39. PANDAS II και Matplotlib

[ΕΠΑΝΑΛΗΨΗ ΜΑΘΗΜΑΤΩΝ 24-26]

39.0.1 Λύσεις προηγούμενων ασκήσεων

Άσκηση 1

Τυπώστε μια σειρά ημερομηνιών από την 1η Ιανουαρίου 2024 μέχρι και την 11η Φεβρουαρίου 2024

Λύση

```
import pandas as pd
date_series = pd.date_range(start = '01-01-2024', end = '02-
11-2024')
print(date_series)
```

Εκτύπωση

```
>>>
====== RESTART: C:/Users/NK/AppData/Local/Programs/Python/Python312/38_ex1.py ======
DatetimeIndex(['2024-01-01', '2024-01-02', '2024-01-03', '2024-01-04',
               '2024-01-05', '2024-01-06', '2024-01-07', '2024-01-08',
               '2024-01-09', '2024-01-10', '2024-01-11', '2024-01-12',
               '2024-01-13', '2024-01-14', '2024-01-15', '2024-01-16',
               '2024-01-17', '2024-01-18', '2024-01-19', '2024-01-20',
               '2024-01-21', '2024-01-22', '2024-01-23', '2024-01-24',
               '2024-01-25', '2024-01-26', '2024-01-27', '2024-01-28',
               '2024-01-29', '2024-01-30', '2024-01-31', '2024-02-01',
               '2024-02-02', '2024-02-03', '2024-02-04', '2024-02-05',
               '2024-02-06', '2024-02-07', '2024-02-08', '2024-02-09',
               '2024-02-10', '2024-02-11'],
              dtype='datetime64[ns]', freq='D')
>>>
```

Άσκηση 2

Εφαρμόστε μια απλή συνάρτηση σε μια δεδομένη σειρά pandas (pandas series). Η σειρά είναι: [3, 6, 9, 12, 15]

και η συνάρτηση προς εφαρμογή: f(x) = x/3.

Ζητήστε πρώτα την εκτύπωση της σειράς.

Χρησιμοποιήστε τη συνάρτηση apply() και lambda function για την εφαρμογή της συνάρτησης στα στοιχεία της σειράς.

Ζητήστε την εκτύπωση της αλλαγμένης σειράς.

Λύση

```python
import pandas as pd
series = pd.Series([3, 6, 9, 12, 15])
print(series) # εκτύπωση της αρχικής σειράς
print()
f_series = series.apply(lambda x:x/3)
print(f_series) # εκτύπωση της νέας σειράς
```

```
>>>
====== RESTART: C:/Users/NK/AppData/Local/Programs/Python/Python312/38_ex2.py ======
0     3
1     6
2     9
3    12
4    15
dtype: int64

0    1.0
1    2.0
2    3.0
3    4.0
4    5.0
dtype: float64
>>>
```

Άσκηση 3

Γράψτε ένα πρόγραμμα χρησιμοποιώντας τη βιβλιοθήκη pandas, το οποίο να προσθέτει, να αφαιρεί να πολλαπλασιάζει και να διαιρεί με τον αριθμό 2, δύο δεδομένες λίστες ακέραιων αριθμών.

Οι λίστες είναι: [2, 4, 6, 8, 10] και [1, 3, 5, 7, 9]

Λύση

```
import pandas as pd
ds1 = pd.Series([2, 4, 6, 8, 10])
ds2 = pd.Series([1, 3, 5, 7, 9])
ds = ds1 + ds2
print("Πρόσθεση 2 σειρών:")
print(ds)
print("Αφαίρεση 2 σειρών")
ds = ds1 - ds2
print(ds)
print("Πολλ/σιασμός 2 σειρών:")
ds = ds1 * ds2
print(ds)
print("Διαίρεση 2 σειρών με το 2:")
ds = ds1 / ds2
print(ds)
```

```
>>>
====== RESTART: C:/Users/NK/AppData/Local/Programs
Πρόσθεση 2 σειρών:
0     3
1     7
2    11
3    15
4    19
dtype: int64
Αφαίρεση 2 σειρών
0    1
1    1
2    1
3    1
4    1
dtype: int64
Πολλ/σιασμός 2 σειρών:
0     2
1    12
2    30
3    56
4    90
dtype: int64
Διαίρεση 2 σειρών με το 2:
0    2.000000
1    1.333333
2    1.200000
3    1.142857
4    1.111111
dtype: float64
>>>
```

39.1.1 Διόρθωση δεδομένων λανθασμένης μορφής

Συνεχίζουμε την ενημέρωση σε σχέση με τις δυνατότητες της βιβλιοθήκης Pandas.

Τα κελιά με δεδομένα λανθασμένης μορφής μπορεί να δυσκολέψουν, ή ακόμα και αδύνατη, την ανάλυση δεδομένων.

Για να το διορθώσουμε, έχουμε δύο επιλογές:. Είτε να αφαιρέσουμε τις σειρές είτε να μετατρέψουμε όλα τα κελιά των στηλών στην ίδια μορφή.
Πάμε να δούμε μια-μια τις λύσεις

Κατεβάζουμε το αρχείο date_data.csv

.

