iscte

INSTITUTO
UNIVERSITÁRIO
DE LISBOA

emprego
digital

UPskill

<Welcome To>

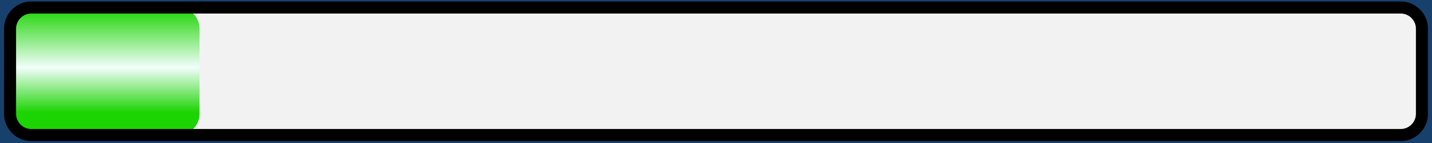
python™

1.

Overall Program Content

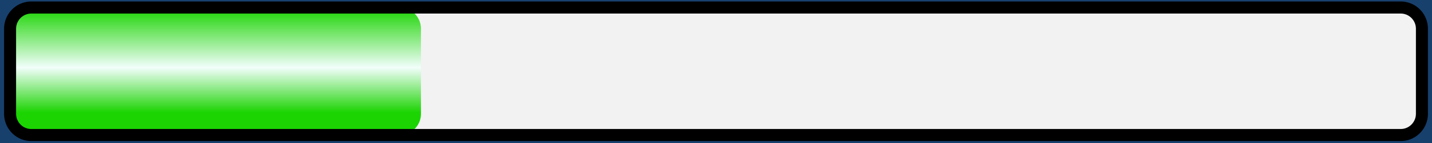
Web development with Python	Hours
Work skills development	50
Python Programming Introduction	150
Web Programming Introduction (html/css)	100
Databases Concepts and Structures	50
Web Servers Programming	150
Web services development	150
Total	650

Python programming Introduction Content



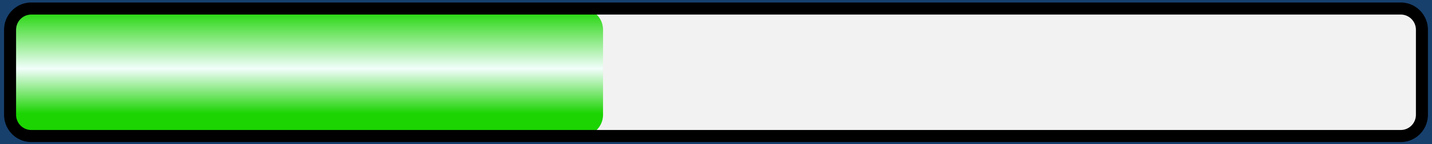
Python programming Introduction Content

2. Data types/outputs/inputs
3. Operators
4. Functions and Modules



Python programming Introduction Content

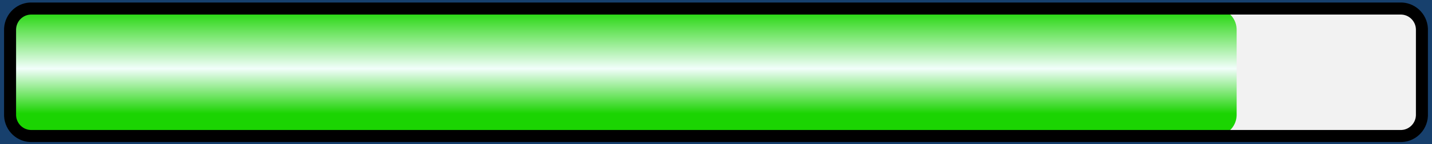
5. Conditional statements and expression
6. Loops
7. Work with standard Library and Modules



Python programming Introduction Content

8. Data structure in python
9. List,
10. Tuple,
11. Dictionaries,
12. Set

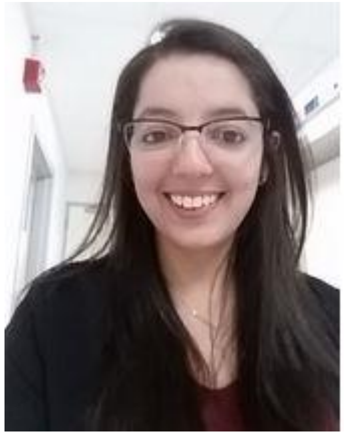
Python programming Introduction Content



Python programming Introduction Content

100% Loaded

Our Professors:



