Python

ΣΗΜΕΙΩΣΕΙΣ ΜΑΘΗΜΑΤΟΣ ΠΛΣ60

2022

Blocks

Τα blocks δημιουργούνται με στοιχισμένες εσοχές. Δεν υπάρχουν άνω τελείες.

Σχόλια

comment

“”” για multiline comments

Variables

Όπως στην JavaScript (αλλά χωρίς var, let, const). Καταλαβαίνει τον τύπο από την τιμή. Μπορούμε να αρχικοποιήσουμε συγκεκριμένο τύπο με constructors (π.χ. int())

```
x = str(3) then x = “3”
y = int(3) then y = 3
z = float(3) then z = 3.0
```

Assign multiple variables in one row

```
x, y, z = “a”, 3’ 4.7
```

Unpacking

```
L = [“a”, 1, 4.7]
x, y, z = L
```

που δίνει

```
x = “a”
y = 1
z = 4.7
```

Όλες οι μεταβλητές είναι αντικείμενα (objects).

Global/Local

Τοπικές μεταβλητές ορίζονται μόνον εντός function blocks. Οποιαδήποτε μεταβλητή ορισμένη έξω από μία συνάρτηση είναι global.

Το global keyword χρησιμοποιείται για να ορίσει ως global ή να αλλάξει μια ήδη ορισμένη global μεταβλητή.

Data types

IMMUTABLE

```
numeric: int, float, complex
str
bool
tuple
bytes
frozenset
range
None
```

MUTABLE

```
list
set
dict
bytearray
memoryview
```

None

as null.

```
type(None) = Nonetype
```

Comparison allowed

```
if x is None
if x == None
```

Strings

Multiline strings with “””

```
a = """ blabla
     blabla """
```

strings are arrays.

```
"s" in mystring
"s" not in mystring
```

zero-based

ισχύει και αρίθμηση από το τέλος αρχίζοντας το μέτρημα με -1

	0	1	2	
s = “	a	b	c	“
	-3	-2	-1	

STRING METHODS

```
s.upper()
s.lower()
```

`s.isalpha()` ελέγχει αν είναι αλφαβητικό

`s.strip()` αφαιρεί κενά από την αρχή και το τέλος

`s.strip('\n')` αφαιρεί \n από την αρχή και το τέλος

```
s.replace(x,y)
```

`s.split(",")` χωρίζει χρησιμοποιώνιας τον χαρακτήρα “,”

`"-".join(iterable)` παίρνει ένα ένα τα στοιχεία του iterable και δημιουργεί ένα string ενώνοντας τα στοιχεία με την παύλα -. Αντί για παύλα, μπορεί να μπει οποιοσδήποτε χαρακτήρας.

FORMAT

```
txt = "some text with {0} and {1}"
txt.format(v0, v1)
```

placeholders can have numbers or not. Also,

```
txt = f"some text with {v0} and {v1}"
```

Special placeholders

`{1: .2f} δίνει 20.00 για τιμή 20`

```
myTuple = (vo, v1)
txt.format(*myTuple)
```

με το * η Python αναπτύσσει τα στοιχεία του tuple σε ορίσματα.

Slicing

```
s = "abcdefg"
s[a:b:c]
```

επιστρέφει το string με στοιχεία από τον δείκτη a μέχρι και τον δείκτη b-1 με βήμα c

Αν οι δείκτες είναι θετικοί, τότε μετρούν με 0-based. Αν οι δείκτες είναι αρνητικοί, τότε μετρούν με -1based.

Αν ο δείκτης είναι εκτός ορίου, αναφέρεται στην κενή θέση. Παράδειγμα,

```
s[0:7]
```

επιστρέφει “abcdefg”

```
s[:b]
```

επιστρέφει το string με στοιχεία από την αρχή μέχρι και τον δείκτη b-1

```
s[a:]
```

επιστρέφει το string με στοιχεία από τον δείκτη a μέχρι και το τέλος

```
s[::2]
```

επιστρέφει το string με στοιχεία από την αρχή ως το και τέλος με βήμα 2

```
s[::-1]
```

επιστρέφει το αντίστροφο του αρχικού string

```
s[-5:-2]
```

επιστρέφει το string με στοιχεία από τον δείκτη -5 μέχρι τον δείκτη -3

```
"abcde"[1:-1]
```

