

Python - Crash Course - 3<sup>rd</sup> ed.  
by  
ERIC MATTHES

- No Starch Press

ISBN-13 : 978-1-7185-0270-3 (PRINT)

Pages : 514

## 2. Variables & Simple Data Types

23/02/2024

(2)

variable inside a String  $\Rightarrow$  f-string  $\Rightarrow$   
 $f\{{\text{variable1}}\} \text{ is } \{{\text{variable2}}\}$ "

e.g.:  $f\text{"Name is } \underline{\text{full-name.title()}}\text{"}$   
 $\hookrightarrow$  title case

white space: - in & it allowed inside a string

Removing whitespace:

string.rstrip() - left side - right side  
string.lstrip() - right - left side  
string.strip() - only both the sides.

? How to remove all whitespaces in a string?

string.replace(occurrence, replacement)  
(e.g.) string.replace(" ", "")

Removing prefixes:

string.removeprefix(prefixString)

? How to remove suffix? .removesuffix(suffixString)

? How to remove occurrence anywhere?  
string.replace(occurrence, replacement);

All integer divides will result in float output

underscores allowed in numbers:

e.g.: 14\_00\_000

multiple assignments:

x, y, z = 10, 56, -5

Comments  $\Rightarrow$  #

>>> import this

$\Rightarrow$  Shows python programming principle

26/02/2024

(2)

### 3. Introducing Lists

[element<sub>1</sub>, element<sub>2</sub>, ..., element<sub>n</sub>]

⇒ mixed data type

index position starts at 0

last item index = -1

last but one = -2 and so on

appending element ⇒

list.append(element)

Empty list = []

inserting element ⇒ list.insert(index, element)

removing element ⇒ del element

remove & use element ⇒ list.pop()  
at last index

remove at any position ⇒ list.pop(index)

remove by value ⇒ list.remove(value)

permanent sort ⇒ list.sort() ⇒ alphabetical

~~list.sort()~~ ⇒ reverse alphabetical

list.sort(reverse=True)

temporary sort ⇒ sorted(list)

permanent reverse order ⇒ list.reverse()  
~~printing in~~ (not sorting)

Length of list ⇒ len(list)

27/02/2024

## 4. Working with LISTS

3.5

for loop  $\Rightarrow$

for item in <list>:

<indent><

body.

7

~~for~~ range() =>

for value in range(1,5):

4

5

2

```
mylist = list(range(1,5))
```

5) convert to list.

```
even_numbers = list(range(2, 11, 2))
```

$\bar{T} \rightarrow$  Step Value

Statistics  $\Rightarrow$  min (<list>  
max (<list>  
sum (<list>)

\* List Comprehensions  $\Rightarrow$  (w/o using for loop).

squares = [value \*\* 2 for value in range(1, 11)]

Slicing a List  $\Rightarrow$  myList[0:3]  $\Rightarrow$  element at index 0, 1, 2

`mylist[:3] ⇒ from 0th element`

`mylist[2:]`  $\Rightarrow$  from index-2 to end.

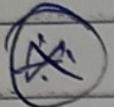
`myList[-3:]` ⇒ last 3 elements.

`mylist[:]` ⇒ entire list

copying a list  $\Rightarrow$  ~~new~~

~~new\_list = mylist[:]~~

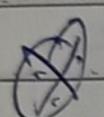
$\hookrightarrow$  a new copy of a list



~~yourlist = mylist~~

$\hookrightarrow$  points to same list (mylist here)

How to know by address or object?  $\rightarrow$  id()



Tuple  $\Rightarrow$

Immutable List (used for fixed)  
constant values

Similar to List but with parentheses

- instead of square bracket

(e.g.) ~~matrix\_dim~~ = (12, 12)  $\Rightarrow$   $12^{11} \times 12^{11}$   
std dimension

$\Rightarrow$  Entire tuple can be redefined

(e.g.): my\_dim = (12, 12)

my\_dim = (10, 9)

## 5. IF statements

if <condition>:  
  <  >                         == equal  
    | = not equal.

optional {  
  if <condition>:  
    <  >                         and => <cond1> and <cond2>  
  } or  
  else:  
    <  >                         or = <cond1> or <cond2>

Value in a list?

value in <list>

Value not in a list?

value not in <list>

Boolean => True  
                False

Empty List?