Joseanne Viana (Josi)

Email: jcova1@iscte-iul.pt

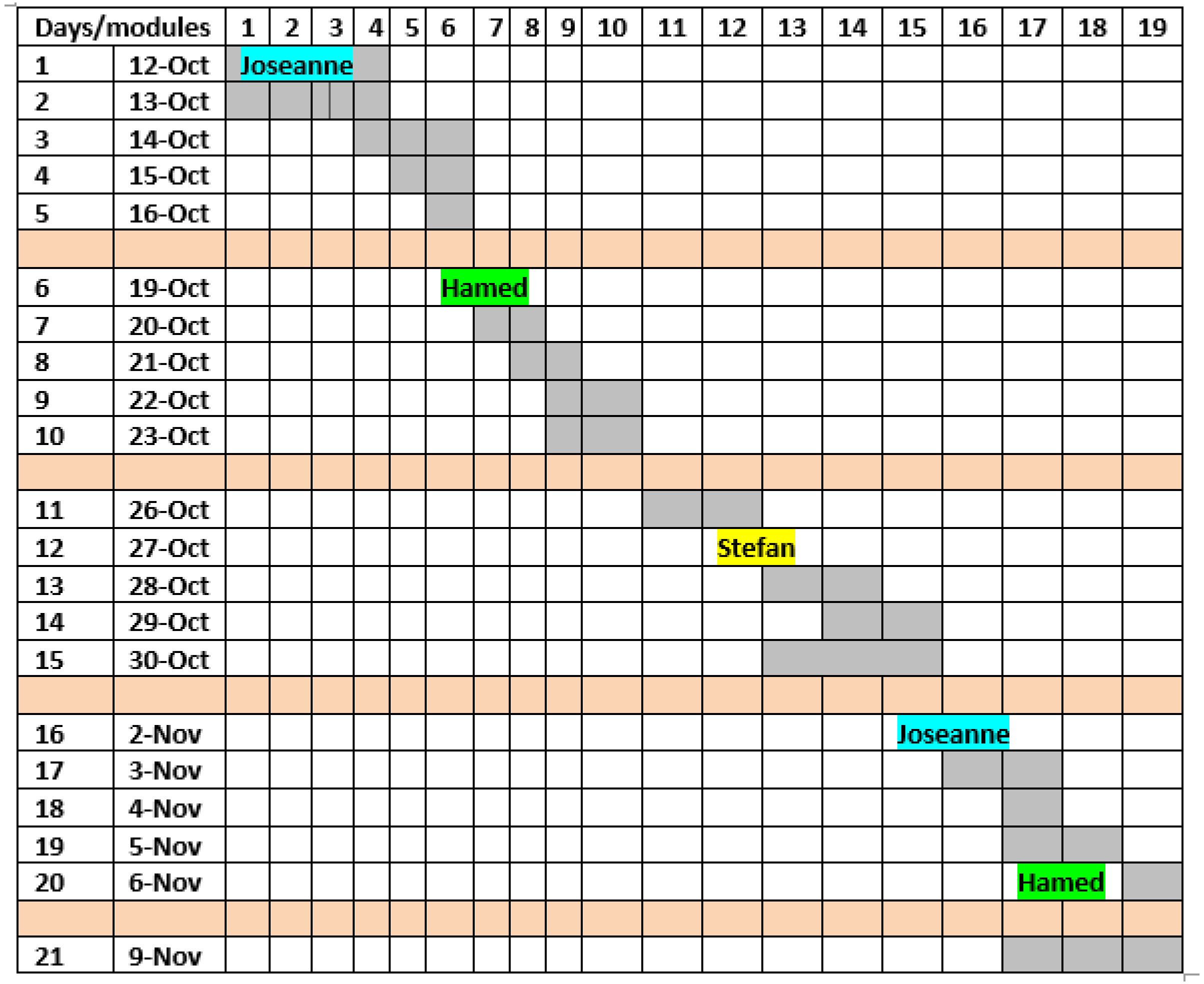
Stefan Postolache

Email:stefanpostolache@edu.ulisboa.pt

Hamed Farkhari

Email:Hamed_Farkhari@iscte-iul.pt

Schedule



Days/modules		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	12-Oct	Joseanne			▨															
2	13-Oct	▨	▨	▨	▨															
3	14-Oct				▨	▨	▨													
4	15-Oct					▨	▨													
5	16-Oct						▨													
6	19-Oct						Hamed													
7	20-Oct							▨	▨											
8	21-Oct								▨	▨										
9	22-Oct									▨	▨									
10	23-Oct									▨	▨									
11	26-Oct											▨	▨							
12	27-Oct												Stefan							
13	28-Oct													▨	▨					
14	29-Oct														▨	▨				
15	30-Oct													▨	▨	▨				
16	2-Nov															Joseanne				
17	3-Nov																▨	▨		
18	4-Nov																	▨		
19	5-Nov																	▨	▨	
20	6-Nov																	Hamed		▨
21	9-Nov																	▨	▨	▨

