ημερομηνια.strftime("%Y-%m-%d")
print("Η σημερινή ημερομηνία είναι : ", d)
#τυπος
print("Ο τύπος της ημερομηνίας ειναι : ",type(d))

# μερα/μηνασ/ετος
d1 = ημερομηνια.strftime("%d/%m/%Y")
print("Η σημερινή ημερομηνία είναι : ", d1)

# μηνας(με χαρακτηρες) μερα ,ετος
d2 = ημερομηνια.strftime("%B %d, %Y")
print("Η σημερινή ημερομηνία είναι : ", d2)
# μηνας/μερα/ετος
d3 = ημερομηνια.strftime("%m/%d/%y")
print("Η σημερινή ημερομηνία είναι : ", d3)
# μηνας(με χαρακτηρες σε συντομογραφια)-μερα-ετος
d4 = ημερομηνια.strftime("%b-%d-%Y")
print("Η σημερινή ημερομηνία είναι : ", d4)
```

Αποτέλεσμα

```
Ημερομηνία σήμερα :  2020-05-04
Ο τύπος της ημερομηνίας είναι :  <class 'datetime.date'>
Η σημερινή ημερομηνία είναι :  2020-05-04
Ο τύπος της ημερομηνίας είναι :  <class 'str'>
Η σημερινή ημερομηνία είναι :  04/05/2020
Η σημερινή ημερομηνία είναι :  May 04, 2020
Η σημερινή ημερομηνία είναι :  05/04/20
Η σημερινή ημερομηνία είναι :  May-04-2020
```

1.5.3 Πρόγραμμα που θα δέχεται από τον χρήστη τον μήνα και το έτος και θα εμφανίζει ημερολόγιο

Πρόγραμμα

```
#εισάγουμε την βιβλιοθήκη calander
import calendar

#Διαλέξτε το ατός
year= int(input("Διαλέξτε το έτος : "))

#Διαλέξτε τον μήνα
monthh = int(input("Διαλέξτε τον μήνα : "))

#Εμφανίστε το ημερολόγιο
print(calendar.month(year, monthh))
```

Αποτέλεσμα

```
Διαλέξτε το έτος : 2020
Διαλέξτε τον μήνα : 5
      May 2020
Mo Tu We Th Fr Sa Su
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31
```

Σχόλιο

Εισάγουμε το calendar και χρησιμοποιούμε την εντολή calendar.month(2020,5)

1.6 ΣΥΝΑΡΤΗΣΕΙΣ

1.6.1 Πρόγραμμα που υπολογίζει το άθροισμα τριών αριθμών και αν είναι ίσοι μεταξύ τους επιστρέφει το τετράγωνο του αθροίσματος τους

Πρόγραμμα

```
#Η συνάρτηση δέχεται 3 αριθμούς z,y,z
def solver(x, y, z):
    sum = x + y + z

    # Αν είναι ίσοι  επιστρέφει το άθροισμα στο τετράγωνο
    if x == y == z:
        sum = sum ** 2
        return sum

# Καλούμε τη συνάρτηση για των 3 διαφορετικούς αριθμούς εμφανίζει None
print(solver(12, 34, 23))
```

```
#εμφανιζει το αθροισμα στο τετραγωνο (επειδη ειναι οι αριθμοι ισοι μεταξυ τους)
print(solver(4, 4, 4))
```

Αποτέλεσμα

```
None
144
```

Σχόλιο

Solver είναι το όνομα της συνάρτησης και δέχεται 3 μεταβλητές. Αρχικά υπολογίζουμε το άθροισμα τους ακόμα και να είναι ίσοι μεταξύ τους. Παρακάτω στο if ελέγχουμε αν είναι ίσοι μεταξύ τους και αν ναι τότε επιστρέφουμε το άθροισμα στο τετράγωνο .

1.6.2 Πρόγραμμα που θα δέχεται από τον χρήστη δεκαδικούς αριθμούς και θα επιστρέφει την διαφορά του μέγιστου από τον ελάχιστο

Πρόγραμμα

```
# Ζητάμε αριθμούς από τον χρήστη.Να χωρίζονται μεταξύ τους με κόμμα
lista=map(float,input("Δώστε κάποιους αριθμούς ").split(","))

#Μετατροπή σε λίστα
lista=list(lista)

