Python for Data Science and Machine Learning

School Year 2023-2024

IST

Course Structure

1	Introduction to Python	OCT 19	OCT 26	NOV 9	NOV 16	NOV 23	NOV 30	DEC 7
2	Introduction to Data Science	DEC 14	DEC 21	JAN 11	JAN 18			
3	Introduction to Machine Learning	JAN 25	FEB 1	FEB 8	FEB 22			
4	Data Science & ML Challenge	FEB 29	MAR 7					

🟥 = Core Topics ⬚ = Optional Topics

IST since 1963 INTERNATIONAL SCHOOL OF TURIN

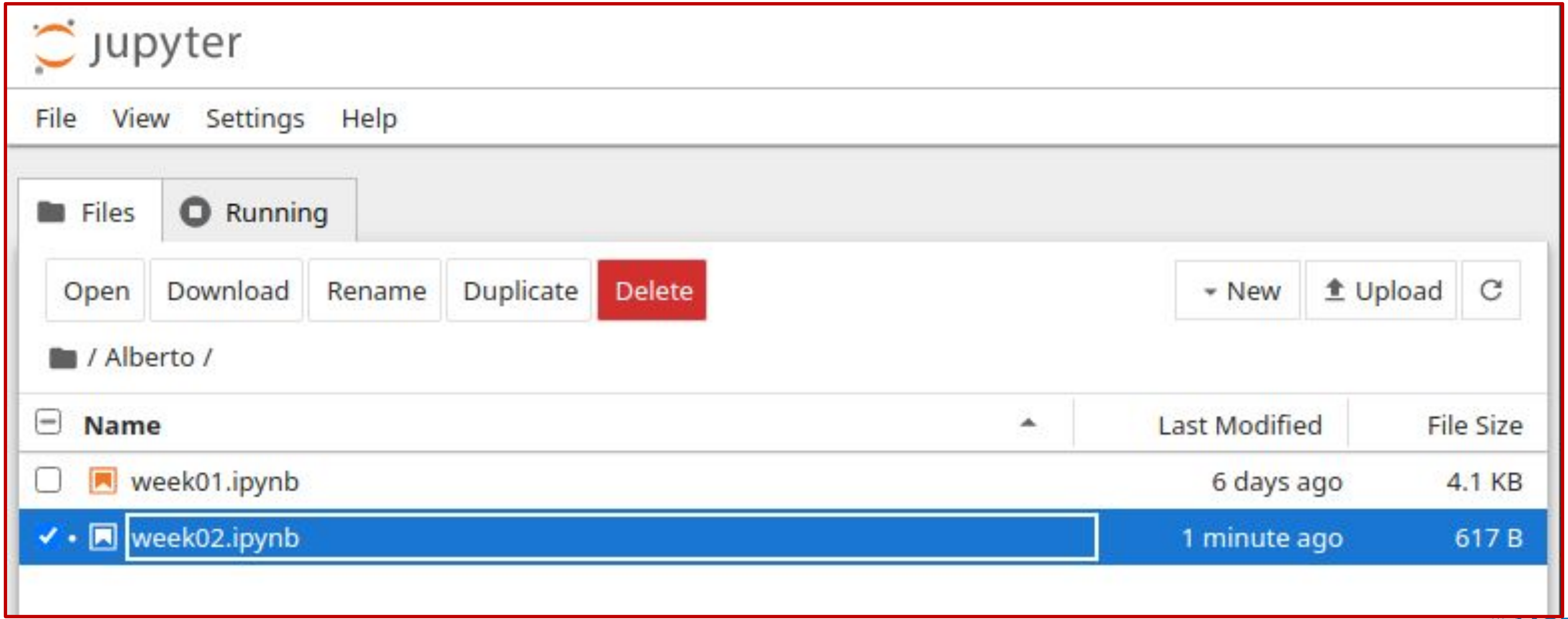
Jupyter Notebook Setup

In a browser:

```
192.168.10.4:8888
```

Password: **ist**

Jupyter Notebook Setup

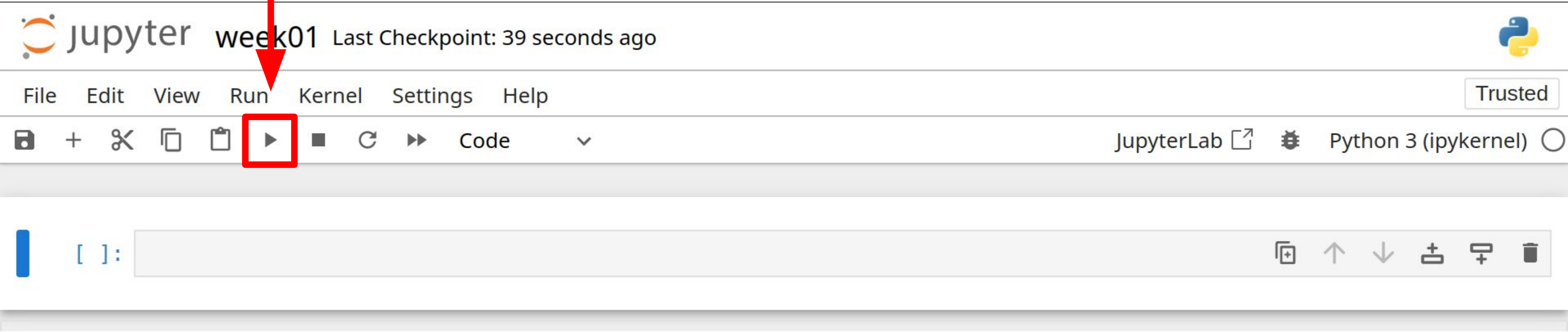


/ Alberto /

	Name	Last Modified	File Size
☐	week01.ipynb	6 days ago	4.1 KB
✓	week02.ipynb	1 minute ago	617 B

Jupyter Notebook Structure

Run Cell


jupyter
week01 Last Checkpoint: 39 seconds ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
[ ]:

Recap: Comparisons

- 5 is larger than 3

```
5 > 3
```

- -5 is larger than 9

```
-5 > 9
```

- 2 is the same as 2

```
2 == 2
```

- **not** (negation)

```
not True
```

```
not (5 < 3)
```

- **and** (both must be true)

```
(5 < 6) and (5 < 10)
```

- **or** (either must be true)

```
(5 < 3) or (5 < 10)
```

Recap: If-Statements

Allow for branches in your code!

```
x = 5

if x < 10:
    print("X is small")
else:
    print("X is large")
```

```
x = 20

if x < 10:
    print("X is small")
else:
    print("X is large")
```

NOTE: You **do not** need an else block, it’s optional.

Recap: If-Statement chaining

You can chain multiple conditions with **`elif`**.

What is the difference between these two snippets of code?

```
x = int(input())

if x < 3:
   print("X is less than 3")
elif x < 10:
   print("X is less than 10")
elif x < 25:
   print("X is less than 25")
```

```
x = int(input())

if x < 3:
   print("X is less than 3")
if x < 10:
   print("X is less than 10")
if x < 25:
   print("X is less than 25")
```

Recap: While-Loops

Allows you to repeat instructions

With an **if-statement**:

```
x = int(input("Insert num < 5: "))

if x >= 5:
   print("ERROR! Wrong number")