Στο πλαίσιο δεδομένων μας, έχουμε δύο κελιά με λάθος μορφή. Αν δούμε τις σειρές 22 και 26 στη στήλη "Date", πρέπει να έχουμε συμβολοσειρές που αντιπροσωπεύουν ημερομηνίες,

Πρώτα τρέχουμε τον κώδικα:

```
import pandas as pd

df = pd.read_csv('date_data.csv')
df['Date'] = pd.to_datetime(df['Date'])

print(df.to_string())
```

...όμως δεν τρέχει, καθότι δεν δέχεται τη μορφή της ημερομηνίας στην οποία θέλουμε να τροποποιήσουμε και δοκιμάζουμε με την παράμετρο format='mixed', η οποία και μας προτείνεται στο σφάλμα του IDLE.

Δηλ, συμπληρώνουμε στη γραμμή

```
df['Date'] = pd.to_datetime(df['Date'],
format='mixed')
```

Έτσι τώρα, έχουμε στην εκτύπωσή μας:

```
>>>
=== RESTART: C:\Users\NK\AppData\Local\Programs\Pyth
    Duration       Date  Pulse  Maxpulse  Calories
0         60 2020-12-01    110       130     409.1
1         60 2020-12-02    117       145     479.0
2         60 2020-12-03    103       135     340.0
3         45 2020-12-04    109       175     282.4
4         45 2020-12-05    117       148     406.0
5         60 2020-12-06    102       127     300.0
6         60 2020-12-07    110       136     374.0
7        450 2020-12-08    104       134     253.3
8         30 2020-12-09    109       133     195.1
9         60 2020-12-10     98       124     269.0
10        60 2020-12-11    103       147     329.3
11        60 2020-12-12    100       120     250.7
12        60 2020-12-12    100       120     250.7
13        60 2020-12-13    106       128     345.3
14        60 2020-12-14    104       132     379.3
15        60 2020-12-15     98       123     275.0
16        60 2020-12-16     98       120     215.2
17        60 2020-12-17    100       120     300.0
18        45 2020-12-18     90       112       NaN
19        60 2020-12-19    103       123     323.0
20        45 2020-12-20     97       125     243.0
21        60 2020-12-21    108       131     364.2
22        45        NaT    100       119     282.0
23        60 2020-12-23    130       101     300.0
24        45 2020-12-24    105       132     246.0
25        60 2020-12-25    102       126     334.5
26        60 2020-12-26    100       120     250.0
27        60 2020-12-27     92       118     241.0
28        60 2020-12-28    103       132       NaN
29        60 2020-12-29    100       132     280.0
30        60 2020-12-30    102       129     380.3
31        60 2020-12-31     92       115     243.0
>>>
```

Τώρα όμως παρατηρούμε ότι στη γραμμή 22 έχουμε και πάλι μια κενή τιμή (NaT = Not a Time) και η λύση που μας μένει είναι να αφαιρέσουμε όλη τη γραμμή.

39.1.2 Διόρθωση με διαγραφή γραμμών

Διαμορφώνουμε λοιπόν τον κώδικά μας ως εξής:

```
import pandas as pd
df = pd.read_csv('date_data.csv')
df['Date'] = pd.to_datetime(df['Date'], format
='mixed')
df.dropna(subset=['Date'], inplace = True)
print(df.to_string())
```

7

κα τώρα έχουμε:

```
>>>
=== RESTART: C:\Users\NK\AppData\Local\Programs\Pyt
    Duration       Date  Pulse  Maxpulse  Calories
0         60 2020-12-01    110       130     409.1
1         60 2020-12-02    117       145     479.0
2         60 2020-12-03    103       135     340.0
3         45 2020-12-04    109       175     282.4
4         45 2020-12-05    117       148     406.0
5         60 2020-12-06    102       127     300.0
6         60 2020-12-07    110       136     374.0
7        450 2020-12-08    104       134     253.3
8         30 2020-12-09    109       133     195.1
9         60 2020-12-10     98       124     269.0
10        60 2020-12-11    103       147     329.3
11        60 2020-12-12    100       120     250.7
12        60 2020-12-12    100       120     250.7
13        60 2020-12-13    106       128     345.3
14        60 2020-12-14    104       132     379.3
15        60 2020-12-15     98       123     275.0
16        60 2020-12-16     98       120     215.2
17        60 2020-12-17    100       120     300.0
18        45 2020-12-18     90       112       NaN
19        60 2020-12-19    103       123     323.0
20        45 2020-12-20     97       125     243.0
21        60 2020-12-21    108       131     364.2
23        60 2020-12-23    130       101     300.0
24        45 2020-12-24    105       132     246.0
25        60 2020-12-25    102       126     334.5
26        60 2020-12-26    100       120     250.0
27        60 2020-12-27     92       118     241.0
28        60 2020-12-28    103       132       NaN
29        60 2020-12-29    100       132     280.0
30        60 2020-12-30    102       129     380.3
31        60 2020-12-31     92       115     243.0
>>>
```

Τα λανθασμένα δεδομένα δεν είναι απαραίτητο να είναι κενές τιμές ή λάθος τύπος δεδομένων, μπορεί απλά να είναι λανθασμένα.

Κάποιες φορές, μπορούμε να διακρίνουμε αυτά τα δεδομένα απλώς κοιτάζοντας το data set, καθότι αντιλαμβανόμαστε τι μορφής πρέπει να είναι τα δεδομένα.