#Ορίζουμε συνάρτηση που βρίσκει την διαφορά του μεγαλύτερου αριθμού με τον μικρότε
ρο
def diafora(lista):
# ταξινόμηση της λίστας
    lista=sorted(lista)
#μέγιστο είναι το τελευταίο στοιχείο της ταξινομημένης λίστας
    max=lista[-1]
#Ελάχιστο είναι το πρώτο στοιχείο της ταξινομημένης λίστας
    min=lista[0]
# Η διαφορά τους
    result=max-min
#Επιστράφει η συνάρτηση την διαφορά
    return "Η διαφορά του μέγιστου και του εάχιστου είναι: {}".format(result)

#Τρέχουμε την συνάρτηση
print(diafora(lista))
```

Αποτέλεσμα

```
Δώστε κάποιους αριθμούς 23,34,45
Η διαφορά του μέγιστου και του εάχιστου είναι: 22.0
```

Σχόλιο

Αρχικά ζητάμε από τον χρήστη float αριθμούς ,όμως με την εντολή split() επιστρέφονται μόνο σαν χαρακτήρες και για να γίνουν float(δεκαδικοί) χρησιμοποιούμε την map().Η map( ) συνάρτηση εφαρμόζει μια δοσμένη συνάρτηση (εδώ την float) σε κάθε στοιχείο μιας λίστας ,tuple,...(εδώ τα δεδομένα που εισάγει ο χρήστης). Δημιουργούμε συνάρτηση μέσα σε αυτήν υπάρχει η εντολή sorted() η οποία ταξινομεί την λίστα .Με lista[0] έχουμε το πρώτο στοιχείο την ταξινομημένης λίστας δηλαδή το max, αντίστοιχα το τελευταίο είναι το min.Τέλος καλούμε την συνάρτηση.

1.6.3 Πρόγραμμα που επιστρέφει το άθροισμα όλων των στοιχείων μιας λίστας δοσμένης από το χρήστη με χρήση συνάρτησης

Πρόγραμμα

```
#Δημιουγία λίστας από τον χρήστη.Με την χρήση του split (το οποίο χρησιμοποιείται
για να χωρίζει
# τα στοιχεία με κάτι
εδώ με κόμμα , τα στοιχειά της λίστας είναι σε μορφή χαρακτήρα
data=(input("Δώστε αριθμούς ").split(","))
#print(data)

#από χαρακτήρες σε δεκαδικούς (float)
data=[float(i) for i in data]
#print(data)

#Ορίζουμε συνάρτηση με όνομα summ
def summ(data):
    result=0
    for i in data:
#κάθε ένα στοιχειό της λίστας προστίθεται στο result
        result=result+i
# η συνάρτηση επιστρέφει το result
    return result

#καλούμε την συνάρτηση και εμφανίζουμε το αποτέλεσμα
print("Το άθροισμα όλων των στοιχείων της λίστας είναι ",summ(data))
```

Αποτέλεσμα

```
Δώστε αριθμούς 12,34,24,56
Το άθροισμα όλων των στοιχείων της λίστας είναι  126.0
```

Σχόλιο

Δημιουργούμε μια λίστα με float αριθμούς και συνάρτηση που για κάθε στοιχείο προστίθεται στο άθροισμα των προηγουμένων. Αρχικοποιούμε το result με μηδέν διότι θέλουμε το άθροισμα

1.6.4 Πρόγραμμα που επιστρέφει τη μέση τιμή όλων των στοιχείων μιας λίστας δοσμένης από το χρήστη με χρήση συνάρτησης

Πρόγραμμα

```
#εισαγουμε την βιβλιοθηκη( για να τρεχει η στρογγυλοποιηση floor)
import math

#Δημιουργία λίστας με αριθμούς από τον χρήστη
x=[int(i) for i in input("Δώστε αριθμούς ").split(",")]

#Ταξινόμηση της λίστας από το μικρότερο στο μεγαλύτερο
data=sorted(x)

