

---

# 38. ΕΠΙΣΤΗΜΗ ΔΕΔΟΜΕΝΩΝ ΚΑΙ PANDAS

---

[ΕΠΑΝΑΛΗΨΗ ΜΑΘΗΜΑΤΟΣ 24]

## 38.0.1 Τι είναι η Επιστήμη των δεδομένων και η Βιβλιοθήκη Pandas

Η επιστήμη των δεδομένων είναι ένας κλάδος της Επιστήμης των Υπολογιστών στον οποίο μελετάμε τη χρήση, αποθήκευση και ανάλυση των δεδομένων ώστε να μπορούμε να εξάγουμε χρήσιμες πληροφορίες απ' αυτά.

Η βιβλιοθήκη Pandas είναι φτιαγμένη ειδικά γι' αυτό το λόγο. Έτσι, διαθέτει κατάλληλες λειτουργίες για τον χειρισμό, την ανάλυση και την εξερεύνηση δεδομένων. Το όνομα της βιβλιοθήκης είναι ένας συγκερασμός των λέξεων "Panel Data" ή "Python Data Analysis". Η βιβλιοθήκη δημιουργήθηκε από τον Wes McKinney το 2008. Ο Wes McKinney είναι προγραμματιστής, επιχειρηματίας και συγγραφέας και μπορούμε να βρούμε την 3<sup>η</sup> έκδοση του βιβλίου του "Python for Data Analysis", [εδώ](#)

### Τι μπορούμε να κάνουμε με την Pandas

Η Pandas είναι μια βιβλιοθήκη, η οποία θα μπορούσαμε να πούμε πως είναι ένας πρόδρομος της τεχνητής νοημοσύνης, καθώς μπορούμε χρησιμοποιώντας τη να υποβάλλουμε ερωτήματα σχεδόν σε φυσική γλώσσα (αγγλικά) όπως:

- Υπάρχει συσχέτιση μεταξύ δύο ή περισσότερων στηλών;
- Ποια είναι η μέση τιμή;
- Μέγιστη τιμή;
- Ελάχιστη τιμή;

Επίσης, μπορούμε να κάνουμε καθαρισμό των δεδομένων, όπως διαγραφή σειρών που δεν μας ενδιαφέρουν ή περιέχουν άσχετες ή και λανθασμένες τιμές, όπως κενές τιμές ή τιμές NULL. Αυτό το χαρακτηριστικό της βιβλιοθήκης ονομάζεται καθαρισμός.

Η βιβλιοθήκη Pandas είναι open source και το αποθετήριο της βρίσκεται στο github, [εδώ](#).

### 38.1.1 Εγκατάσταση και εισαγωγή

Η εγκατάσταση γίνεται όπως και με κάθε βιβλιοθήκη της Python, φυσικά με προϋποθέσεις την ύπαρξη της Python και του pip.

```
pip install pandas
```

#### Import Pandas

Με την εντολή import εισάγουμε και χρησιμοποιούμε τη βιβλιοθήκη όπως παρακάτω:

```
import pandas as pd
mydataset = {
    'cars': ["BMW", "Volvo", "Ford"],
    'passings': [3, 7, 2]
}

myvar = pd.DataFrame(mydataset)

print(myvar)
```

Η έξοδος είναι:

```
>>>
=== RESTART: C:\Users\NK\AppData\Local\Programs\Python\Python311\pandas_ex.py ===
   cars  passings
0   BMW         3
1  Volvo         7
2   Ford         2
>>>
```

Όπως και άλλες φορές, με τις βιβλιοθήκες μπορούμε να χρησιμοποιούμε aliases (ψευδώνυμα). Επίσης, μπορούμε όποτε θέλουμε να ελέγχουμε και την έκδοση της Pandas ή οποιασδήποτε βιβλιοθήκης, με την εντολή:

```
import pandas as pd
print(pd.__version__)
```

και η έξοδος στο IDLE είναι:

```
>>>
=== RESTART: C:\Users\NK\AppData\Local\Programs\Python\Python311\pandas_ex.py ===
2.1.0
>>>
```

## 38.1.2 Τι είναι οι σειρές – Series

Οι Pandas Series είναι όπως οι στήλες σε έναν πίνακα. Είναι ένας μονοδιάστατος πίνακας ο οποίος μπορεί να αποθηκεύσει δεδομένα οποιουδήποτε τύπου.