Αν δούμε τα προηγούμενα μόλις δεδομένα, θα δούμε ότι στη γραμμή 7, η διάρκεια (Duration) είναι 450, ενώ όλες οι άλλες τιμές είναι μεταξύ του 30 και του 60.

Δεν είναι απαραίτητο να είναι λάθος, αλλά έχοντας για παράδειγμα τη γνώση ότι αυτό είναι ένα data set για την ώρα εκγύμνασης κάποιου, συμπεραίνουμε πως η τιμή είναι λάθος, καθότι είναι σχετικά απρόσμενο το άτομο να γυμνάστηκε για 450 λεπτά μόνο για μια μέρα.

Πώς μπορούμε να διορθώσουμε τέτοιου είδους λανθασμένα δεδομένα;

39.1.3 Διόρθωση με αντικατάσταση τιμών, μια-μια ή με loops

Στο παράδειγμά μας, κατά πάσα πιθανότητα πρόκειται για ένα λάθος πληκτρολόγησης, όπου αντί για «45», πληκτρολογήθηκε «450».

Οπότε, μπορούμε να πάρουμε την απόφαση να αντικαταστήσουμε τη συγκεκριμένη τιμή στη γραμμή 7, με την τιμή «45».

Σε κώδικα έχουμε:

```
df.loc[7, 'Duration'] = 45
```

Δηλ.:

```
import pandas as pd

df = pd.read_csv('date_data.csv')

df.loc[7,'Duration'] = 45

print(df.to_string())
```

Σε μικρά data sets μπορούμε να αντικαταστήσουμε τα λανθασμένα δεδομένα, ένα προς ένα. Φυσικά αυτό είναι αδύνατο σε μεγαλύτερα data sets.

Σε τέτοιες περιπτώσεις μπορούμε να δημιουργήσουμε κάποιους κανόνες, για παράδειγμα να βάλουμε κάποια όρια για τις αποδεκτές τιμές και να αντικαταστήσουμε οποιεσδήποτε τιμές είναι έξω απ' αυτά.

Στο παρακάτω παράδειγμα, εφαρμόζουμε ένα loop στις τιμές της στήλης "Duration" και αν η τιμή είναι μεγαλύτερη από 120, την ορίζουμε σε «120»:

```
import pandas as pd

df = pd.read_csv('data.csv')

for x in df.index:
 if df.loc[x, "Duration"] > 120:
 df.loc[x, "Duration"] = 120

print(df.to_string())
```

Τμήμα της εκτύπωσής μας με τις αλλαγμένες τιμές, είναι:

83	120	100	130	500.0
84	45	100	120	225.3
85	30	151	170	300.1
86	45	102	136	234.0
87	120	100	157	1000.1
88	45	129	103	242.0
89	20	83	107	50.3
90	120	101	127	600.1
91	45	107	137	NaN
92	30	90	107	105.3
93	15	80	100	50.5
94	20	150	171	127.4
95	20	151	168	229.4
96	30	95	128	128.2
97	25	152	168	244.2
98	30	109	131	188.2
99	90	93	124	604.1
100	20	95	112	77.7
101	90	90	110	500.0
102	90	90	100	500.0
103	90	90	100	500.4
104	30	92	108	92.7
105	30	93	128	124.0
106	120	90	120	800.3
107	30	90	120	86.2
108	90	90	120	500.3
109	120	137	184	1860.4

Όπως ειπώθηκε και προηγουμένως, ένας άλλος τρόπος να αντιμετωπίσουμε τα λανθασμένα δεδομένα, είναι να διαγράψουμε τις γραμμές. Μ' αυτό τον τρόπο, δεν χρειάζεται να βρούμε μια τιμή αντικατάστασης και το επιλέγουμε αυτό, όταν κρίνουμε πως η συγκεκριμένη διαγραφή δεν θα επηρεάσει το αποτέλεσμα της ανάλυσής μας.

Έτσι, μπορούμε να εφαρμόσουμε το παραπάνω loop, ως εξής:

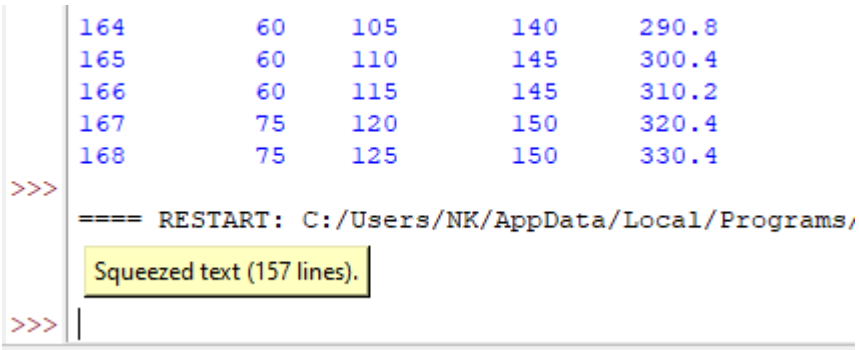
```
Import pandas as pd
df = pd.read_csv('data.csv')

for x in df.index:
 if df.loc[x, "Duration"] > 120:
 df.drop(x, inplace = True)

# Πρέπει να θυμόμαστε να συμπεριλαμβάνουμε τη
δήλωση 'inplace = True' ώστε οι αλλαγές να
γίνονται στο  αρχικό DataFrame αντί να παίρνουμε
ένα αντίγραφό του

print(df.to_string())
```

Τώρα πλέον, έχουμε 157 γραμμές και όχι 168:



39.1.4 Διόρθωση με απαλοιφή διπλοτύπων

Τώρα, κοιτάζοντας το δοκιμαστικό μας dataset, μπορούμε να δούμε ότι οι γραμμές 11 και 12 είναι διπλότυπα.