#Ορίζουμε συνάρτηση που δέχεται την ταξινομημένη λίστα και επιστρέφει
την μέση τιμή
def finder_median(data):
# Αν έχουμε περιττό μήκος πινάκα
    if len(data)%2!=0:
#Βρίσκουμε το μεσαίο αριθμό του πίνακα ο οποίος είναι και η μέση τιμή
        l=data[int(math.floor(len(data)/2))]
#Επιστρέφουμε το αποτέλεσμα
        print("Η μέση τιμή του πίνακα {} είναι {}".format(x,l))

# Αν έχουμε άρτιο μήκος πίνακα
    else:
# Βρίσκουμε την θέση του μεσαίου αριθμού του πίνακα
        l = int(math.floor(len(data) / 2))
#ορίζουμε ως a,b τα δυο μεσαία στοιχεία και
επιστρέφουμε τον μέσο όρο τους
        a = data[l]
        b = data[l-1]
        result=(a+b)/2
        print("Η μέση τιμή του πίνακα {} είναι {}".format(x,result))

#Τρέχουμε την συνάρτηση
finder_median(data)
```

Αποτέλεσμα

```
Δώστε αριθμούς 10,20,30,40
Η μέση τιμή του πίνακα [10, 20, 30, 40] είναι 25.0
```

Σχόλιο

Εισάγουμε δεδομένα και τα ταξινομούμε. Η ταξινόμηση είναι απαραίτητη (από στατιστική).Άμα δώσει ο χρήστης περιττό αριθμό στοιχείων ο μεσαίος είναι το

αποτέλεσμα αν όχι περνούμε την μέση τιμή των δυο μεσαίων σαν αποτέλεσμα. Το floor χρησιμοποιείται για στρογγυλοποίηση προς τα κάτω

1.7 ΓΕΩΜΕΤΡΙΚΕΣ

1.7.1 Πρόγραμμα που δέχεται τις διαστάσεις ενός τριγώνου και επιστρέφει αν είναι ορθογώνιο ή όχι.

Πρόγραμμα

```
#Διαστάσεις από τον χρήστη
data= list(map(int,input("Δώστε τις διαστάσεις του τριγώνου ").split(",
")))

#ταξινόμηση των διαστάσεων από το μικρότερο) στο μεγαλύτερο)
x,y,z = sorted(data)

#από πυθαγόρειο έχουμε την συνθήκη
if x**2+y**2==z**2:
    print('Είναι ορθογώνιο τρίγωνο')
else:
    print("Δεν είναι ορθογώνιο τρίγωνο")
```

Τρέχουμε το πρόγραμμα δυο φορές για επαλήθευση για να επαληθεύσουμε τη λειτουργία του. Την πρώτη φορά με διαστάσεις 5,6,7 που δεν αντιστοιχούν σε ορθογώνιο τρίγωνο

```
Δώστε τις διαστάσεις του τριγώνου 5,6,7
Δεν είναι ορθογώνιο τρίγωνο
```

Τη δεύτερη φορά δίνω διαστάσεις ορθογωνίου τριγώνου με πλευρές 3,4,5

```
Δώστε τις διαστάσεις του τριγώνου 3,4,5
Είναι ορθογώνιο τρίγωνο
```

Σχόλιο

Έγινε ταξινόμηση από το μικρότερο στο μεγαλύτερο έτσι ώστε να διασφαλίσουμε ότι η υποτείνουσα είναι η μεγαλύτερη από όλες τις άλλες πλευρές .Για παράδειγμα αν

είναι οι πλευρές από 1 cm η κάθε μια η υποτείνουσα δεν γίνεται να είναι 10 γιατί δεν θα ήταν καν τρίγωνο. Τέλος αν ισχύει το πυθαγόρειο θεώρημα είναι ορθογώνιο τρίγωνο αλλιώς όχι.

1.7.2 ΠΡΟΓΡΑΜΜΑ ΠΟΥ ΔΕΧΕΤΑΙ ΤΗΝ ΑΚΤΙΝΑ ΕΝΟΣ ΚΥΚΛΟΥ ΚΑΙ ΕΠΙΣΤΡΕΦΕΙ ΤΟ ΕΜΒΑΔΟ ΤΟΥ

Πρόγραμμα

```
#εισάγουμε το π από την math βιβλιοθήκη
from math import pi

#Ζητάμε από τον χρήστη την ακτίνα του κύκλου
radius=float(input("Δωστε την ακτίνα : "))#δέχεται δεκαδικό αριθμό