Δημιουργία Pandas Series από μια λίστα:

```
import pandas as pd

a = [1, 7, 2]

myvar = pd.Series(a)

print(myvar)
```

3

Η έξοδος τώρα είναι:

```
>>> = RESTART: C:\Users\NK\AppData\Local\Programs\Python\Python311\pandas_ex.py
0    1
1    7
2    2
dtype: int64
>>> |
```

Αν αυτή η εκτύπωση μας φαίνεται παράξενη, είναι διότι δεν έχουμε ακόμα αναφέρει τις ετικέτες.

### 38.1.3 Τι είναι οι ετικέτες – Labels

Πρόκειται για την αρίθμηση των τιμών. Αν δεν έχει αναφερθεί κάτι άλλο, τότε οι τιμές «ετικετάρονται» με τον index number τους, όπως στην παραπάνω εκτύπωση. Η αρίθμηση ξεκινάει από το «0» και μπορούμε να χρησιμοποιούμε τις ετικέτες για να έχουμε πρόσβαση στις τιμές τους, όπως παρακάτω.

Επιστροφή της 1<sup>ης</sup> τιμής (με index (0)):

```
import pandas as pd

a = [1, 7, 2]

myvar = pd.Series(a)

print(myvar[0])
```

Η έξοδος είναι: "1"

## Δημιουργία ετικετών

Με την παράμετρο `index`, μπορούμε να δημιουργούμε τις δικές μας ετικέτες:

```
import pandas as pd
```

```
a = [2, 9, 3]

myvar = pd.Series(a, index = ["a", "b",
"c"]) print(myvar)
```

Η έξοδος είναι:

```
>>>
=== RESTART: C:\Users\NK\AppData\Local\Programs\Python\Python311\pandas_ex.py ==
a      2
b      9
c      3
dtype: int64
>>> |
```

Όπως και παραπάνω, όταν έχουμε δημιουργήσει ετικέτες, μπορούμε να προσπελάσουμε μια τιμή με την ετικέτα της:

```
Print(myvar["b"])
```

Η έξοδος είναι: «9».

Οι παραπάνω δηλώσεις, μας θυμίζουν λεξικά. Πράγματι, μπορούμε να δημιουργήσουμε Series από λεξικά.

### 38.1.4 Αντικείμενα τύπου κλειδί: τιμή

```
import pandas as pd

lessons =
{"day1": algebra, "day2": Language, "day3":
Geography }

myvar = pd.Series(calories)

print(myvar)
```

Έξοδος:

5

```
>>>
- RESTART: C:\Users\NK\AppData\Local\Programs\Python\Python311\pandas_ex.py
day1    420
day2    380
day3    390
dtype: int64
>>> |
```

Ή μπορούμε να επιλέξουμε τα δεδομένα που χρειαζόμαστε, χρησιμοποιώντας τα index numbers:

```
import pandas as pd

lessons = {"day1": 420, "day2": 380, "day3": 390}

myvar = pd.Series(lessons, index = ["day1",
"day2"]) print(myvar)
```

```
>>>
=== RESTART: C:\Users\NK\AppData\Local\Programs\Python\Python311\pandas_ex.py ===
day1    420
day2    380
dtype: int64
>>> |
```

Τι είναι τα DataFrames

Τα data sets όσον αφορά τη βιβλιοθήκη Pandas, είναι συνήθως πολυδιάστατοι πίνακες που ονομάζονται DataFrames.

Αν Series είναι μια στήλη, τότε DataFrame είναι όλος ο πίνακας.