επιστρέφει “bcd”, διότι το 1 μετράει από αριστερά με 0-based και το -1 μετράει από δεξιά με -1based και είναι exclusive.

```
"abcde"[1:3:-1]
```

επιστρέφει [ ] διότι ξεκινάει από τον δείκτη 1 και αντί να κινηθεί προς τα δεξιά όπου βρίσκεται το 3, κινείται προς τα αριστερά εξαιτίας του αρνητικού βήματος. Συνεπώς, αποκλίνει.

```
"abcde"[-3:-1:-1]
```

επιστρέφει [ ] διότι ξεκινάει από τον δείκτη -3 και αντί να κινηθεί προς τα δεξιά όπου βρίσκεται το -1, κινείται προς τα αριστερά εξαιτίας του αρνητικού βήματος. Συνεπώς, αποκλίνει.

Operators

// floor division παράδειγμα: 15//2=7

```
x is y
```

συγκρίνει το x με το y σε επίπεδο αντικειμένου (θέσης μνήμης) και όχι μόνον τιμής

```
x is not y
in
not in
and
or
not
```

```
mylist += [6]
```

προσθέτει το στοιχείο 6 στην mylist σαν να κάνουμε append. Το αντικείμενο mylist παραμένει το ίδιο.

```
mylist = mylist + [6]
```

δημιουργεί νέο αντικείμενο mylistList

List

Lists are **dynamic arrays**. **Ordered**.

```
[x1, x2, x3, … , xn]
```

x can be of different type

```
mylist = list(x1, x2, x3)
```

where list() is the constructor.

Ισχύει ο μηχανισμός τεμαχισμού για τις λίστες.

METHODS

```
insert(position, element)
append(element)
extend(list)
remove(element)
pop(position)
```

`pop()` αφαιρεί το τελευταίο

```
del mylist[0] or del mylist
clear()
sort()
sort(reverse=True)
```

Στην insert, αν το position είναι πέραν του τέλους της λίστας, τότε προσθέτει στο τέλος της λίστας.

LIST COMPREHENSION

```
newlist = [expression for item in oldlist if condition]
```

Το μήκος της λίστας υπολογίζεται με την len

```
len(myList)
```

map

Αν έχουμε μία συνάρτηση f(x) και μία λίστα *mylist* και θέλουμε μία νέα λίστα $L=\left[f\left(mylist_i\right)\right]$, τότε υπάρχει η συνάρτηση *map* που κάνει το εξής:

```
L = list(map(f, mylist))
```

filter

Η συνάρτηση filter φιλτράρει τα στοιχεία μίας λίστας επιστρέφοντας μία νέα λίστα με μόνον εκείνα τα στοιχεία που πληρούν μία συνθήκη. Παράδειγμα:

```
seq = [1,2,3,4,5,6]
list(filter(lambda num: num % 2 == 0, seq))
```

```
[2, 4, 6]
```

όπου μέσα στην filter υπάρχει μια ανώνυμη συνάρτηση (lambda expression) που ελέγχει κάθε στοιχείο της λίστας seq και επιστρέφει True αν είναι άρτιος αριθμός. Στην συνέχεια, η filter λαμβάνει μόνον αυτά τα στοιχεία της λίστας seq δημιουργώντας μία νέα λίστα.

Tuple

Ordered.

```
(x1, x2, x3, …, xn)
```

tuple constructor

```
mytuple = tuple(x1, x2, x3)
```

tuple με ένα στοιχείο μόνον

```
y = (x1,)
```

ADDITION

x, y tuples, τότε x + y είναι επίσης tuple

```
z = x * 2
```

δίνει την πλειάδα z με τα διπλάσια στοιχεία της x.

METHODS

```
del mytuple
```

Unpacking

Σε μια λίστα από πλειάδες, αν θέλω μόνον τα δεύτερα στοιχεία κάθε πλειάδας, τότε κάνω unpacking:

```
x = [(1,2), (3,4), (5,6)]
```

```
for a,b in x:
    print(b)
```

```
2
4
6
```

Dictionary

Ordered.

```
mydict = {“key1”: value1, “key2”: value2}
```

Values can be of any type.

Any immutable object can be a key. Κλειδιά διαφορετικών τύπων μπορεί να υπάρχουν στο ίδιο λεξικό.