#υπολογίζουμε το εμβαδόν
area=pi*radius**2

#Εμφανίζουμε το αποτέλεσμα
print("Το εμβαδόν του κύκλου με ακτίνα {} είναι {} ".format(radius,area
))
```

Αποτέλεσμα

```
Δώστε την ακτίνα : 10
Το εμβαδόν του κύκλου με ακτίνα 10.0 είναι 314.1592653589793
```

Σχόλιο

Ο χρήστης ορίζει την ακτίνα, και από τον γνωστό γεωμετρικό τύπο $\pi*R^2$ έχουμε το αποτέλεσμα .Το pi το εισάγουμε από το math (βιβλιοθήκη) .

1.7.3 Πρόγραμμα που επιστρέφει το εμβαδό τριγώνου

Πρόγραμμα

```
#βάση
b = float(input("Δώστε την βάση του τριγώνου: "))

#ύψος
h = float(input("Δώστε το ύψος του τριγώνου: "))
```

```
#το εμβαδόν
area = b*h/2

print("Το εμβαδόν του τριγώνου ειναι: ", area)
```

Αποτέλεσμα

```
Δώστε την βάση του τριγώνου: 4
Δώστε το ύψος του τριγώνου: 3
Το εμβαδόν του τριγώνου ειναι:  6.0
```

Σχόλιο

Έγινε ταξινόμηση από το μικρότερο στο μεγαλύτερο έτσι ώστε να διασφαλίσουμε ότι η υποτείνουσα είναι η μεγαλύτερη από όλες τις άλλες πλευρές .Για παράδειγμα αν είναι οι πλευρές από 1 cm η κάθε μια η υποτείνουσα δεν γίνεται να είναι 10 γιατί δεν θα ήταν καν τρίγωνο .Τέλος αν ισχύει το πυθαγόρειο θεώρημα είναι ορθογώνιο τρίγωνο αλλιώς όχι.

1.8 Διαχείριση κλάσεων (CLASS)

1.8.1 Δημιουργία CLASS

Δημιουργία μιας class η οποία θα καλείται και θα εμφανίζει το αποτέλεσμα της πράξης x^n (pow(x,n)) ,πχ x=2,n=3 το αποτέλεσμα που επιστρέφει είναι $2^3=8$ σε μορφή πράξης.

Πρόγραμμα

```
#Ορίζουμε μια κλάση με όνομα power
class power():
#
    def __init__ (self,x,n):
        self.x=x
        self.n=n
#δημιουργούμε συνάρτηση που επιστρέφει το pow(x,n)
    def result(self): return self.x**self.n

#βάζουμε τις τιμές που θέλουμε και εμφανίσουμε το αποτέλεσμα
first=power(-3,5)
print("{} ** {} = {}".format(first.x,first.n,first.result()))

#βάζουμε τις τιμές που θέλουμε και εμφανίσουμε το αποτέλεσμα
second=power(2,11)
print("{} ** {} = {}".format(second.x,second.n,second.result()))
```

Αποτέλεσμα

```
-3 ** 5 = -243
2 ** 11 = 2048
```

Σχόλιο

Αρχικά δημιουργούμε μια κλάση. Με το init (self) η κύρια χρησιμότητα του είναι για να αρχικοποιήσει δεδομένα στα μέλη μιας κλάσης (σαν μέλη εδώ μπορούμε να πούμε ότι είναι το first ,second). Έπειτα ορίζουμε την συνάρτηση result η οποία επιστρέφει το αποτέλεσμα. Τέλος καλούμε τις κλάσεις και τυπώνουμε τα αποτελέσματα.

1.8.2 Δημιουργία κλάσης employee με όνομα , επίθετο , μισθό και email

Πρόγραμμα

```
class Employee():
#Για κάθε εργαζόμενο χρειαζόμαστε όνομα, επίθετο, μισθό και email
    def __init__    (self,name,last,pay):
        self.name=name
        self.last=last
        self.pay=pay
    # το email το δημιουργούμε
        self.email=name+"_"+last+".@gmail.com"