## 38.1.5 Δημιουργία dataframe με δύο series

```
import pandas as pd

data = {
    "calories": [420, 380, 390],
    "duration": [50, 40, 45]
}

myvar = pd.DataFrame(data)
print(myvar)
```

έτσι έχουμε:

```
>>> |
=== RESTART: C:\Users\NK\AppData\Local\Programs\Python\Python311\pandas_ex.py ==
day1    420
day2    380
dtype: int64
>>> |
```

Έτσι μπορούμε να πούμε ότι DataFrame είναι μια δομή διδιάστατη, όπως ένας διδιάστατος πίνακας, με γραμμές και στήλες.

Ας δούμε ένα DataFrame:

```
data = {
    "calories": [420, 380, 390],
    "duration": [50, 40, 45]
}
```

```
#load data into a DataFrame object:
df = pd.DataFrame(data)
```

```
print(df)
```

```
>>> |
=== RESTART: C:\Users\NK\AppData\Local\Programs\Python\Python311\pandas_ex.py ==
   calories  duration
0        420         50
1        380         40
2        390         45
>>> |
```

## 38.1.6 Ανεύρεση γραμμής σε ένα dataframe

Η Pandas, χρησιμοποιεί το χαρακτηριστικό «loc» για να επιστρέψει μία ή περισσότερες γραμμές.



Παράδειγμα αναφοράς στην πρώτη γραμμή του DataFrame με το index (0):

```
import pandas as pd
```

7

```
data = {  
    "calories": [420, 380, 390],  
    "duration": [50, 40, 45]  
}  
  
#load data into a DataFrame object:  
df = pd.DataFrame(data)  
  
print(df.loc[0])
```

Έξοδος;

```
>>> |  
    === RESTART: C:\Users\NK\AppData\Local\Programs\Python\Python311\pandas_ex.py ==  
    calories    420  
    duration    50  
    Name: 0, dtype: int64  
>>> |
```

Επιστροφή 2 γραμμών, χρησιμοποιώντας λίστα indexes:

```
print(df.loc[[0, 1]])
```

```
>>> |  
    === RESTART: C:\Users\NK\AppData\Local\Programs\Python\Python311\pandas_ex.py ==  
    calories  duration  
0         420         50  
1         380         40  
>>> |
```

## 38.1.7 Ονομαζόμενα indexes

Με τα indexes, όπως κάναμε και παραπάνω, μπορούμε να ονομάζουμε τις αριθμήσεις μας:

```
import pandas as pd

data = {
    "calories": [420, 380, 390],
    "duration": [50, 40, 45]
}

df = pd.DataFrame(data, index =
    ["day1", "day2", "day3"])

print(df)
```

Έξοδος:

```
>>>
=== RESTART: C:\Users\NK\AppData\Local\Programs\Python\Python311\pandas_ex.py ==
      calories  duration
day1         420         50
day2         380         40
day3         390         45
>>>
```

Έτσι, μπορούμε να επιστρέψουμε οποιοδήποτε index, με μια απλή αναφορά στο όνομά του.

## Ανεύρεση ονομαζόμενου index

```
print(df.loc["day2"])
```

```
>>> == RESTART: C:\Users\NK\AppData\Local\Programs\Python\Python311\pandas_ex.py ==
calories      380
duration       40
Name: day2, dtype: int64
>>>
```

### 38.1.8 Φόρτωση αρχείων σε DataFrame

Αν τα datasets βρίσκονται σε ένα αρχείο, η βιβλιοθήκη Pandas, μπορεί να τα φορτώσει σε ένα DataFrame. Τα δεδομένα πρέπει να βρίσκονται σε αρχείο csv.