No duplicates (i.e. with the same key) allowed.

Αυτό σημαίνει ότι ο compiler δέχεται διπλότυπα με βάση το κλειδί, αλλά κρατάει μόνον το τελευταίο.

```
mydict[“key1”]
```

accesses the element with key = “key1”

METHODS

```
mydict[“key3”] = value3
```

προσθέτει ένα νέο στοιχείο. Το ίδιο κάνει και η

```
mydict.update({“key3”: value3})
```

```
get(key, value)
```

επιστρέφει την τιμή με κλειδί key. Αν το κλειδί key δεν υπάρχει, τότε επιστρέφει την τιμή value.

Set

Not ordered.

No duplicates.

Only immutable elements

```
myset = {‘x1’, 5, True}
myset = set((‘x1’, 5, True))
```

METHODS

```
myset.add(element)
myset.update(other set, tuple, dictionary)
Myset.union(anotherset)
myset.remove(‘x1’)
myset.discard
myset.pop()   don’t know which element
myset.clear()
del myset
```

if-else

```
if condition:
        statement
elif condition:
        statement
else:
        statement
```

TERNARY

```
x = value1 if condition else value2
```

while

```
while condition:
        statement
else:
        statement
```

else is optional and its code runs only when while don't run. **Else code does not run if code exits with break.**

break

continue

For

```
for x in iterable:
        statement
else:
        statement
```

else is optional and its code runs only when for finishes the iterations. **Else code does not run if code exits with break.**

Lambda

```
x = lambda a: a+10
```

is the same as x(a) = a + 10

Build-in functions

`chr()`	Unicode No → character
`ord()`	Character → Unicode No
`range(5)`	0 ως 4
`range(2,5)`	2,3,4
`range(2,9,3)`	2 ως 8 με βήμα 3
`isinstance(x, int)`	checks if x is int or not
`id(object)`	Επιστρέφει ένα μοναδικό αναγνωριστικό του αντικειμένου object, δηλ. την διεύθυνσή του στην μνήμη.
`min(), max()`	
`abs(), pow()`	
`type(obj)`	

Functions

```
def functionName(args):
        implementation
```

ARGUMENTS

- καθόλου
- μία ή περισσότερες μεταβλητές
- *args για tuple
- **args για dictionary
- x = a1 όπου a1 είναι default value
- λίστα

ALTERNATIVE CALL

```
def somef(a, b):
        implementation
somef(b = 30, a = 40)
```

RETURNS

Επιστρέφει None ή μία ή περισσότερες τιμές

PASS DATA BY

immutable arguments → μεταφορά με τιμή

mutable arguments → μεταφορά με αναφορά

ΠΡΟΣΟΧΗ

```
somefunction(*args, *args)
```

not allowed

DOCSTRINGS

Πρόκειται για κείμενο που γράφεται στην 1η γραμμή του implementation (και τελειώνει με τελεία), εκτείνεται σε περισσότερες (με την 2η πάντα κενή) και δίνει πληροφορίες για την συνάρτηση.

```
def myfunction(x, y):
        """ Calculates the sum of two values.

        The values must be integers or float.
        """
        implementation


help(myfunction)
```

Όταν κληθεί η help, θα επιστρέψει το παραπάνω επεξηγηματικό κείμενο.

Class

```
class myClass:
	z = value1
	# Constructor
	def __init__(self, v1, v2):
		self.f1 = v1
		self.f2 = v2
	# Destructor
	def __del__(self):
		some code
	# Checks equality
	def __eq__(self, other):
		some code
	def __str__(self):
		implementation
	del myMethod(self, x1):
		implementation
```

f1, f2: instance variables

z: class variable (static)

self is like this in Java and C#

CLASS VARIABLES – INSTANCE VARIABLES

Οι μεταβλητές αντικειμένου ορίζονται μόνον μέσα στις μεθόδους της κλάσης. Ο constructor και ο destructor είναι μέθοδοι.

Οι μεταβλητές κλάσης είναι static. Καλούνται με

- το όνομα της κλάσης ή
- με το self.__class__ (εντός της κλάσης) ή
- με το objectname.__class__ (εκτός της κλάσης), ή
- self.όνομαμεταβλητής, αν δεν υπάρχει μεταβλητή στιγμιοτύπου με το ίδιο όνομα.