Για να ανακαλύψουμε και τα υπόλοιπα, μπορούμε να χρησιμοποιήσουμε τη συνάρτηση duplicated(), η οποία επιστρέφει τιμές Boolean για κάθε γραμμή, δηλαδή, True για κάθε γραμμή που

είναι διπλή, αλλιώς False:

```
import pandas as pd

df = pd.read_csv('date_data.csv')

print(df.duplicated())
```

Ένα δείγμα της εκτύπωσης του παραπάνω κώδικα είναι:

```
>>>
=== RESTART:
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11    False
12     True
13    False
14    False
```

Μπορούμε επίσης να απομακρύνουμε τελείως τις διπλότυπες εγγραφές:

```
import pandas as pd

df = pd.read_csv('date_data.csv')

df.drop_duplicates(inplace = True)

print(df.to_string())
```

39.1.5 Εύρεση σχέσεων – Data Correlations

Μια γνωστή μέθοδος της βιβλιοθήκης Pandas είναι η corr(). Αυτή η μέθοδος υπολογίζει τη σχέση κάθε στήλης με την άλλη.

Κατεβάζουμε ένα νέο αρχείο data.csv από δω, τρέχουμε τον παρακάτω κώδικα, για να δούμε τη σχέση των στηλών μεταξύ τους:

```
import pandas as pd
df = pd.read_csv('data2.csv')
print(df.corr())
```

Ας την τρέξουμε για να σχολιάσουμε την εκτύπωση:

```
>>>
=== RESTART: C:\Users\NK\AppData\Local\Programs\Py
              Duration     Pulse  Maxpulse  Calories
Duration      1.000000 -0.155408  0.009403  0.922717
Pulse        -0.155408  1.000000  0.786535  0.025121
Maxpulse      0.009403  0.786535  1.000000  0.203813
Calories      0.922717  0.025121  0.203813  1.000000

>>>
```

ΣΗΜΕΙΩΣΗ: Η μέθοδος corr() αγνοεί τις μη αριθμητικές στήλες.
Το αποτέλεσμα της corr()μεθόδου είναι ένας πίνακας με πολλούς αριθμούς που αντιπροσωπεύει πόσο καλή είναι η σχέση μεταξύ δύο στηλών.

Κάθε αριθμός ποικίλλει από το -1 έως το 1.

Το 1 σημαίνει ότι υπάρχει μια σχέση 1 προς 1 (μια τέλεια συσχέτιση) και για αυτό το data set, κάθε φορά που μια τιμή ανέβαινε στην πρώτη στήλη, ανέβαινε και η τιμή στην άλλη.

Το 0,9 είναι επίσης μια καλή σχέση, και αν αυξήσουμε τη μία τιμή, πιθανότατα θα αυξηθεί και η άλλη.

Το -0,9 θα ήταν εξίσου καλή σχέση με το 0,9, αλλά αν αυξήσετε τη μία τιμή, η άλλη πιθανότατα θα μειωθεί.

Το 0,2 σημαίνει ΟΧΙ καλή σχέση, που σημαίνει ότι αν η μία τιμή ανεβαίνει δεν σημαίνει ότι και η άλλη θα αυξηθεί.

Τι είναι ένας καλός συσχετισμός; Εξαρτάται από τη χρήση, αλλά είναι ασφαλές να πούμε ότι πρέπει να έχουμε τουλάχιστον 0.6(ή -0.6) για να έχουμε έναν καλό συσχετισμό.