if <list>:

  | not empty  
  | action  
  else:

  | empty  
  | action

## 6. Dictionaries

28 | 02 | 2024

- key-value pair

3

`dict = {<key>:<value>, ... }`

(e.g.): alien\_Q = {'color': 'green', 'points': 5}  
so, points = alien\_Q['points']

\* Adding new pair  $\Rightarrow$

$$\text{alien} \odot [{}^1x_{{}^1p}{}^0s] = 0$$

alien- $\Theta$  ['y-pos'] = 25

\* Empty dict  $\Rightarrow$  alien = {}

modifying value  $\Rightarrow$  alien[0]['color'] = 'yellow'

Removing a pair  $\Rightarrow$  del alien- $\otimes$  [points]

Read a value with exception handling (e.g. no such key)

(e.g.) alien

point\_value = alien\_0.get('points', 'newer', value)

(e.g.):

point\_value = alien\_0.get('points')

here it returns special value None

## Looping:

for key, value in `<dict>.items()`:

body

② for key in <dict>.keys():  $\Rightarrow$  only keys

Looping (contd.)

③ for key in sorted(dict).keys():  $\Rightarrow$  keys in sorted order.

④ for value in dict.values():  $\Rightarrow$  only values in dict.

\* Sets: Braces, but no key-value pair.

(e.g.): languages = {'python', 'rust', 'c'}

Nesting - List of Dictionaries:

~~mylist = [dict1, dict2, ... dictn]~~

Nesting - List in a Dictionary:

(e.g.) pizza = {  
    'crust': 'thick',  
    'toppings': ['mushrooms', 'extra-cheese']  
}

Nesting - Dictionary in a Dictionary

(e.g.): users = {  
    'swamy': {  
        'first-name': 'Karappuswamy',  
        'last-name': 'Thangaraj',  
    },  
    'sangeeth': {  
        'first-name': 'Sangeeth',  
        'last-name': 'Alagavel',  
    },

## 7. User input & while loops 29/02/2024

input-value = input ("What is your name?") ①

input a number  $\Rightarrow$  All inputs are strings. So use `int()` to convert to number.

modulo operator  $\Rightarrow \%$   $\rightarrow$  returns remainder of two numbers divided

$$4 \% 3 \Rightarrow 1$$

while loops:

`while <Condition>:`

Condition examples:

`<expr> <operator> number <= 5`

status  $\rightarrow$  boolean

~~break~~

Exit a loop  $\Rightarrow$  break

Continue to next iteration skipping rest of body  $\Rightarrow$  continue

Loop with ~~list~~ list  $\Rightarrow$  each iteration fetches a value until it completes

`{ value = list.pop()`

`}`

Loop with a specific item in a list  $\Rightarrow$

`while <item> in <list>:`

01/03/2024

1:5

## 8. Functions

def <function-name>(<args>):

<  
body  
>

- Arguments  $\Rightarrow$  Let def describe\_pet(type, name):

Positional args

- order of args

(e.g.): describe\_pet('dog', 'willie')

Keyword args.

- name-value pairs in any order

(e.g.):  
describe\_pet(name='willie'  
              type='dog')

Default values  $\Rightarrow$

def describe\_pet(name, type='dog').

<  
>

default args should be last one

describe\_pet(name='willie')  
(or)

describe\_pet('willie')

- Return values  $\Rightarrow$

return <simple value> or <dict> or  
<list>

- Optional argument  $\Rightarrow$

def get\_name(first\_name, last\_name,  
              middle\_name="")

Passing a list =>

`def greet_users(username_list):`

modifying a list in a function =>

④ The modifications are permanent.

🚫 Preventing a function from Modifying a List =>

(Sending a copy)

Call => `function_name(list[:])`

→ copy of list.

🚫 Passing an arbitrary number of arguments =>

① `def function(*args):`

→ args. of Tuple

e.g. ① ('pepperoni')

② ('mushroom', 'pepperoni', 'cheese')

② ~~`def function(*args, regular_positional_args)`~~

Mixing with regular positional argument =>

`def function(size, *toppings):`

🚫 Using arbitrary keyword arguments =>

- Not knowing ahead of time what kind of information will be passed to the function.