### Παράδειγμα:

```
import pandas as pd

df = pd.read_csv('C:\\Users\\NK\\Desktop\\data.csv')

print(df)
```

Έξοδος:

```

>>>
--- RESTART: C:\Users\NK\AppData\Local\Programs\Python\Python311\pandas_ex.py ---
   Duration  Pulse  Maxpulse  Calories
0         60    110      130     409.1
1         60    117      145     479.0
2         60    103      135     340.0
3         45    109      175     282.4
4         45    117      148     406.0
..      ...    ...      ...      ...
164        60    105      140     290.8
165        60    110      145     300.4
166        60    115      145     310.2
167        75    120      150     320.4
168        75    125      150     330.4

[169 rows x 4 columns]
>>>

```

## 38.1.9 Άνοιγμα αρχείων

Τα αρχεία csv είναι απλά αρχεία κειμένου και χρησιμοποιούνται ευρέως για την αποθήκευση μεγάλων data sets.

Μπορούν, όπως ήδη είδαμε να διαβαστούν από το Pandas.

Μπορούμε να κατεβάσουμε ένα αρχείο csv, το data.csv, από [δω](#)

Τώρα, ας το φορτώσουμε σε ένα DataFrame:

```

import pandas as pd

df =

pd.read_csv('C:\\Users\\NK\\Desktop\\data.csv')

print(df.to_string())

```

Από το IDLE θα πάρουμε μια έξοδο με συμπιεσμένο περιεχόμενο, όπως γίνεται και σε κάθε περιεχόμενο, όταν είναι αρκετά μεγάλο για εκτύπωση.

Εδώ, χρησιμοποιούμε την to\_string() για να πάρουμε την

εκτύπωση όλου του DataFrame.

Αν το DataFrame είναι πολύ μεγάλο, με αρκετές γραμμές, το Pandas θα φορτώσει μόνο τις 5 πρώτες και τις 5 τελευταίες γραμμές.

Ας το δούμε, χωρίς τη χρήση της `to_string()`

```
import pandas as pd

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

print(df)
```

Παίρνουμε:

```
>>>
=== RESTART: C:\Users\NK\AppData\Local\Programs\Python\Python311\pandas_ex.py ===
   Duration  Pulse  Maxpulse  Calories
0         60    110      130     409.1
1         60    117      145     479.0
2         60    103      135     340.0
3         45    109      175     282.4
4         45    117      148     406.0
..      ...    ...      ...      ...
164        60    105      140     290.8
165        60    110      145     300.4
166        60    115      145     310.2
167        75    120      150     320.4
168        75    125      150     330.4

[169 rows x 4 columns]
>>>
```

Ο παραπάνω αριθμός των 5 και 5 επιστρεφόμενων γραμμών, είναι καθορισμένος στις επιλογές ρυθμίσεων του Pandas. Μπορούμε να τον ελέγξουμε:

```
import pandas as pd

print(pd.options.display.max_rows)
```

Στα περισσότερα συστήματα, από προεπιλογή, θα πάρουμε σαν έξοδο τον αριθμό «60».

Αυτό σημαίνει ότι αν το Dataframe περιέχει περισσότερες από 60

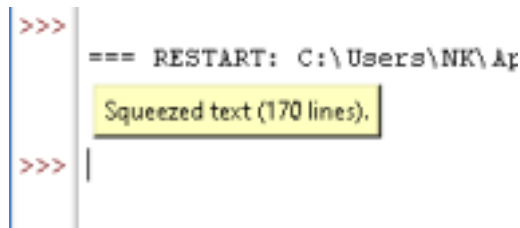
γραμμές, η δήλωση `print(df)` θα επιστρέψει μόνο τις επικεφαλίδες των στηλών και τις πρώτες και τελευταίες 5 στήλες. Μπορούμε να αλλάξουμε τον μέγιστο αριθμό των γραμμών με την ίδια δήλωση, δίνοντας τον δικό μας:

```
import pandas as pd

pd.options.display.max_rows = 9999

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

print(df)
```



```
>>>
>>>
>>>
```

=== RESTART: C:\Users\NK\AppData\Local\Temp\ipykernel\_12345\12345.py ===

Squeezed text (170 lines).

### 38.1.10 Ανάγνωση αρχείων json

Τα μεγάλα σύνολα δεδομένων (big data) συχνά αποθηκεύονται ή εξάγονται ως αρχεία JSON.