<Let’s get started >

Contents

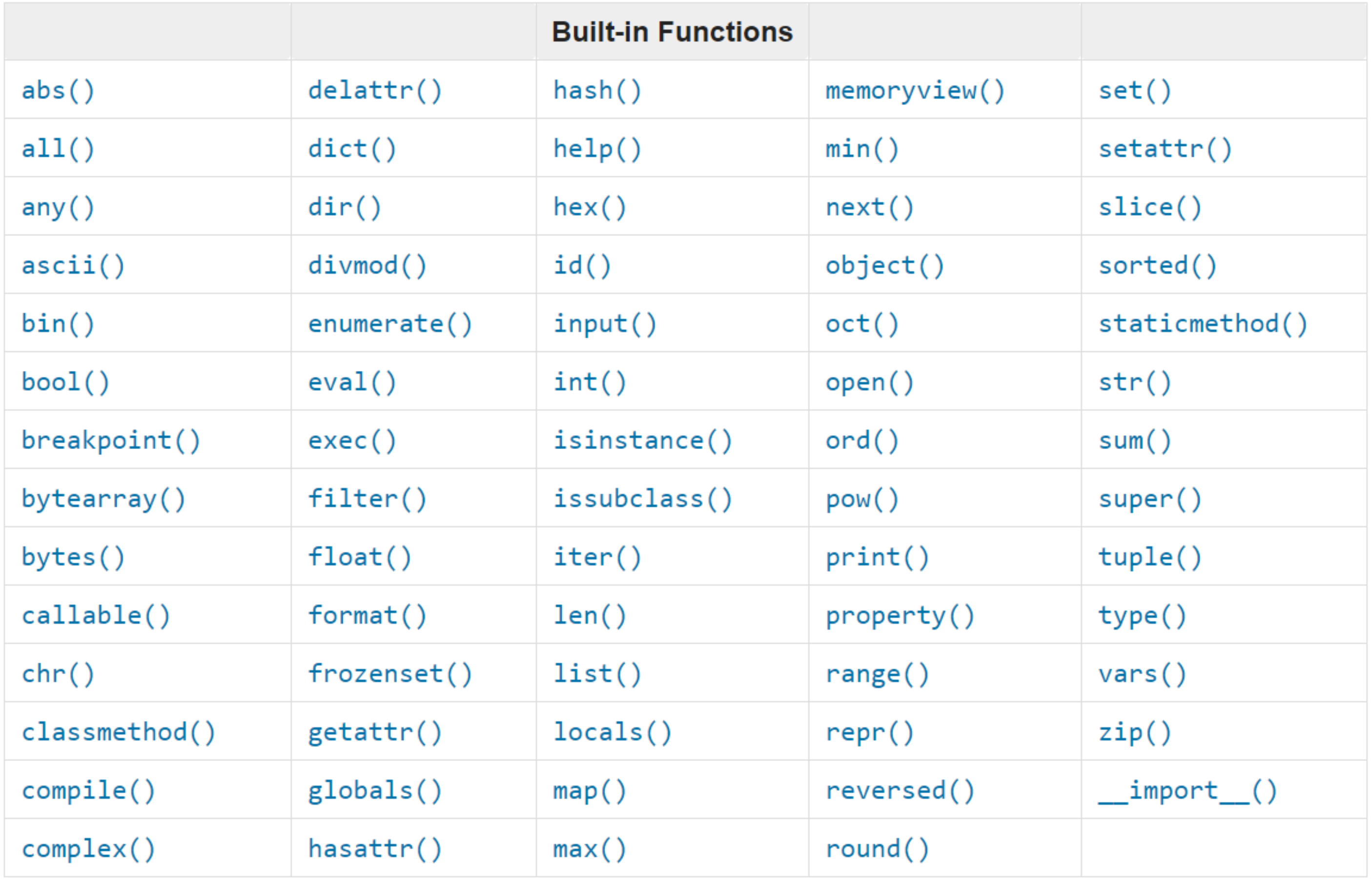
Standard Library & Modules

iscte
INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

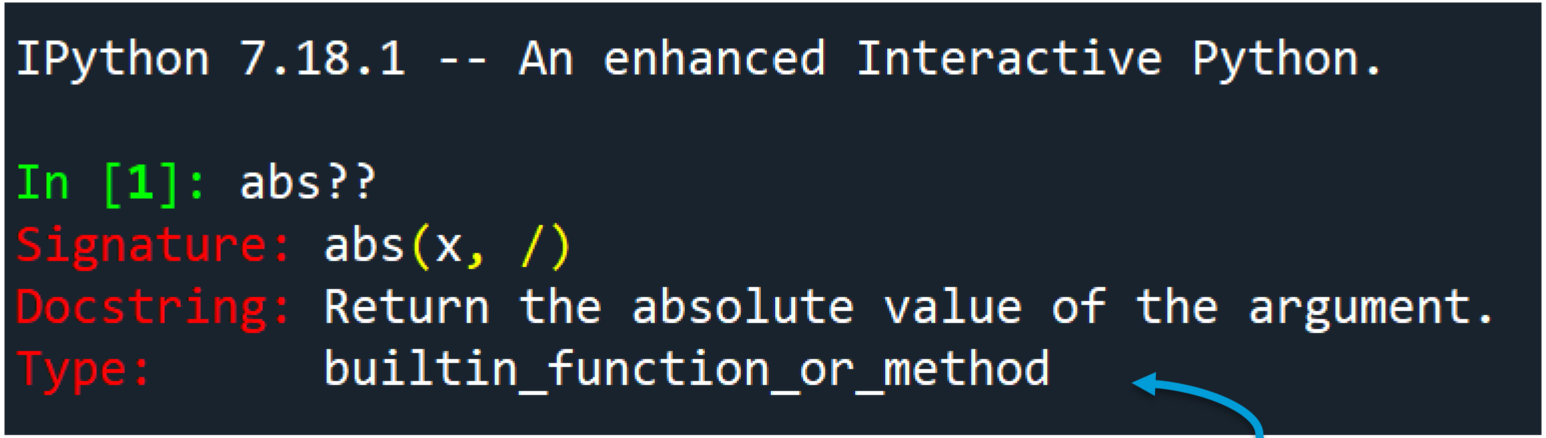
Built-in Functions

The Python interpreter has a number of functions and types built into it that are always available.



		Built-in Functions		
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

Built-in Functions



```
IPython 7.18.1 -- An enhanced Interactive Python.

In [1]: abs??
Signature: abs(x, /)
Docstring: Return the absolute value of the argument.
Type:      builtin_function_or_method
```

But I can not see the code of abs function!

You can see the "builtin" word in the type section

Standard Library

The Python Standard Library contains a huge number of useful **modules**

It is important to become familiar with the Python Standard Library since many problems can be solved quickly if you are familiar with the range of things that these libraries can do.

The library contains built-in modules some **(written in C)** that provide access to system functionality such as file I/O

Standard Library

The following are among the most important modules in standard library:

Time , sys , os , math , random
Pickle , urllib , re , cgi , socket

Module

A module is a file consisting of Python code.

A module can define functions, classes and variables.

A module can also include runnable code.

Import modules

1) import module

Example:

```
import  math
math . pi                             # 3.141592653589793
```

2) import module as new_name

Example:

```
import  math as m
m . pi                                # 3.141592653589793
math . pi                             # Error: name 'math' is not defined
```

Import modules

3) From module import submodule/function/variables

Example:

```
From   os   import   getcwd
getcwd()                    # 'C:\\Users\\user'
```

4) From module import submodule as new_name

Example:

```
from   os   import   getcwd   as   gc
gc ()                     # 'C:\\Users\\user'
```