(e.g.) `def build_profile(first, last, **user_info):`

< :

return user\_info

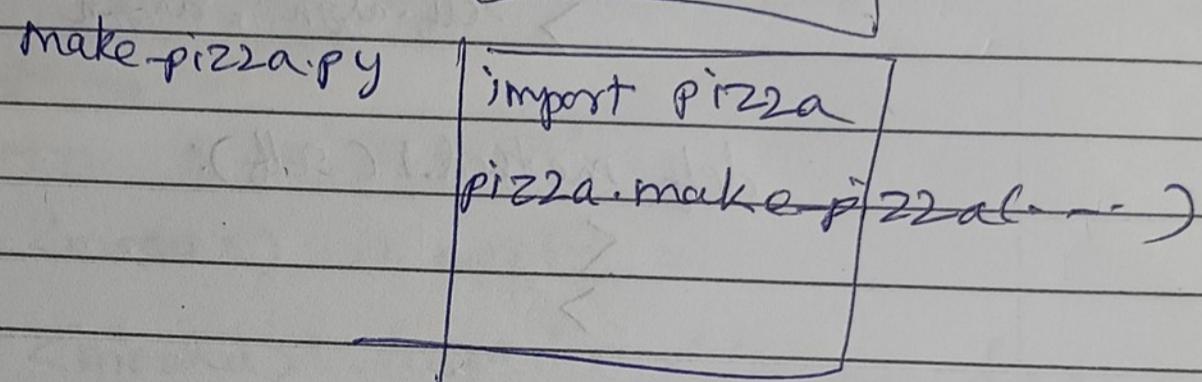
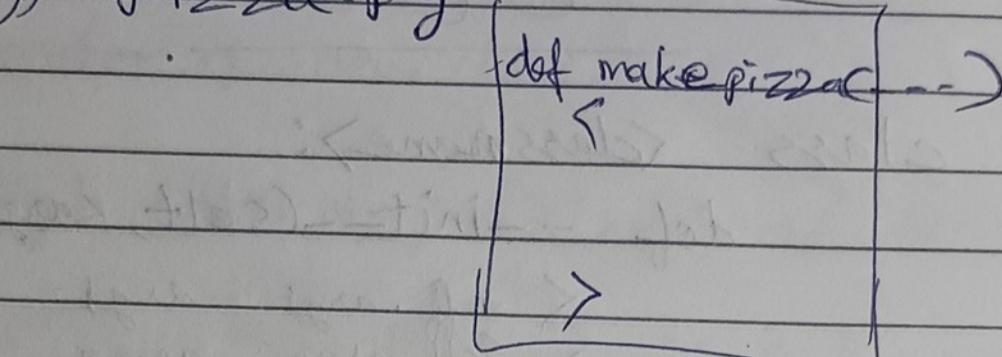
>

`user_profile = build_profile('albert', 'einstein',  
location='princeton',  
field='physics')`

02/03/2024

(Sat) 6

modules  $\Rightarrow$  a file ending in .py  
(e.g.) pizza.py



$\Rightarrow$  importing specific functions only

(e.g.): from pizza import make\_pizza

2: from pizza import ~~any~~ make\_pizza,  
deliver\_pizza,  
make\_bill...

$\Rightarrow$  Alias for a function

(e.g.): from pizza import make\_pizza as mkpz

$\Rightarrow$  Alias for a module

(e.g.) import pizza as pz

$\Rightarrow$  importing all functions in a module

(e.g.) ~~import~~ from pizza import \*

## 9. CLASSES

### OOP

class <classname>:

def \_\_init\_\_(self, <args>):

< self.arg1 = arg1

> self.arg1 = arg1

def method1(self):

<  
>

my\_instance = <classname>(<args>)

~~mymethod~~ →

my\_instance.method1()

setting default value of attribute => \_\_init\_\_(),

self.<attribute> = <value>

Inheritance => child class

class <childclassname>(<parentclassname>):

def \_\_init\_\_(self, <args>):

super().\_\_init\_\_(<args>)

my\_child\_instance = <childclassname>(<args>)

overriding method from the parent class =>

def <method name  
same as parent>(self):

<

>

If overridden method has different parameters than parent?  
→ still it will override

Instances as Attributes =>  
create an instance of another class with  
constructor of new class.