Τα JSON είναι αρχεία απλού κειμένου, αλλά έχουν τη μορφή ενός αντικειμένου και είναι πολύ γνωστά στον κόσμο του προγραμματισμού, συμπεριλαμβανομένου του Pandas.

Στα παραδείγματά μας θα χρησιμοποιήσουμε ένα αρχείο JSON που ονομάζεται 'data.json' και μπορούμε να το κατεβάσουμε από [δω](#).

```
import pandas as pd

df = pd.read_json('data.json')

print(df.to_string())
```

## Λεξικό ως JSON

Τα αντικείμενα JSON έχουν την ίδια μορφή με τα λεξικά Python.

Εάν ο κώδικας JSON δεν βρίσκεται σε αρχείο, αλλά σε λεξικό της Python, μπορούμε να τον φορτώσουμε απευθείας σε ένα DataFrame:

Παράδειγμα:

```
import pandas as pd

data = {
    "Duration":{
        "0":60,
        "1":60,
        "2":60,
        "3":45,
        "4":45,
        "5":60 },
    "Pulse":{
        "0":110,
        "1":117,
        "2":103,
        "3":109,
        "4":117,
        "5":102 },
    "Maxpulse":{
        "0":130,
        "1":145,
        "2":135,
        "3":175,
        "4":148,
        "5":127 },
```

```
"Calories":{
  "0":409,
  "1":479,
  "2":340,
  "3":282,
  "4":406,
  "5":300 }
}
```

```
df = pd.DataFrame(data)
```

```
print(df)
```

**Και σαν έξοδο παίρνουμε:**

```
>
  Duration  Pulse  Maxpulse  Calories
0        60    110      130      409
1        60    117      145      479
2        60    103      135      340
3        45    109      175      282
4        45    117      148      406
5        60    102      127      300
```

## 38.1.11 Ανάλυση των Dataframes

### Προβολή των Δεδομένων

Μία από τις πιο χρησιμοποιούμενες μεθόδους για γρήγορη επισκόπηση του DataFrame είναι η μέθοδος `head()`.

Η μέθοδος `head()` επιστρέφει τις κεφαλίδες και έναν καθορισμένο αριθμό σειρών, ξεκινώντας από την κορυφή.

Μπορούμε για παράδειγμα να έχουμε για γρήγορη επισκόπηση ενός DataFrame εκτυπώνοντας τις πρώτες 10 σειρές.



```
import pandas as pd

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

print(df.head(10))
```

Η έξοδος είναι:

	Duration	Pulse	Maxpulse	Calories
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0
3	45	109	175	282.4
4	45	117	148	406.0
5	60	102	127	300.5
6	60	110	136	374.0
7	45	104	134	253.3
8	30	109	133	195.1
9	60	98	124	269.0

Υπάρχει επίσης η μέθοδος `tail()` για την προβολή των *τελευταίων* σειρών του DataFrame. Επίσης, επιστρέφει και τις κεφαλίδες και ο αριθμός σειρών, είναι όπως και παραπάνω συγκεκριμένος.

Παράδειγμα:

```
print(df.tail())
```

	Duration	Pulse	Maxpulse	Calories
164	60	105	140	290.8
165	60	110	145	300.4
166	60	115	145	310.2
167	75	120	150	320.4
168	75	125	150	330.4

## 38.1.12 Πληροφορίες για τα δεδομένα

Το αντικείμενο DataFrame έχει μια μέθοδο που ονομάζεται `info()`, η οποία μας δίνει περισσότερες πληροφορίες σχετικά με το σύνολο των δεδομένων.

Ας πάρουμε μια εκτύπωση της εντολής:

```
print(df.info())
```

```
>>> print(df.info())
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 169 entries, 0 to 168
Data columns (total 4 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Duration    169 non-null    int64
1   Pulse       169 non-null    int64
2   Maxpulse    169 non-null    int64
3   Calories    164 non-null    float64
dtypes: float64(1), int64(3)
memory usage: 5.4 KB
None
>>>
```

Από την παραπάνω εκτύπωση, μαθαίνουμε ότι στο έγγραφο, υπάρχουν 169 καταχωρήσεις και 4 στήλες δεδομένων.