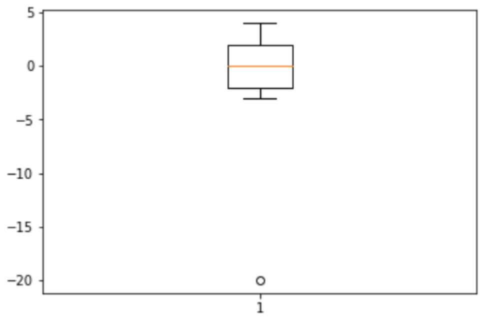
Import modules Example

Example:

```
from   math   import    e , pi
e                                    # 2.718281828459045
```

Example:

```
import   numpy   as     np
import   pandas  as     pd
import   matplotlib.pyplot   as   plt
data = np.array ( [-20 , -3 , -2 , -1 , 0 , 1 , 2 , 3 , 4] )
plt.boxplot( data )
```

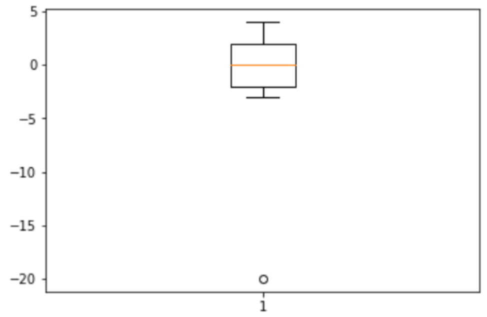


<Exercise 1>

Exercise

Exercise:

```
import   numpy   as    np
import   pandas  as    pd
import   matplotlib.pyplot   as   plt
data = np.array ( [-20 , -3 , -2 , -1 , 0 , 1 , 2 , 3 , 4] )
plt.boxplot( data )
```



Replace the below code with the other type of import modules and remove '.' ?

```
import   matplotlib.pyplot   as   plt
from matplotlib  import   pyplot   as   plt
```

Import modules
Bad idea!

This way is Not a good idea!

5) From module import * #means import everything

Example:

```
from math import *

pi                              # 3.141592653589793
```

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

Import everything Example

Why?

From module import * #means import everything

Module A contain F function
Module B contain F functions too.

From A import *
From B import *
F(...) $\longleftarrow$ F belong to A Or B ?

Example:
sum??
From numpy import *
sum??

Import modules Example

Example:

```
import matplotlib.pyplot as plt
```

```
from matplotlib import pyplot as plt
```

```
import random
import math


for i in range(5):
    print(random.randint(1, 25))

print(math.pi)
```

Example:

import module

module . function()

Output:

```
22
13
22
16
12
3.1415...
```

Find Functions Attributes

`dir()`

1. for a module object return the module's attributes.

2. for any other object return its attributes, its class's attributes, and recursively the attributes of its class's base classes.

What is the result of these codes?

Example

```
import   math
dir ( math )
```

Example

```
s  =  ‘ a ’
dir ( s )
```

Help and Documentation

help() __doc__

help (len)

```
Help on built-in function len in module builtins:

len(obj, /)
    Return the number of items in a container.
```

len . __doc__

```
'Return the number of items in a container.'
```

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

?
??

```
def square(a):
    "Return the square of a"
    return a ** 2
```

```
Square?

Signature: square(a)
Docstring: Return the square of a
Type:      function
```

```
Square??

Signature: square(a)
Source:
def square(a):
    "Return the square of a"
    return a ** 2
Type:      function
```

<Exercise 2>

Exercise

What these function do?

In math module:

fmod(9,4)

gcd(30,4)

fabs(-4)

In random module:

randint(1, 5)

choice([1, 5])

a = [1, 2, 3, 4]

shuffle(a)

Some Module Example

```
import math
print( math.sqrt(4))          #2.0
print( math.trunc(2.3))     #2
print( math.floor(2.3))     #2
print( math.ceil(2.3))       #3
print( math.factorial(4))    #24 , 4! = 4*3*2*1
print( math.log2(32))        #5.0
print( math.log10(100))    #2.0
print( math.e)                     #2.7
print( math.log(32))           #3.46
print( math.sin(5))             #-0.9
print( math.fmod(9,4))      #1.0 , 9%4
print( math.gcd(30,4))      #2 , greatest common divisor
print( math.fabs(-4))          #4.0 , float abs
print( abs(-4))                   #4
print( math.pow(2,3))        # 8.0
print( pow(2,3))                 # 8
print( math.pi)                   # 3.1415926…
print(f'{math.pi :.2f}')      # 3.14
```

iscte
INSTITUTO UNIVERSITÁRIO DE LISBOA
emprego digital

Some Module Example

```
import random

print( random . randint(1, 5))     # random number between 1 to 5

print( random . choice([1, 5]))  # random choice between only 1 or 5

a = [1,  2,  3,  4]

random . shuffle(a)                          # randomize arrangement of a

print(a)                                     # [4, 2, 1, 3]
```