Importing multiple classes from a module =

→ put put all classes in a module

→ ~~import \*~~

→ ~~from <module> import \*~~

from <module> import <specific class>

~~from import <class>~~

## 10. FILES & EXCEPTIONS

Reading from a file:

```
from pathlib import Path  
path = Path(<filename>)
```

↳ either absolute or  
relative path with  
forward slash (/)

```
contents = path.read_text()      => entire file  
lines = contents.splitlines()    => format it as  
for line in lines:  
    print(line)
```

if <pattern> in ~~lines~~ contents : => check for a pattern  
in file content

<  
>

Writing to a file:

```
path.write_text(<string content>)
```

↳ only writes string,  
numeric data to be  
converted to string.

↳ separated by \n for  
newlines.

Exceptions =>

try:

< code which may throw exception => print(5/0)

except <exception object>: → ZeroDivisionError

else: <exception handler> → print("divide by zero is illegal")  
<normal handler>

- Try exception throwing start with try-except block.

FileNotFoundException exception =>

try:

contents = path.read\_text(<filename>)

except FileNotFoundError:

print(contents)

print("unable to read file")

else:

print(contents)

Note: content.split() returns ~~no. of words~~ list of words.

- multiple files handling =>

filenames = ['file1', 'file2', ..., 'fileN']

for filename in filenames:

path = Path(filename)

- failing silently =>

except <exception object>:

pass

Storing Data  $\Rightarrow$

JSON - JavaScript Object Notation

~~from json~~

import json

path1 = Path(<jsonfile>)

contents1 = json.dumps(<data>)  $\Rightarrow$  writing data  
(e.g.) list as json format

Path1.write\_text(contents1)

path2 = Path(<jsonfile>)

contents2 = path2.read\_text()

<data> = json.loads(contents2)

(e.g.) list

- Note: check if file exists  $\Rightarrow$

if path.exists():



03/03/2024

## 11. TESTING YOUR CODE

(2)

Installing [pytest] =>

\$ ~~python -m pip~~

\$ ~~python -m pip install  
upgrade~~

\$ python -m pip install --upgrade pip ↴

\$ python -m pip install --user pytest ↴

Testing a Function =>

① < my-functionfile.py

```
def myfunction( )
```

```
>
```

② Now need to write a pytest file as below :

test-myfunctionfile.py / mandatory

```
from myfunctionfile import  
myfunction
```

```
def test_<testname>():
```

```
<"call the myfunction to be  
tested.
```

```
assert <condition>
```

③ Run test: \$ pytest ↴

of | 03 | 2024

1

Testing a Class =>

Using Fixtures => Helps to set up a test environment

import pytest

from <tested module> import <test class>

for <- @pytest.fixture

def <common function - for all tests>():

<

> returns <common data> → input for all other tests follow

def test - test1(common data):

<

> def test - test2(common data):

<

>

05/03/2014

## Non-Gaming

(4)

### ~~Learnings from PROJECTS~~

~~PROJECT-1~~

#### 12. A Ship that Fires Bullets

① `sys.exit()` → To exit python application

② `if __name__ == '__main__':`

→ This check is to test whether this python file is the top level one called from shell.

③ If double underscore is missing in `__init__` of constructor (even if single underscore), python doesn't throw error, it silently ignores and considers it as just another function. So this constructor function won't be called during object creation.

[Page] 1232 ④ It is convention that helper function name is ~~not~~ prefix with a underscore (-).

⑤ `float(value)` returns floating point value when you want to preserve decimals.

⑥ Added feature in game:

While shooting a bullet, a wav file is played for nice game effect.

## 13. ALIENS

06/03/2024

Adds delay in seconds  $\Rightarrow$

(4)

~~from time import sleep~~

sleep(<secs>)

(X) Added Alien-Hit & Ship-Hit audio effects.

## 14. SCORING

08/03/2024

(3)

formatted string  $\Rightarrow$

f "<{<variable>:;}"

(2)

$\hookrightarrow$  add , in appropriate decimal places.