#τα στοιχειά ενός εργαζομένου
emp1=Employee("Nikos","Mitrakas",100)
#εμφανίζουμε τα στοιχειά
print(emp1.name)
print(emp1.last)
print(emp1.email)
print(emp1.pay)
#τα στοιχειά ενός άλλου
emp2=Employee("Ioanna","Georgouvia",4440)
print("################")
#εμφανίζουμε τα στοιχειά
print(emp2.name)
print(emp2.last)
print(emp2.email)
print(emp2.pay)
```

Αποτέλεσμα

```
Nikos
Mitrakas
Nikos_Mitrakas.@gmail.com
100
################
Ioanna
Georgouvia
Ioanna_Georgouvia.@gmail.com
4440
```

Σχόλιο

Ορίζουμε κλάση -αρχικοποιούμε τα δεδομένα -καλούμε την κλάση.

1.8.3 Στην προηγούμενη κλάση ορίζουμε συνάρτηση που επιστρέφει όνομα επίθετο μαζί.

Πρόγραμμα

```
class Employee():
#Για κάθε εργαζόμενο χρειαζόμαστε όνομα, επίθετο, μισθό και email
    def __init__    (self,name,last,pay):
        self.name=name
        self.last=last
        self.pay=pay
    # το email το δημιουργούμε
        self.email=name+"_"+last+".@gmail.com"
    def fullname(self):
        return "{} {}".format(self.name, self.last)

#τα στοιχειά ενός εργαζόμενου
emp1=Employee("Nikos","Mitrakas",100)
#εμφανίζουμε το ονοματεπώνυμο
print(emp1.fullname())

#τα στοιχειά ενός άλλου
emp2=Employee("Ioanna","Georgouvia",4440)
print("################")
#εμφανίζουμε το ονοματεπώνυμο
print(emp2.fullname())
```

Αποτέλεσμα

```
Nikos Mitrakas
```

```
#################
Ioanna Georgouvia
```

Σχόλιο

Ορίζουμε μια συνάρτηση μέσα στην κλάση που εμφανίζει μαζί όνομα επίθετο

1.8.4 Πρόσθεση συνάρτησης στην προηγούμενη κλάση που θα υπολογίζει την 10% αύξηση επί του μισθού

Πρόγραμμα

```
class Employee():
#10%
    rate = 1.10
#Για κάθε εργαζόμενο χρειαζόμαστε όνομα, επίθετο, μισθό και email
    def __init__    (self,name,last,pay):
        self.name=name
        self.last=last
        self.pay=pay
    # το email το δημιουργούμε
        self.email=name+"_"+last+".@gmail.com"

    def fullname(self):
        return "{} {}".format(self.name, self.last)

    def raisee(self):  ##raise of the payment!
        return int(self.pay * Employee.rate)


#τα στοιχειά ενός εργαζόμενου
emp1=Employee("Nikos","Mitrakas",100)
#εμφανίζουμε τα στοιχειά
print("Πριν την αύξηση {} μετά την αύξηση {}".format(emp1.pay,emp1.rais
ee()))

#τα στοιχειά ενός άλλου
emp2=Employee("Ioanna","Georgouvia",4440)
print("#################")
#εμφανίζουμε τα στοιχεία
print("Πριν την αύξηση {} μετά την αύξηση {}".format(emp2.pay,emp2.rais
ee()))
```

Αποτέλεσμα

```
Πριν την αύξηση 100 μετά την αύξηση 110
#################
```

```
Πριν την αύξηση 4440 μετά την αύξηση 4884
```

Σχόλιο

Στην νέα συνάρτηση που ορίσαμε απλά πολλαπλασιάζουμε το μισθό με 1.1 για να βρούμε τον νέο μισθό μετά την αύξηση, και εμφανίζουμε μήνυμα . Ορίζουμε το rate στην αρχή.

1.9 ΕΥΡΕΣΗ ΤΥΧΑΙΩΝ ΑΡΙΘΜΩΝ

1.9.1 ΠΡΟΓΡΑΜΜΑ ΤΥΧΑΙΑΣ ΣΕΙΡΑΣ

Πρόγραμμα που επιστρέφει σε τυχαία σειρά όλα τα στοιχεία μιας λίστας από το 1 έως το 15

Πρόγραμμα

```
#εισάγουμε την βιβλιοθήκη
import random

#Δημιουργούμε λίστα από το 1 έως το 15
mylist = [i for i in range(1,16)]

#shuffle : ανακάτεμα των στοιχείων της λίστας τυχαία
random.shuffle(mylist)