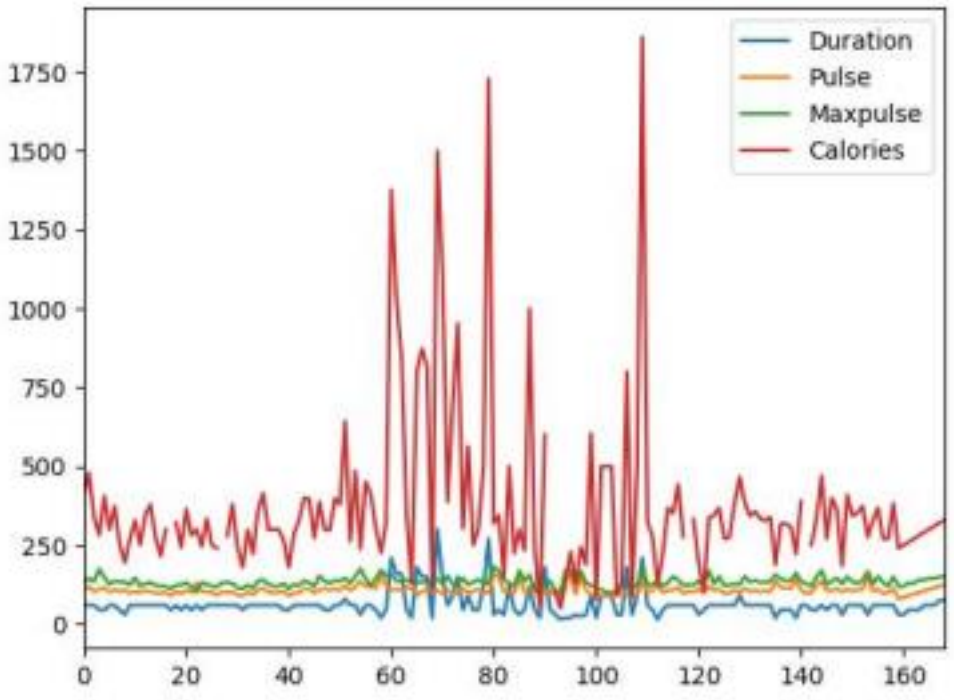
Συνεχίζοντας τον σχολιασμό μας, παρατηρούμε ότι η στήλες “Duration” και “Duration” έχουν την τέλεια συσχέτιση (1). Αυτό είναι φυσιολογικό για κάθε στήλη, καθότι η καθεμιά συσχετίζεται με τον εαυτό της.

Η στήλη "Duration" και "Calories" έχουν μια συσχέτιση 0.922721, η οποία είναι ένας πολύ καλός συσχετισμός, και μπορούμε να προβλέψουμε ότι όσο περισσότερο γυμνάζεται κάποιος, τόσο περισσότερες θερμίδες καίει και το αντίστροφο: αν κάψουμε πολλές θερμίδες, μάλλον εξασκηθήκαμε για πολλή ώρα.

Οι στήλες "Duration" και "Maxpulse" έχουν μια συσχέτιση 0.009403, η οποία είναι πολύ κακή συσχέτιση, που σημαίνει ότι δεν μπορούμε να προβλέψουμε τον μέγιστο παλμό κοιτάζοντας απλώς τη διάρκεια της εκγύμνασης και το αντίστροφο.

39.1.6 Απεικόνιση διαγραμμάτων

Ας δούμε το παρακάτω διάγραμμα:



Το Pandas χρησιμοποιεί τη plot()μέθοδο για να δημιουργεί διαγράμματα.

Μπορούμε να χρησιμοποιήσουμε το Pyplot, μια υπομονάδα της βιβλιοθήκης Matplotlib για να απεικονίσουμε το διάγραμμα στην οθόνη.

Όμως, θα πρέπει πρώτα να εγκαταστήσουμε και κατόπιν να εισάγουμε τη βιβλιοθήκη:

```
pip install matplotlib
```

Εισαγάγετε pyplot από το Matplotlib και οπτικοποιήστε το DataFrame μας:

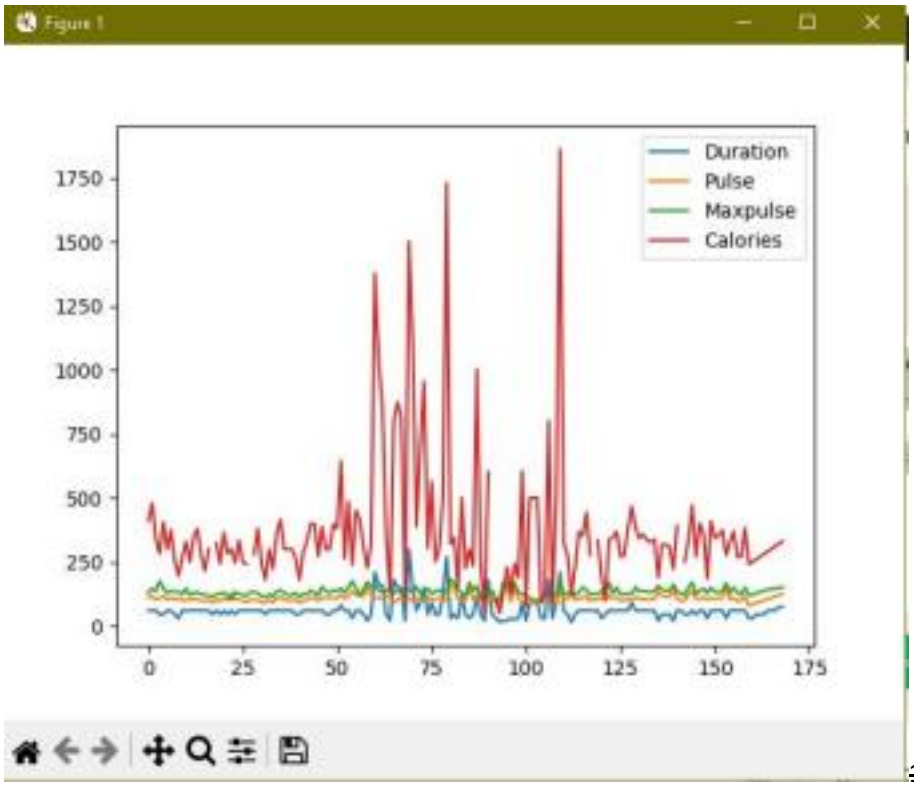
```
import pandas as pd

import matplotlib.pyplot as plt

df = pd.read_csv('data2.csv')

df.plot()

plt.show()
```



~~18~~

Για να ορίσουμε ένα διάγραμμα διασποράς χρησιμοποιούμε την έκφραση:

kind = 'scatter'

Ένα διάγραμμα διασποράς χρειάζεται άξονες x και y.