11/03/2024

PROJECT 2

## 15. GENERATING DATA

③

### matplotlib

①

```
import matplotlib.pyplot as plt  
squares = [1, 4, 9, 16, 25]  
fig, ax = plt.subplots()  
ax.plot(squares)  
plt.show()
```

X is missing  
- Must enter  
x param  
for meaningful plot

optional:  
lineWidth  
fontStyle  
fontSize

an.set\_title()  
an.set\_xlabel()  
an.set\_ylabel()  
an.tick\_params()

(~~center of~~) plt.style

plt.style.use()

②

ax.scatter

```
ax.scatter(x, y)  
optimal: color = < >  
optimal: s = <n>  
+ single value or list of values
```

x-values = range(1, 100)

y-values = [x\*\*2 for x in x-values]

ax.axis([0, 100, 0, 100])

xrange      xrange

plt.savefig() → to save plot as figure

## Random walks

from random import choice

choice(<list of values>)  $\Rightarrow$

- returns a random value from the list.

This is used to generate x & y values for data set used to plot.

## Rolling Dice with Plotly

from →

— Install plotly & pandas —

from random import randint

randint(<from value>, <To value>)

$\hookrightarrow$  returns in given range

Here die rolled 100 times & results are analysed

- and found no. of times each face (side) is thrown  
 $\Rightarrow$  frequencies[], poss\_results: [1, 2, 3, 4, 5, 6]

Histogram for above Die rolling case:

import plotly.express as px

fig = px.bar(x=poss\_results,  
y=frequencies)

fig.show()

px.bar() options  $\Rightarrow$  title = " "

labels = {'x': <xlabel>, 'y': <ylabel>}

fig.write\_html(<HTMLfilename.html>)

## 16. DOWNLOADING DATA

12 | 03 | 2024

(2)

### CSV file handling :

```
from pathlib import Path  
import csv
```

```
path = Path(<path of file>)
```

```
lines = path.read_text().splitlines()
```

```
an object} ← reader = csv.reader(lines)  
of a row } header_row = next(reader)  
          print(header_row)
```

```
for index, column_header in  
    enumerate(header_row):  
    print(index, column_header)
```

```
for row in reader:
```

Here row[0] - 1st column of row  
row[4] - 5th column of row

### datetime handling =>

```
from datetime import datetime
```

```
date object ← date = datetime.strptime(  
                  '2021-07-01', '%Y-%m-%d')
```

Error handling =>  
(missing data on csv)

```
try:  
    <extract column value>  
except ValueError:  
    print(<index> at <column>)  
else:  
    <normal processing>
```

14/03/2024

2:5

## JSON file handling

```
import pathlib import Path  
import json
```

```
path = Path("filepath.json")  
contents = path.read_text()
```

dictionary  $\leftarrow$  json\_data = json.loads(contents)

len(json\_data)  $\Rightarrow$  no. of records.

< iterate through dictionary key  
and prepare list of data >

building world map  $\Rightarrow$

```
import plotly.express as px
```

```
fig = px.scatter_geo(lati, long, title)
```

```
fig.show()
```

optional:

size  $\Rightarrow$  size of dot  
size of dot

```
color_continuous_scale = "color"  
labels = {'color': 'magnitude'}  
projections = natural earth
```

## 17. WORKING WITH APIs

\$ python -m pip install --use requests

import requests

r = requests.get (<url with query>,  
<headers>)

→ r.status\_code => status code

response\_dict = r.json()

\* Aware of API Rate limit

\* Most probably you need API key or access token.

15/03/2024

PROJECT 3

## 18. Getting Started with DJANGO

(2)

- create venv

Web framework

```
$ python -m venv ll-env ↴
```

```
$ source ll-env/bin/activate ↴
```

```
[ll-env]$ deactivate ↴
```

```
$ pip install --upgrade pip
```

```
$ pip install django
```

```
$ django dr $ ls
```

```
$ django-admin startproject <projectname>
```

```
$ ls
```

```
$ python manage.py migrate ↴
```

```
$ ls
```

```
$ python manage.py runserver <port no>
```

optional.

Making Pages

16/03/2024

(1)

Making Pages (contd.)

17/03/2024

(3)

## 19. USER ACCOUNTS

18/03/2024

②

forms => from django import forms

## setting up User Accounts

19/03/2024

③

SQLITE3

Additional learning while debugging:

# apt install sqlite3

(11.env)\$ python manage.py dbshell

{  
    sqlite commands.

)

## 20. STYLING AND

## DEPLOYING AN APP

20/03/2024

①

Styling an app for various devices =>

django-bootstrap5 app

21/03/2024

②

Bug fixing done:

① Topics - navbar link right aligned  
due to improper </div> usage.

② Edit entry: added a new entry  
instead of editing due to  
wrong html called.  
— End —

③