#εμφανίζουμε το αποτέλεσμα
print(mylist)
```

Αποτέλεσμα

Τρέχουμε 3 φορές το πρόγραμμα

```
[13, 10, 11, 6, 4, 15, 7, 9, 14, 12, 2, 5, 8, 3, 1]
```

```
[7, 14, 8, 3, 5, 6, 9, 4, 12, 11, 1, 10, 2, 15, 13]
```

```
[4, 15, 12, 10, 9, 3, 14, 1, 2, 6, 5, 13, 8, 7, 11]
```

Σχόλιο

Τρέξαμε το πρόγραμμα 3 φορές και αναμενόμενα είχαμε διαφορετικά αποτελέσματα αφού είναι τυχαίοι αριθμοί. Η εντολή random.randrange(0,110,10) παράγει τυχαίους αριθμούς από το 0 μέχρι και το 100 με βήμα 10 .Το random μπροστά στο randrange(0,110,10) συμβολίζει την βιβλιοθήκη .Τέλος ανάλογα με το αποτέλεσμα με την βοήθεια του if statement εμφανίζεται κατάλληλο μήνυμα.

1.9.2 ΠΡΟΓΡΑΜΜΑ ΕΠΙΣΤΡΟΦΗΣ ΤΥΧΑΙΑΣ ΣΕΙΡΑΣ

Πρόγραμμα

```
#εισάγουμε την βιβλιοθήκη
import random

#Δημιουργούμε λίστα από το 1 έως το 10
mylist = [i for i in range(1,10)]

#shuffle : ανακάτεμα των στοιχείων της λίστας τυχαία
random.shuffle(mylist)

#εμφανίζουμε το αποτέλεσμα
print(mylist)
```

Αποτέλεσμα

Τρέχουμε 3 φορές το πρόγραμμα

```
[7, 8, 9, 10, 6, 3, 5, 1, 4, 2]
```

```
[9, 1, 3, 7, 8, 2, 10, 4, 6, 5]
```

```
[10, 4, 6, 9, 2, 5, 7, 8, 3, 1]
```

Σχόλιο

Κάθε φορά θα έχουμε άλλη σειρά (μπορεί να τύχει να είναι ίδια καμία φορά γιατί έχουμε μικρό δείγμα αλλά και πάλι μικρή η πιθανότητα)

1.9.3 Επιλογή τυχαίου άρτιου αριθμού

Πρόγραμμα

```
#εισάγουμε την βιβλιοθήκη
import random

#για συντομία σε μια γραμμή
print (random.choice([i for i in range(11) if i%2==0]))# mod(i,2)==0 σή
μαινε ι άρτιος αριθμός
```

Αποτέλεσμα

Τρέχουμε το πρόγραμμα 3 φορές

```
0
4
8
```

Σχόλιο

Κάθε φορά θα τρέχουμε το πρόγραμμα πιθανόν να εμφανίζει διαφορετικό αριθμό.

1.10 ΕΝΤΟΛΗ ASSERT

1.10.1 Πρόγραμμα που διασφαλίζει ότι η λίστα περιέχει μόνο άρτιους αριθμούς

Πρόγραμμα που διασφαλίζει ότι η λίστα περιέχει μόνο άρτιους αριθμούς. Η λίστα δημιουργείται από τον χρήστη

Πρόγραμμα

```
#Πρόγραμμα

#δημιουργία λίστας ακεραίων από τον χρήστη
```

```
data=map(int,input().split(","))
data1=list(data)
print(data1)

#εξασφαλίστε ότι η λίστα περιέχει μόνο αρτίους αριθμούς
for i in data1: assert i%2==0
```

Αποτέλεσμα

Τρέχουμε 2 φορές το πρόγραμμα

Την πρώτη φορά δίνουμε και περιττούς αριθμούς

```
12,3,5,14,8
[12, 3, 5, 14, 8]
Traceback (most recent call last):
  File "d:/Έγγραφα/repos/Enotita_10.Assertion.py", line 10, in <module>
    for i in data1: assert i%2==0
AssertionError
```

Τη δεύτερη φορά μόνο άρτιους