Στο παρακάτω παράδειγμα θα χρησιμοποιήσουμε τη στήλη "Duration" για τον άξονα x και "Calories" για τον άξονα y.

Ορίζουμε τα ορίσματα x και y ως εξής:

x = 'Duration', y = 'Calories'

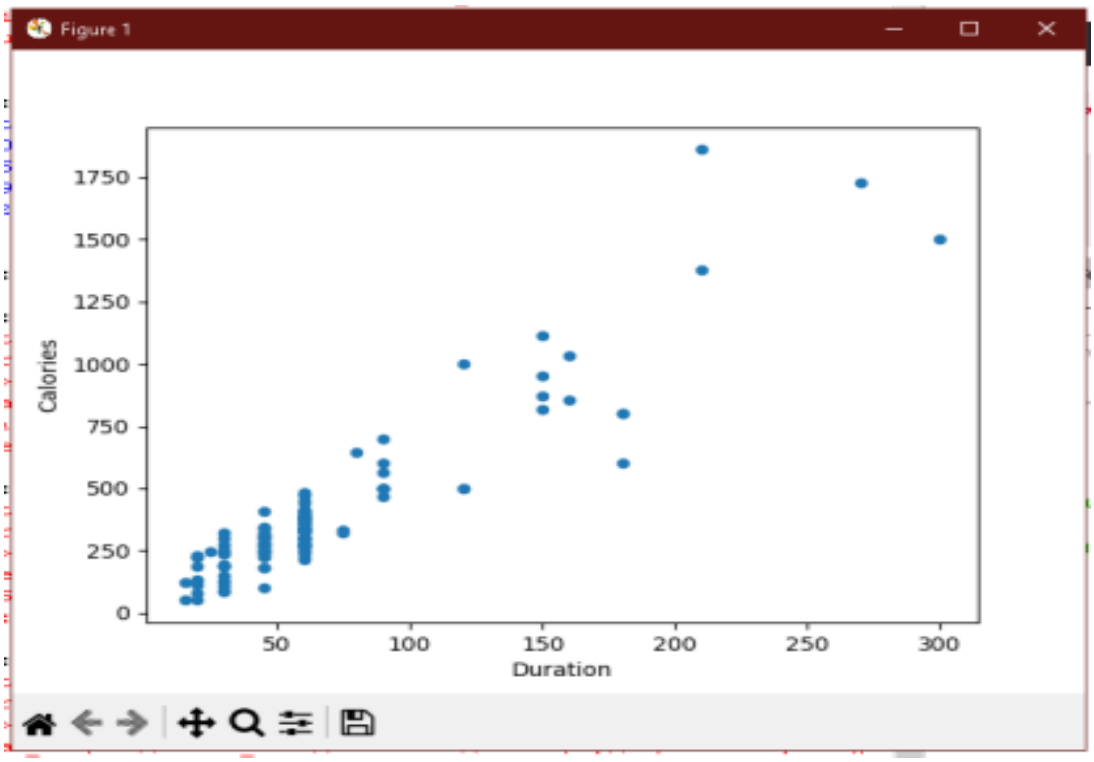
```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv(data2.csv')

df.plot(kind = 'scatter', x = 'Duration', y
= 'Calories')

plt.show()
```

και το διάγραμμα που παίρνουμε είναι:

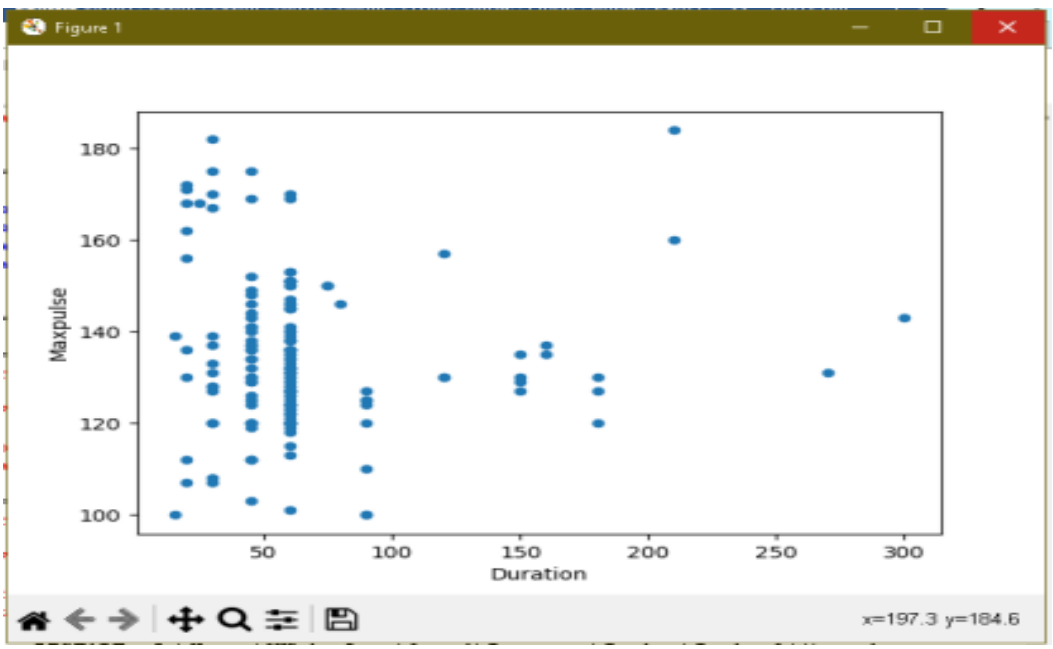


ΣΗΜΕΙΩΣΗ: Στο προηγούμενο παράδειγμα, είδαμε ότι η συσχέτιση μεταξύ "Duration" και "Calories" ήταν 0.922721, και καταλήξαμε στο γεγονός ότι μεγαλύτερη διάρκεια σημαίνει περισσότερες θερμίδες που καίγονται. Βλέποντας το scatterplot, μπορούμε αμέσως να έχουμε αυτή την εικόνα.

39.1.7 Μελέτη διαγραμμάτων

Ας δημιουργήσουμε ένα άλλο scatterplot, όπου υπάρχει κακή σχέση μεταξύ των στηλών, όπως , μεταξύ των στηλών"Duration" και "Maxpulse", με τη συσχέτιση 0.009403:

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('data2.csv')
df.plot(kind = 'scatter', x = 'Duration', y =
'Maxpulse')
plt.show()
```



Θα χρησιμοποιήσουμε τώρα το όρισμα kind για να δηλώσουμε ότι θέλουμε ένα ιστόγραμμα:

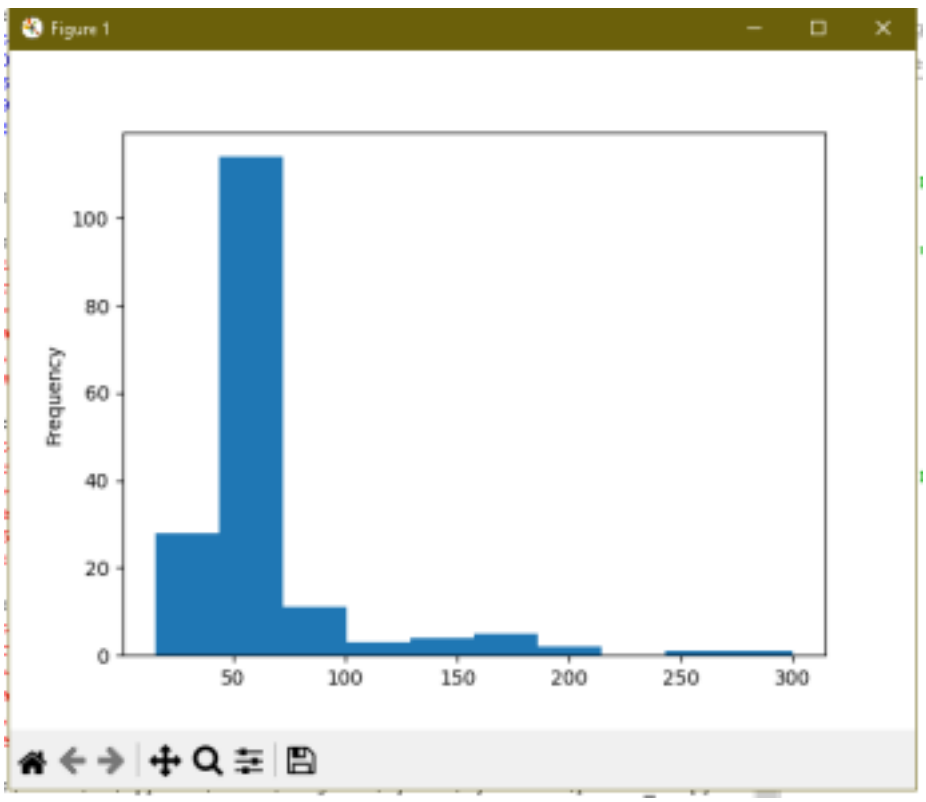
kind = 'hist'

Ένα ιστόγραμμα χρειάζεται μόνο μία στήλη και μας δείχνει τη συχνότητα κάθε διαστήματος, π.χ. πόσες προπονήσεις διήρκησαν μεταξύ 50 και 60 λεπτών;

Στο παρακάτω παράδειγμα θα χρησιμοποιήσουμε τη στήλη "Duration" για να δημιουργήσουμε το ιστόγραμμα:

Παράδειγμα

```
df["Duration"].plot(kind = 'hist')
```



39.2.0 Παραδείγματα για εξάσκηση

Άσκηση 1

Δημιουργήστε ένα DataFrame από ένα λεξικό με τα παρακάτω δεδομένα:

data = {'name': ['Alice', 'Bob', 'Charlie', 'David'],

'age': [25, 30, 22, 28]}

Λύση 1