Προσοχή! Οι class variables κληρονομούνται. Όμως, αν η class variable της παραγόμενης κλάσης αλλάξει τιμή, δεν επηρεάζει την τιμή της αντίστοιχης class variable της μητρικής κλάσης.

Οι class variables και οι instance variables μπορούν να έχουν το ίδιο όνομα.

```
class A:
    val = 1
class B(A):
    pass
class C(A):
    pass
print(A.val, B.val, C.val)
B.val = 2
print(A.val, B.val, C.val)
A.val = 3
print(A.val, B.val, C.val)
```

Αποτέλεσμα

```
1 1 1
1 2 1
3 2 3
```

Όταν υπάρχει pass, τότε η κλάση κληρονομεί τα πάντα από την μητρική. Αν αλλάξει η τιμή της στατικής της μητρικής, αλλάζει και στην παραγόμενη. Μόλις, όμως ορίσουμε τιμή στην στατική της παραγόμενης, τότε καταργείται η επίδραση του pass και η παραγόμενη έχει στατική με την τιμή που της ορίστηκε.

```
del obj1.v1
```

διαγράφει το field v1

```
del obj1
```

διαγράφει το obj1

PUBLIC – PRIVATE

Όλες οι μέθοδοι είναι public. Για να τις ορίσουμε ως private, το όνομά τους πρέπει να ξεκινά με διπλή κάτω παύλα.

STATIC METHODS

```
@staticmethod
def myStaticMethod( ):
	implementation
```

no self needed.

ΕΝΣΩΜΑΤΩΜΕΝΕΣ ΜΕΤΑΒΛΗΤΕΣ

```
a = myClass(v1, v2)
a.__dict__
```

εμφανίζει τις instance variables του a

```
myClass.__dict__
```

εμφανίζει μεταβλητές και μεθόδους της κλάσης

Inheritance

```
class Parent:
        def __init__(self, v1, v2):
                self.f1 = v1
                self.f2 = v2
        def myMethod(self):
                implementation


class Child(Parent):
        def __init__(self, v1, v2, v3):
                super().__init__(v1, v2)
                Parent.__init__(self, v1, v2)
                self.f3 = v3
```

Προσοχή στο κόκκινο σημείο: δεν θέλει self διότι η super() καλεί το αντικείμενο της SuperClass.

Η Python υποστηρίζει πολλαπλή κληρονομικότητα.

Η Python υποστηρίζει methods overriding.

Η Python δεν υποστηρίζει method overloading.

Η Python δεν υποστηρίζει method overloading. Αν ορίσουμε δύο μεθόδους με το ίδιο όνομα και διαφορετική υπογραφή, ο compiler δεν το βγάζει λάθος, αλλά στην κλήση της μεθόδου καλεί μόνον την τελευταία έκδοσή της.

Input

```
variable1 = input("message to the user")
```

Iterators

Αντικείμενα που είναι αριθμήσιμα μπορούν να δημιουργήσουμε έναν iterator.

Παράδειγμα

```
mytuple = ("one", 2, 3.0)
myIterator = iter(mytuple)
next(myIterator)
```

η next δίνει το επόμενο στοιχείο.

Modules

Κώδικας αποθηκευμένος σε αρχείο κειμένου με κατάληξη py.

```
import myModule
import myModule as m
from myModule import myFunction
dir(MyModule)
```

δίνει τα περιεχόμενα του MyModule

Χρήσιμα modules

- datetime
- math
- json
- re (regular expression)
- random

```
import random
x = random.randomrange(1,10)
```

```
from datetime import datetime
s="28/10/1970"
obj = datetime.strptime(s, '%d/%m/%Y')
```

JSON

```
import json
jsonDestinationObj = open(file, 'w')
json.dump(rawdata, jsonDestinationObj)
jsonStr = json.dumps(rawdata)
```

```
mydict = json.load(jsonFile)
mydict = json.loads(jsonString)
```

Η dump μετατρέπει το rawdata σε json και το αποθηκεύει στο jsonDestinationObj.

Η dumps μετατρέπει το rawdata σε json και το αποθηκεύει σε μορφή string. Ειδικά για την dumps έχουμε και τα εξής keywords:

- indent = 3 τρία κενά για κάθε εσοχή
- separators(".", "=") που αλλάζει separator
- sort_keys=True που ταξινομεί με βάση το κλειδί

Η load φορτώνει από αρχείο json.