```
2,4,6,8,10,18
[2, 4, 6, 8, 10, 18]
```

Σχόλιο

Την πρώτη φορά που τρέξαμε το πρόγραμμα ο χρήστης έβαλε και περιττούς αριθμούς ,σαν αποτέλεσμα είχε να εμφανίσει AssertionError. Την δεύτερη φορά είχαμε μόνο άρτιους αριθμούς και δεν είχαμε πρόβλημα.

1.11 REGULAR EXPRESION

1.11.1 Συνάρτηση που θα επιστρέφει αν η λέξη "basketball" υπάρχει στην πρόταση

Συνάρτηση που θα επιστρέφει αν η λέξη «basketball» υπάρχει στην πρόταση. Στη συνέχεια θα καλέσουμε τη συνάρτηση 5 φορές

Πρόγραμμα

```
#εισάγουμε την re
import re

#Δημιουργούμε μια συνάρτηση
def text_match(text):
#pattern είναι αυτό που ψάχνουμε μέσα στην πρόταση
    pattern = 'basketball'

    if re.search(pattern, text):
#αν υπάρχει
        return 'Βρέθηκε!'
    else:
#αν δεν υπάρχει
        return('Δεν βρέθηκε!')

#Καλούμε την συνάρτηση 5 φορές
print(text_match("asdfsdffitdfs"))
print(text_match("dsfdsfbasketball efrew"))
print(text_match("asdewfevbasketballsref"))
print(text_match("aabbsdfsdfcbasketballs"))
print(text_match("aabbσαδφσαδω A3233ΔΨΑΨΔΦ TH4FWTF4"))
```

Αποτέλεσμα

```
Δεν βρέθηκε!
Βρέθηκε!
Βρέθηκε!
Βρέθηκε!
Δεν βρέθηκε!
```

Σχόλιο

Η εντολή re.search(pattern,data) ψάχνει να βρει αν η pattern υπάρχει στα data .Αν υπάρχει είναι True αλλιώς False.

2 ΠΡΟΓΡΑΜΜΑΤΑ ΓΛΩΣΣΑΣ R

2.1 ΕΝΤΟΛΕΣ TOUPPER ΚΑΙ TOLOWER

ΠΡΟΓΡΑΜΜΑ

```
names<-c("nikos","giorgos","kostas","Mitsos")
names
toupper(names)
#TOLOWER() PEZA
t="GeiasouGeorge"
```

```
tolower(t)
```

ΑΠΟΤΕΛΕΣΜΑ

```
[Running] Rscript "d:\Έγγραφα\repos\zis\tempCodeRunnerFile.r"
[1] "nikos"    "giorgos" "kostas"  "Mitsos"
[1] "NIKOS"    "GIORGOS" "KOSTAS"  "MITSOS"
[1] "geiasougeorge"

[Done] exited with code=0 in 0.441 seconds
```

ΣΧΟΛΙΟ

Η εντολή toupper() εμφανίζει τα στοιχεία ενός διανύσματος με κεφαλαία γράμματα ,εάν είναι ήδη κεφαλαία τότε τα εμφανίζει όπως είναι .Σε κάποιες περιπτώσεις υπάρχουν κεφάλαια και πεζά στο ίδιο στοιχείο τότε πάλι μετατρέπει τα πεζά σε κεφαλαία .Αντίστοιχα λειτουργεί το tolower().

2.2 ΔΗΜΙΟΥΡΓΙΑ CHARACTERS AS.CHARACTER-IS.CHARACTER

2.2.1 ΔΗΜΙΟΥΡΓΙΑ ΔΙΑΝΥΣΜΑΤΟΣ ΧΑΡΑΚΤΗΡΩΝ

```
############################DIANISMATA XAAKTIRWN
#DIMIOURGIA
teams<-c("paok","aek","olympiakos","larisa")
teams
```

ΑΠΟΤΕΛΕΣΜΑ

```
[1] "paok"       "aek"        "olympiakos" "larisa"
```

ΣΧΟΛΙΟ

Όπως και στα αριθμητικά διανύσματα που είδαμε παραπάνω μπορούμε να δημιουργήσουμε διάνυσμα χαρακτήρων με χρήση c().

2.2.2 ΕΝΤΟΛΗ CHARACTER

ΠΡΟΓΡΑΜΜΑ