```
import pandas as pd
df = pd.DataFrame({'name': ['Alice', 'Bob', 'Charlie',
'David'], 'age': [25, 30, 22, 28]})
print(df)
```

Παράδειγμα 2

Εξάγετε το παραπάνω dataframe στο αρχείο ‘data3.csv’ ώστε να μπορούμε να το επεξεργαζόμαστε περαιτέρω

Λύση 2.1

```
import pandas as pd

# Δημιουργία του DataFrame
data = {'name': ['Alice', 'Bob', 'Charlie', 'David'],
        'age': [25, 30, 22, 28]}
df = pd.DataFrame(data)

# Εξαγωγή σε αρχείο CSV
df.to_csv('data3.csv', index=False)
```

Λύση 2.2

Με την παραπάνω λύση, δεν γνωρίζουμε αν το αρχείο μας δημιουργήθηκε ή όχι. Πώς θα μπορούσαμε να προσθέσουμε ένα try-except block ώστε να παίρνουμε ένα μήνυμα επιτυχίας ή αποτυχίας της δημιουργίας του αρχείου μας;

```
import pandas as pd

# Δημιουργία του DataFrame
data = {'name': ['Alice', 'Bob', 'Charlie', 'David'],
        'age': [25, 30, 22, 28]}
df = pd.DataFrame(data)

try:
    # Εξαγωγή σε αρχείο CSV
    df.to_csv('data3.csv', index=False)
    print("Το αρχείο data3.csv δημιουργήθηκε επιτυχώς.")
except Exception as e:
    print(f"Σφάλμα κατά την εξαγωγή του αρχείου CSV: {e}")
```

39.2.1

Παράδειγμα 3

Προσθέστε μια νέα στήλη "city" με τιμές "New York", "San Francisco", "Los Angeles", "Chicago" στο DataFrame από την προηγούμενη άσκηση.

Λύση 3

```
import pandas as pd
df = pd.read_csv('data3.csv')
df['city'] = ['New York', 'San Francisco', 'Los Angeles',
'Chicago']
print(df)
```

Παράδειγμα 4

Εμφανίστε μόνο τις πρώτες 2 γραμμές του DataFrame.

Λύση 4

```
import pandas as pd
df = pd.read_csv('data3.csv')
print(df.head(2))
```

Παράδειγμα 5

Υπολογίστε τον μέσο όρο των ηλικιών στο DataFrame.

Λύση 5

```
import pandas as pd
df = pd.read_csv('data3.csv')
average_age = df['age'].mean()
print(f"Ο μέσος όρος των ηλικιών είναι: {average_age}")
```

39.2.2

Παράδειγμα 6

Σχεδιάστε ένα διάγραμμα γραμμής με τις ηλικίες.

Λύση 6

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('data3.csv')
plt.plot(df['name'], df['age'], marker='o')
```

```
''' marker='o' σημαίνει ότι κάθε σημείο θα εμφανίζεται ως κουκίδα (circle).
Αυτό καθιστά ευκολότερη την αναγνώριση των σημείων στο γράφημα.
Άλλες επιλογές για την παράμετρο marker περιλαμβάνουν,
's' για τετράγωνα, '^' για τρίγωνα, κλπ'''
plt.xlabel('Ονόματα')
plt.ylabel('Ηλικίες')
plt.title('Διάγραμμα Ηλικιών')
plt.show()
```

Παράδειγμα 7

Φιλτράρετε τις γραμμές του DataFrame όπου η ηλικία είναι μικρότερη από 30.

Λύση 7

```
import pandas as pd

df = pd.read_csv('data3.csv')
filtered_df = df[df['age'] < 30]
print(filtered_df)
```

39.2.3 Ασκήσεις

Άσκηση 1

Δημιουργήστε ένα DataFrame που περιέχει τυχαία δεδομένα για τις στήλες "Όνομα", "Ηλικία", "Βαθμολογία" για 10 φοιτητές. Χρησιμοποιήστε ένα for loop για να το κάνετε αυτό. Η στήλη "Βαθμολογία" θα πρέπει να περιέχει τυχαίους αριθμούς μεταξύ 0 και 100. Θα εισάγετε επίσης τη βιβλιοθήκη numpy, και σαν πρώτη γραμμή του προγράμματός σας, εισάγετε «np.random.seed(42)» (ο

αριθμός 42 είναι τυχαίος και χρησιμοποιείται ώστε αν και λαμβάνουμε τυχαία δεδομένα, να είναι πάντα τα ίδια).

39.2.4

Άσκηση 2

Χρησιμοποιήστε το παραπάνω dataframe, για να υπολογίστε τον μέσο όρο της "Βαθμολογίας" για τους φοιτητές που είναι 20 χρόνων και έχουν βαθμολογία πάνω από 50.

39.2.5

Άσκηση 3

Χρησιμοποιήστε το παραπάνω dataframe, για να σχεδιάστε ένα ιστόγραμμα για τις ηλικίες των φοιτητών, με διαίρεση σε 3 διαστήματα ηλικίας: 18-20, 21-23, 24-26 (θα χρησιμοποιήσετε την παράμετρο “bins” για τη διαίρεση των διαστημάτων). Για διευκόλυνσή σας, η παράμετρος θα είναι bins = [18, 20, 23, 26]. Προσθέστε τον κώδικα στο αρχείο .py που δημιουργήσατε προηγουμένως.

ΚΑΛΗ ΜΕΛΕΤΗ