Η loads φορτώνει από string json.

Exception

```
try:
        #some code
except [ExceptionName]:
        #some code
else:
        #code to run if no exception risen
finally:
        #code to run no matter what
```

Ρίχνουμε μία εξαίρεση ως εξής:

```
raise ExceptionName(message)
```

Files

```
1       f = open(filepath, mode, encoding='utf-8')
2       f.read()
3       f.read(5)
4       f.readline()
5       f.seek(0)
6       for x in f:
                print(x)
7       f.write(something)
8       f.close()
```

mode:

- a – for appending
- r – for reading
- w – for writing
- x – create
- b - binary

Η 2 διαβάζει όλο το αρχείο.

Η 3 διαβάζει μόνον τους πρώτους 5 χαρακτήρες

Η 4 διαβάζει μία γραμμή. Αν κληθεί ξανά, διαβάζει την επόμενη γραμμή.

Η 5 μας μεταφέρει στην αρχή του αρχείου

Αν δεν δηλωθεί keyword, τότε το αρχείο ανοίγεται για ανάγνωση.

WITH

Η with είναι η using της C#

```
with open(filepath, 'r') as f:
        f.read( )
```

```
import os
os.path.exists(filepath | directory)
os.rmdir(directory)
os.remove(pathfile)
os.listdir(directory)
os.path.join(path1, path2)
os.path.isdir(path)
os.rename(oldname, newname)
os.path.abspath(file)

import shutil
shutil.copy(source, destination)
shutil.move(source, destination)
```

MS SQL

```
import pyodbc

cnx = pyodbc.connect(connectionstring)
c = cnx.cursor()
count = c.execute(sqlQuery).rowcount
c.executemany(sqlQuery)

row = c.fetchone()
while row:
        #do something
        row = c.fetchone()

rows = c.execute(sqlQuery).fetchall()

for row in c.execute("select a, b from R"):
        print(row.a, row.b)

# insert query
count = c.execute(sqlQuery).rowcount
cnx.commit()
```

Όταν ο cursor καλεί τις γραμμές μία μία, τότε εγκλωβίζεται στην σύνδεση αυτή και δεν μπορεί να καλέσει κάποια άλλη σύνδεση για εκτέλεση άλλου query. Σε αυτή την περίπτωση, πρέπει να δημιουργήσουμε και δεύτερο cursor, ή να πάρουμε όλες μαζί τις rows σε ένα αντικείμενο.

Παράδειγμα παραμετροποιημένης εντολής SQL:

```
params = (value1, value2, value3)
rowsAffected = cursor.execute("SELECT * FROM TABLE1
WHERE X = ? AND y = ?", params).rowcount
```

Py Installer

Δημιουργεί **εκτελέσιμα αρχεία Python**

First install the package pyinstaller by typing:

```
pip install pyinstaller
```

Then, add the directory of installation into the PATH. The directory is mentioned in the Warning section after the installation.

Then, at the folder where the py file is, execute the following command:

```
pyinstaller --onefile myfile.py
```

Search in the folder dist for the executable file.

Run the executable file by typing

```
./myfile
```

Or this

```
myfile
```

Pyodbc

Διαδικασία εγκατάστασης για Windows:

- Εγκατάσταση πακέτου από Microsoft
- pip install πακέτο.whl

Διαδικασία εγκατάστασης οδηγού Microsoft ODBC για Linux:

https://learn.microsoft.com/en-us/sql/connect/odbc/linux-mac/installing-the-microsoft-odbc-driver-for-sql-server?view=sql-server-ver16&tabs=debian18-install%2Csuse17-install%2Cdebian8-install%2Credhat7-13-install%2Crhel7-offline

1. Select Debian
2. Run the first two commands
3. Then, run the command for Debian 11
4. Run all the remaining commands

It seems that unixodbc needs the version 17 of ODBC driver, so install this driver (from the same website) too by just typing

```
sudo apt-get install -y msodbcsql17
```

Then, install pyodbc using pip.