Η `info()` μέθοδος μας λέει επίσης πόσες Non-Null τιμές υπάρχουν σε κάθε στήλη και στο σύνολο δεδομένων μας φαίνεται ότι υπάρχουν 164 από 169 Non-Null τιμές στη στήλη "Calories".

Που σημαίνει ότι υπάρχουν 5 σειρές χωρίς καμία απολύτως αξία, στη στήλη «Calories», για κάποιο λόγο.

Οι κενές ή οι μηδενικές τιμές μπορεί να δημιουργήσουν θέμα κατά την ανάλυση δεδομένων και θα πρέπει να εξετάσουμε το ενδεχόμενο κατάργησης των σειρών με κενές τιμές.

Αυτό είναι ένα βήμα προς αυτό που ονομάσαμε προηγουμένως καθαρισμό δεδομένων. Ας δούμε

περισσότερα γι' αυτό .

### 38.1.13 Καθαρισμός δεδομένων

Καθαρισμός δεδομένων σημαίνει να διορθώνουμε άχρηστα δεδομένα στο data set μας.

- Άχρηστα ή και επιζήμια δεδομένα μπορεί να είναι:
- Κενά κελιά
- Δεδομένα σε λάθος μορφή
- Λανθασμένα δεδομένα
- Διπλά δεδομένα

Ας δούμε πώς τα διαχειριζόμαστε όλα αυτά. Κατεβάζουμε ένα νέο αρχείο, το [data2.csv](#)

Αν ανοίξουμε το αρχείο, θα δούμε ότι

```

>>> import pandas as pd
>>> df = pd.read_csv("C:\\Users\\NK\\Desktop\\data2.csv")
>>> print(df)

```

	Duration		Date	Pulse	Maxpulse	Calories
0	0	60	'2020/12/01'	110	130	...
1	1	60	'2020/12/02'	117	145	...
2	2	60	'2020/12/03'	103	135	...
3	3	45	'2020/12/04'	109	175	...
4	4	45	'2020/12/05'	117	148	...
5	5	60	'2020/12/06'	102	127	...
6	6	60	'2020/12/07'	110	136	...
7	7	450	'2020/12/08'	104	134	...
8	8	30	'2020/12/09'	109	133	...
9	9	60	'2020/12/10'	98	124	...
10	10	60	'2020/12/11'	103	147	...
11	11	60	'2020/12/12'	100	120	...
12	12	60	'2020/12/12'	100	120	...
13	13	60	'2020/12/13'	106	128	...
14	14	60	'2020/12/14'	104	132	...
15	15	60	'2020/12/15'	98	123	...
16	16	60	'2020/12/16'	98	120	...
17	17	60	'2020/12/17'	100	120	...
18	18	45	'2020/12/18'	90	112	...
19	19	60	'2020/12/19'	103	123	...
20	20	45	'2020/12/20'	97	125	...
21	21	60	'2020/12/21'	108	131	...
22	22	45	NaN	100	119	...
23	23	60	'2020/12/23'	130	101	...
24	24	45	'2020/12/24'	105	132	...
25	25	60	'2020/12/25'	102	126	...
26	26	60	2020/12/26	100	120	...
27	27	60	'2020/12/27'	92	118	...
28	28	60	'2020/12/28'	103	132	...
29	29	60	'2020/12/29'	100	132	...
30	30	60	'2020/12/30'	102	129	...
31	31	60	'2020/12/31'	92	115	...

```

>>>

```

- Το data set περιέχει ένα κενό κελί ("Date" στη σειρά 22).
- Λάθος μορφή/τύπο δεδομένων ("Date" στη σειρά 26).
- Λάθος δεδομένα ("Duration" στη σειρά 7).
- Διπλότυπα (σειρές 11 και 12).