```
#character() DIANISMA ME KENA
character(length=3)
```

ΑΠΟΤΕΛΕΣΜΑ

```
[1] "" "" ""
```

2.2.3 ΕΝΤΟΛΗ AS.CHARACTER

ΠΡΟΓΡΑΜΜΑ

```
#AS.CHARACTER()
a<-100:110
a
#apo arithmoi se xaraktires
as.character(a)
```

ΑΠΟΤΕΛΕΣΜΑ

```
[1] 100 101 102 103 104 105 106 107 108 109 110
[1] "100" "101" "102" "103" "104" "105" "106" "107" "108" "109" "110"
```

ΣΧΟΛΙΟ

Με χρήση της εντολής as.character() μπορούμε να μετατρέψουμε ένα αριθμητικό διάνυσμα σε διάνυσμα χαρακτήρων.

2.2.4 ΕΝΤΟΛΗ IS.CHARACTER

ΠΡΟΓΡΑΜΜΑ

```
#IS.CHARACTER() ELEXNOS AN EINAI XARAKTIRAS
#1 paradeigma
teams
is.character(teams)
#2 paradeigma
x<-c(1:10)
x
is.character(x)
#3 paradeigma
xx=c("anna","nikos",NA)
xx
is.character(xx)
```

ΑΠΟΤΕΛΕΣΜΑ

```
[1] "paok"       "aek"        "olympiakos" "larisa"
[1] TRUE
 [1]  1  2  3  4  5  6  7  8  9 10
```

```
[1] FALSE
[1] "anna"  "nikos" NA
[1] TRUE
```

ΣΧΟΛΙΟ

Η εντολή as.character() ελέγχει αν ένα στοιχείο ή ένα διάνυσμα είναι χαρακτήρας ,διάνυσμα χαρακτήρων αντίστοιχα.

3 ΠΡΟΓΡΑΜΜΑ ΓΛΩΣΣΑΣ SQL

3.1 ΔΗΜΙΟΥΡΓΙΑ DATABASE ΚΑΙ ΠΙΝΑΚΑ ΚΑΙ ΕΙΣΑΓΩΓΗ ΣΤΟΙΧΕΙΩΝ ΚΑΙ ΕΠΙΛΟΓΗ ΤΟΥΣ ΜΕ ΒΑΣΗ ΚΑΠΟΙΑ ΚΡΙΤΗΡΙΑ

Στο παρακάτω πρόγραμμα θα δημιουργήσουμε μια καινούρια database. Στη συνέχεια μέσα σε αυτή θα δημιουργήσουμε έναν καινούριο πίνακα δεδομένων κάποιων υπαλλήλων με όνομα Employees ο οποίος θα έχει 5 στήλες, η $1^{η}$ θα είναι το id του πίνακα, η $2^{η}$ το επίθετο , η $3^{η}$ το όνομα, η $4^{η}$ η πόλη του υπαλλήλου και η $5^{η}$ η χώρα στην οποία κατοικεί ο υπάλληλος.

Στη συνέχεια στον πίνακα θα εισάγουμε κάποιες τιμές και τέλος θα επιλέξουμε εκείνες τις σειρές του πίνακα που μένουν στην πόλη Κοζάνη.

```
--Δημιουργία Database

Create Database Data_Science

--Χρήση της Database
use Data_Science

--Δημιουργία Πίνακα Employees

CREATE TABLE Employees (
ID int,
LastName varchar(255),
FirstName varchar(255),
City varchar(255),
Country varchar(255)
);

--Εισαγωγή τιμών στον πίνακα
INSERT INTO Employees VALUES(1,'Papadopoulos','Giorgos','Athina','Greece');
INSERT INTO Employees VALUES(2,'Mitrakas','Nikos','Kozani','Greece');
INSERT INTO Employees VALUES(3,'Tsiotras','Nikos','Thessaloniki','Greece');
INSERT INTO Employees VALUES(1,'Georgouvia','Ioanna','Kozani','Greece');

--Επιλογή των υπαλλήλων που μένουν στην πόλη Κοζάνη
select * from Employees where City='Kozani';
```

ΑΠΟΤΕΛΕΣΜΑ

```
ID     LastName     FirstName    City   Country
2      Mitrakas     Nikos        Kozani Greece
1      Georgouvia   Ioanna       Kozani Greece
```