Pymssql

Πρόκειται για πακέτο που **δεν χρειάζεται** τον οδηγό ODBC της Microsoft για να λειτουργήσει. Η εγκατάσταση του πακέτου γίνεται ως εξής:

```
pip install pymssql
```

Στο ακόλουθο παράδειγμα φαίνεται η εφαρμογή των μεθόδων του πακέτου:

```
conn = pymssql.connect(server=myserver,
user=myusername, password=mypassword,"mydb")

cursor = conn.cursor()

cursor.execute('SELECT * FROM persons WHERE salesrep=
%s', 'John Doe')

for row in cursor:

    print("ID=%d, Name=%s" % (row['id'],
row['name']))

conn.close()
```

Αν χρειαζόμαστε να καλέσουμε την fetchall ή την rowcount, τότε αυτές καλούνται ως μέθοδοι του cursor. Δηλαδή,

```
ourdata = cursor.fetchall()

rowsNumReturned = cursor.rowcount()
```

Install other Python version in parallel

1. Download Python tar package and unzip it
2. In a terminal opened at the unzipped folder run the following commands:
   make
   make test
   sudo make altinstall
3. Test by typing Python3.11 -- version

HTTP Request

A HTTP client sends a HTTP request to the server containing:

- URL
- Request method
- Headers
- Body

Request methods:

- GET (ανάκτηση δεδομένων)
- POST (αποστολή δεδομένων)
- PUT (ενημέρωση δεδομένων)
- DELETE (διαγραφή δεδομένων)

Για requests η Python έχει 2 βιβλιοθήκες:

- urllib.request
- requests

```
import urllib.request
response = urllib.request.urlopen(url, data)
respStr = response.read().decode('utf-8')
result = json.loads(respStr)
```

```
import request
response = request.get(url)
response = request.post(url, data = {'key': value})
response = request.put(url, data = {'key': value})
response = request.delete(url)
```

Pandas

```
pip install pandas
import pandas as pd
```

Series

This is like a column in a table

```
a=[1,7,2]
myvar = pd.Series(a)
print(myvar)
```

gives

```
0  1
1  7
2  2
```

Define an index for a Series element

```
a=[1,7,2]
myvar = pd.Series(a, index=["x", "y", "z"])
print(myvar)
print(myvar["y"])
```

```
x    1
y    7
z    2
dtype: int64
7
```

A Series can be derived from a dictionary

```
calories = {"day1":420, "day2":380, "day3": 390}
v = pd.Series(calories)
print(v)
```

```
day1    420
day2    380
day3    390
dtype: int64
```

DataFrames

DataFrames are 2-dimensional arrays

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

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

```
   calories  duration
0       420        50
1       380        40
2       390        45
```

Locate a row with **loc** method like this:

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

v = pd.DataFrame(data)
print(v.loc[0])
```

```
calories    420
duration     50
Name: 0, dtype: int64
```

Like Series, one can have **custom indexes** in DataFrames instead of 0,1,2,…

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

v = pd.DataFrame(data, index=["day1", "day2", "day3"])
print(v.loc[["day1","day2"]])
```

```
      calories  duration
day1       420        50
day2       380        40
```

To create a DataFrame from a Python dictionary, use the DataFrame constructor:

`pd.DataFrame(dictDataObj)`

To create a DataFrame from a list of tuples, use this:

`pd.DataFrame.from_records(listoftuples)`

CSV Files

Working with csv files having custom separators:

```
f = pd.read_csv('shippingdata.csv', sep=';')
print(f)
```

```
   Ordername finishingDate        ETD        ETA   BookingNo
0  D2022-048     12/8/2022  28/8/2022  4/10/2022  SEL1425498
1  D2022-049     12/8/2022  26/8/2022  2/10/2022  SEL1428256
2  D2022-050     12/8/2022  26/8/2022  2/10/2022  SEL1428256
3  D2022-051     12/8/2022  26/8/2022  2/10/2022  SEL1428256
4  D2022-052     12/8/2022  26/8/2022  2/10/2022  SEL1428256
5  D2022-053     12/8/2022  26/8/2022  2/10/2022  SEL1428256
```

The returned object is a DataFrame.

To save a dataframe to csv, type

`df.to_csv(filename, sep=';', decimal=',')`

where the decimal parameters changes period '.' to comma ',' in values.

Head method

`head(arg)` method returns headers and `n=arg` rows. If arg is omitted, then n=5

Tail method

Like `head(arg)`, the `tail(arg)` returns the last rows. If arg is omitted, then it returns 5 rows.