## 38.1.14 Καθαρισμός διπλών γραμμών και απαλοιφή σειρών

Καθότι όπως είπαμε, τα κενά κελιά μπορεί να μας δώσουν λανθασμένα αποτελέσματα στις αναλύσεις μας, ένας τρόπος να τα αφαιρέσουμε είναι να αφαιρέσουμε τις γραμμές που περιέχουν τα κενά κελιά.

Επειδή συνήθως διαχειριζόμαστε πολλά δεδομένα (big data), η αφαίρεση μερικών γραμμών δεν θα επηρεάσει το αποτέλεσμα μας.

Ας δούμε πως επιστρέφουμε ένα DataFrame χωρίς κενά κελιά

```
import pandas as pd

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

new_df = df.dropna()

print(new_df.to_string())
```

ΣΗΜΕΙΩΣΗ: Από προεπιλογή, η μέθοδος dropna() επιστρέφει ένα νέο DataFrame και δεν θα αλλάξει το πρωτότυπο

```

>>> import pandas as pd
>>> df = pd.read_csv("C:\\Users\\NK\\Desktop\\data.csv")
>>> new_df = df.dropna()
>>> print(new_df.to_string())

```

	Duration	Pulse	Maxpulse	Calories
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0
3	45	109	175	282.4
4	45	117	148	406.0
5	60	102	127	300.5
6	60	110	136	374.0
7	45	104	134	253.3
8	30	109	133	195.1
9	60	98	124	269.0
10	60	103	147	329.3
11	60	100	120	250.7
12	60	106	128	345.3
13	60	104	132	379.3
14	60	98	123	275.0
15	60	98	120	215.2
16	60	100	120	300.0
18	60	103	123	323.0
19	45	97	125	243.0
20	60	108	131	364.2
21	45	100	119	282.0
22	60	130	101	300.0
23	45	105	132	246.0
24	60	102	126	334.5
25	60	100	120	250.0
26	60	92	118	241.0
28	60	100	132	280.0
29	60	102	129	380.3
30	60	92	115	243.0
31	45	90	112	180.1
32	60	101	124	299.0
33	60	92	112	243.0

Για να αλλάξουμε το αρχικό DataFrame, χρησιμοποιούμε το όρισμα `inplace = True`:

```

import pandas as pd

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

df.dropna(inplace = True)

print(df.to_string())

```

## 38.1.15 Αντικατάσταση κενών τιμών

Ένας άλλος τρόπος αντιμετώπισης των κενών κελιών είναι να εισάγουμε στη θέση τους μια νέα τιμή.

Με αυτόν τον τρόπο δεν χρειάζεται να διαγράψουμε ολόκληρες σειρές μόνο και μόνο λόγω κάποιων κενών κελιών.

Η μέθοδος `fillna()` μας επιτρέπει να αντικαταστήσουμε τα κενά κελιά με μια τιμή. Στην προκειμένη περίπτωση, αντικαθιστούμε τα κενά κελιά με την τιμή 130

```
>>> import pandas as pd
>>> df = pd.read_csv("C:\\Users\\NK\\Desktop\\data.csv")
>>> df.fillna(130)

   Duration  Pulse  Maxpulse  Calories
0         60    110      130     409.1
1         60    117      145     479.0
2         60    103      135     340.0
3         45    109      175     282.4
4         45    117      148     406.0
..      ...    ...      ...      ...
164        60    105      140     290.8
165        60    110      145     300.4
166        60    115      145     310.2
167        75    120      150     320.4
168        75    125      150     330.4

[169 rows x 4 columns]
>>>
```

Μπορούμε να αντικαταστήσουμε κενά κελιά, μόνο σε συγκεκριμένες στήλες, καθορίζοντας τις στήλες. Στο παρακάτω παράδειγμα, αντικαθιστούμε τις κενές τιμές στη στήλη "Calories" με τον αριθμό 130.

```
import pandas as pd

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

df["Calories"].fillna(130, inplace = True)
```