Some Module Example

```
import sys

print( sys . version)          # 3.8.5

print( sys . platform)          # win32



import platform

platform . release()          # 10 ➔ it means windows 10
```

Some Module Example

```
import datetime

now = datetime . datetime . now()

print(now)                          # 2020-10-20 11:30:47.724484

print( now . year)                  # 2020

print( now . month)                 # 10

print( now . day)                   # 20

print(datetime . datetime . today() )  # 2020-10-20 11:31:45.597811

type(now)            # datetime.datetime
```

<Exercise 3>

Exercise

According Previous slide example with some changes,

use datetime module

If the current minute is odd

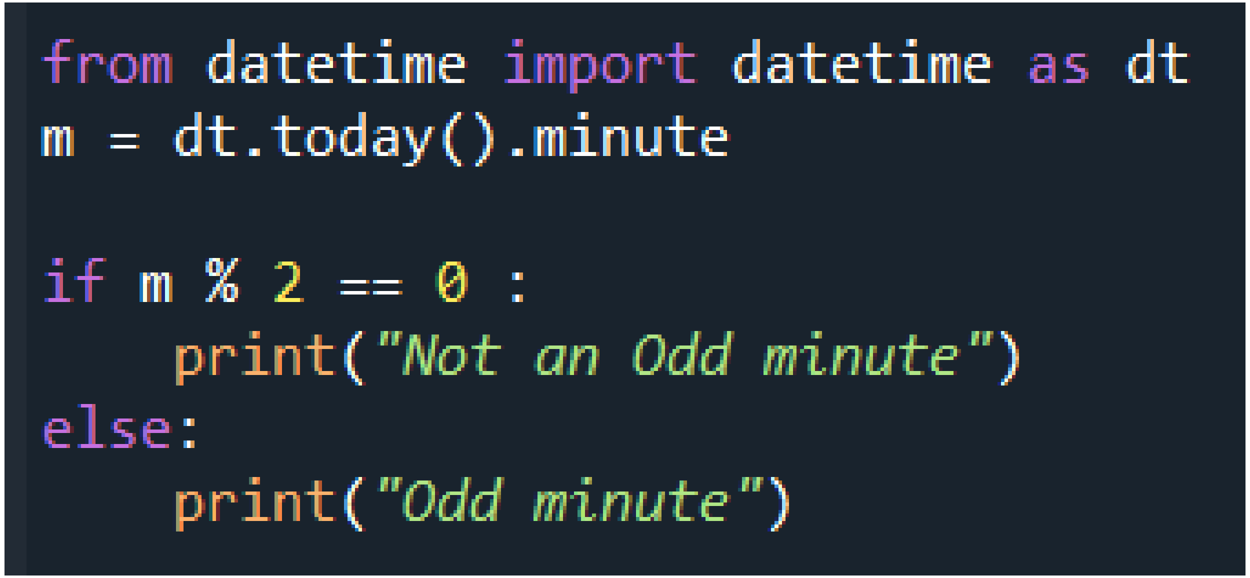
show "Odd minute" in output

If the current minute is even

show "Not an Odd minute" in output

Exercise

Solution 1:

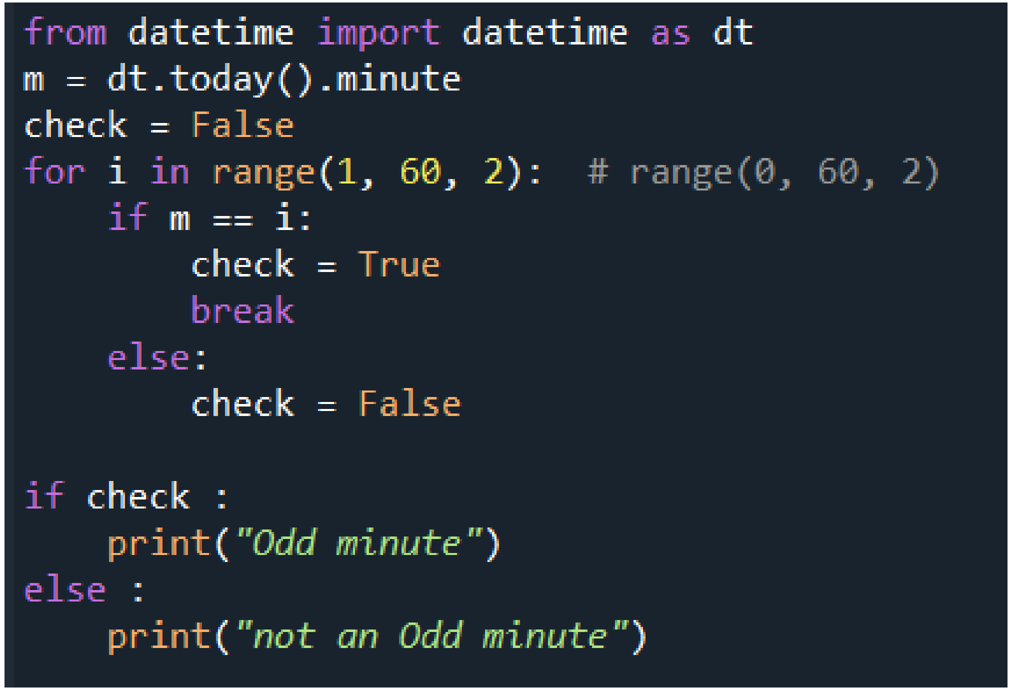


```python
from datetime import datetime as dt
m = dt.today().minute

if m % 2 == 0 :
    print("Not an Odd minute")
else:
    print("Odd minute")
```

Exercise

Solution 2:

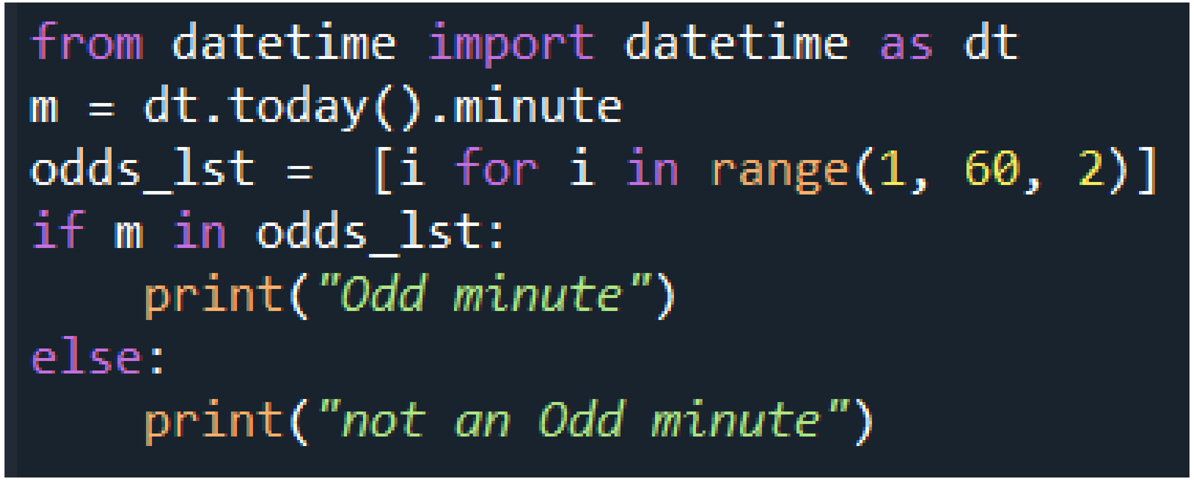


```python
from datetime import datetime as dt
m = dt.today().minute
check = False
for i in range(1, 60, 2):  # range(0, 60, 2)
    if m == i:
        check = True
        break
    else:
        check = False

if check :
    print("Odd minute")
else :
    print("not an Odd minute")
```

Exercise

Solution 3:



```python
from datetime import datetime as dt
m = dt.today().minute
odds_lst =  [i for i in range(1, 60, 2)]
if m in odds_lst:
    print("Odd minute")
else:
    print("not an Odd minute")
```

“

- *Make it work*
- *Make it Right*
- *Make it Fast*

O futuro profissional começa aqui

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

UPskill