Info method

```
f = pd.read_csv('shippingdata.csv', sep=';')
print(f.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Ordername      6 non-null      object
 1   finishingDate  6 non-null      object
 2   ETD            6 non-null      object
 3   ETA            6 non-null      object
 4   BookingNo      6 non-null      object
dtypes: object(5)
memory usage: 368.0+ bytes
None
```

String

To return the whole object (e.g., DataFrame), use

`to_string()`

Max Rows

Use this property to set/update/display the maximum number of rows in a display:

```
pd.options.display.max.rows
```

JSON

```
dj=pd.read_json('data.json')
print(dj.to_string())
```

The returned object is a DataFrame.

Drop N/A

The method `dropna()` removes missing values from a DataFrame returning a new DataFrame.

To modify the same DataFrame, use `dropna(inplace=True)`.

Fill N/A

Use the method `fillna(value)` to replace N/A with value.

To modify the same DataFrame, use

`fillna(value, inplace=True)`.

```
import pandas as pd
f1 = pd.read_csv('shippingdata.csv', sep=';')

f1['BookingNo'].fillna('SEL0000000', inplace=True)
print(f1)
```

```
   Ordername finishingDate        ETD        ETA   BookingNo
0  D2022-048     12/8/2022  28/8/2022  4/10/2022  SEL1425498
1  D2022-049     12/8/2022  26/8/2022  2/10/2022  SEL0000000
2  D2022-050     12/8/2022  26/8/2022  2/10/2022  SEL1428256
3  D2022-051     12/8/2022        NaN  2/10/2022  SEL1428256
4  D2022-052     12/8/2022  26/8/2022  2/10/2022  SEL1428256
5  D2022-053     12/8/2022  26/8/2022  2/10/2022  SEL1428256
```

For replacement one can use

- The average (mean) value calculated with the help of the function

  `Dataframe.mean()`

- The value in the middle (after sorting the value). This value is called *the median* and the corresponding function is

  `Dataframe.median()`

- The most frequent value. The function used to calculate it is

  `Dateframe.mode()`

  As this function returns a Series, apply it like this:

  `Dateframe.mode()[0]`

Read from Excel

```
pip install xlrd
```

Then, for the 1st sheet

```
df = pd.read_excel(filename)
```

or for specific sheet (starting from 0)

```
df = pd.read_excel(filename, sheet_name = 1)
```

Write to Excel

This save a dataframe to an Excel file:

```
# Write DataFrame to Excel file with sheet name
df.to_excel('Courses.xlsx', sheet_name='Technologies')
```

where the sheet_name is optional.

To save some columns only, do this:

```
# Save Selected Columns to Excel File
df.to_excel('Courses.xlsx', columns = ['Fee','Duration'])
```

To write to multiple sheets, do this:

```
# Write to Multiple Sheets
with pd.ExcelWriter('Courses.xlsx') as writer:
    df.to_excel(writer, sheet_name='Technologies')
    df2.to_excel(writer, sheet_name='Schedule')
```

To append to an already existed file, do this:

```
# Append DataFrame to existing excel file
with pd.ExcelWriter('Courses.xlsx',mode='a') as writer:
    df.to_excel(writer, sheet_name='Technologies')
```

Venv

Python virtual environment

For **installing** the virtual environment to a Linux machine type this:

```
sudo apt install python3-venv
```

Then, **create** a new virtual environment by typing

```
python3 -m venv testing
```

where testing is the name of the virtual environment.

Inside the folder named “testing” some basic Python modules have been copied.

Activate the virtual environment like this:

```
source testing/bin/activate
```

which calls a script saved in the folder testing/bin

Deactivate the virtual environment by simply typing

```
deactivate
```

JupyterLab inside venv

First, create the virtual environment. For example, testvenv. Then, activate the virtual environment. Install Jupyter inside the virtual environment by typing

```
pip install jupyterlab
```

and create a new kernel inside it by typing

```
pip install ipykernel
```

Being inside the virtual environment, introduce the new kernel to JupyterLab by typing

```
python3 -m ipykernel install --user --name=testvenv
```

Initialize JupyterLab with jupyter-lab command and at the top right side of the screen, select the kernel with name testnenv.