Ζητώντας και μια εκτύπωση: `print(df)`

```
>>> df["Calories"].fillna(130, inplace = True)
>>> print(df)
   Duration  Pulse  Maxpulse  Calories
0         60    110       130     409.1
1         60    117       145     479.0
2         60    103       135     340.0
3         45    109       175     282.4
4         45    117       148     406.0
..      ...    ...      ...      ...
164        60    105       140     290.8
165        60    110       145     300.4
166        60    115       145     310.2
167        75    120       150     320.4
168        75    125       150     330.4

[169 rows x 4 columns]
>>>
```

## 38.1.16 Αντικατάσταση τιμών με `mean`, `median` και `mode`

Αποτελεί συνηθισμένο τρόπο η αντικατάσταση κενών κελιών με τον υπολογισμό της μέσης τιμής (`mean` = μέσος όρος), της διάμεσης τιμής (`median` = ο μέσος όρος των μοναδικών τιμών) ή της πιο συχνά εμφανιζόμενης τιμής (`mode`) της στήλης.

Έτσι έχουμε τις μεθόδους `mean()`, `median()` και `mode()` για τον υπολογισμό και την αντικατάσταση των κενών τιμών.

### Παράδειγμα 1

Υπολογισμός και αντικατάσταση κενών τιμών με τη `mean()`



```
import pandas as pd

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

x = df["Calories"].mean()

df["Calories"].fillna(x, inplace = True)
print(df)
```

```
>>> import pandas as pd
>>> df = pd.read_csv("C:\\\\Users\\NK\\Desktop\\data.csv")
>>> x = df["Calories"].mean()
>>> df["Calories"].fillna(x, inplace = True)
>>> print(df)
   Duration  Pulse  Maxpulse  Calories
0         60    110      130     409.1
1         60    117      145     479.0
2         60    103      135     340.0
3         45    109      175     282.4
4         45    117      148     406.0
..      ...    ...      ...      ...
164        60    105      140     290.8
165        60    110      145     300.4
166        60    115      145     310.2
167        75    120      150     320.4
168        75    125      150     330.4

[169 rows x 4 columns]
>>>
```

## Παράδειγμα 2

Υπολογισμός και αντικατάσταση κενών τιμών με τη median()

```
import pandas as pd

df =

pd.read_csv('C:\\Users\\NK\\Desktop\\data.csv')

x = df["Calories"].median()

df["Calories"].fillna(x, inplace = True)

print(df)
```

```
>>>
==== RESTART: C:/Users/NK/AppData/Local/Programs/Python/Python38-32/Python.exe
      Duration  Pulse  Maxpulse  Calories
0           60    110        130     409.1
1           60    117        145     479.0
2           60    103        135     340.0
3           45    109        175     282.4
4           45    117        148     406.0
..          ...    ...        ...      ...
164          60    105        140     290.8
165          60    110        145     300.4
166          60    115        145     310.2
167          75    120        150     320.4
168          75    125        150     330.4

[169 rows x 4 columns]
>>>
```

### Παράδειγμα 3

Υπολογισμός και αντικατάσταση κενών τιμών με τη mode()

```
import pandas as pd

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

x = df["Calories"].mode()[0]

df["Calories"].fillna(x, inplace = True)

print(df)
```

25

```
>>>
==== RESTART: C:/Users/NK/AppData/Local/Programs/Pyt
      Duration  Pulse  Maxpulse  Calories
0           60    110      130     409.1
1           60    117      145     479.0
2           60    103      135     340.0
3           45    109      175     282.4
4           45    117      148     406.0
..          ...    ...      ...      ...
164          60    105      140     290.8
165          60    110      145     300.4
166          60    115      145     310.2
167          75    120      150     320.4
168          75    125      150     330.4

[169 rows x 4 columns]
>>>
```

## 38.2.1 Ασκήσεις

### Άσκηση 1

Τυπώστε μια σειρά ημερομηνιών από την 1η Ιανουαρίου 2024 μέχρι και την 11η Φεβρουαρίου 2024 χρησιμοποιώντας τη βιβλιοθήκη pandas.

### 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.

### 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].