   x = int(input("Insert num < 5: "))

print("CORRECT!")
```

With a **while-loop**:

```
x = int(input("Insert num < 5: "))

while x >= 5:
   print("ERROR! Wrong number")
   x = int(input("Insert num < 5: "))

print("CORRECT!")
```

Recap: For-Loops

Repeat a specific amount of times

With a **while-loop**:

```
x = 0

while x < 5:
    print(x)
    x += 1
```

With a **for-loop**:

```
for x in range(5):
    print(x)
```

Recap: For-Loops

Anatomy of a for-loop:

1. Uses the **for** keyword
2. Ends with a colon (**:**)
3. Takes up to 3 parameters:
   a. Start number (optional, default is 0)
   b. End number
   c. Step-size (optional, default is 1)

```
for x in range(10):
    print(x)
```

```
for x in range(2, 10):
    print(x)
```

```
for x in range(2, 10, 3):
    print(x)
```

Remember: String Operations

Remember the following string operation shorthands:

Repetition:

```
x = "*"
print(x * 5)
```

Concatenation:

```
x = "***"
y = "..."
print(x + y + x)
```

Exercise

Complete the **4.0** program.

- It takes an input **num** from the user.
- It should print out a full square pattern of size **num**.

Example if **num=5**:

```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

Exercise 4.0 - Solution

```python
num = int(input("Insert the size of the square: "))

for x in range(num):
    print("*" * num)
```

Exercise

Complete the **4.1** program.

- It takes an input **num** from the user.
- It should print out a triangular pattern of size **num**.

Example if **num=5**:

```
....*
...**
..***
.****
*****
```

Exercise 4.1 - Solution

```python
num = int(input("Insert the size of the triangle: "))

for x in range(1, num + 1):
    n_stars = x
    n_dots = num - n_stars
    stars = n_stars * "*"
    dots = n_dots * "."
    print(dots + stars)
```

Lists

Modifiable containers for data.

With **variables**:

```
num1 = 42
num2 = 100
num3 = 10

print(num1)
print(num2)
print(num3)
```

With a **list**:

```
nums = [42, 100, 8]

print(nums)
```

Lists

Anatomy of a list:

1. Uses square brackets **[ ]**
2. Elements separated by comma **,**
3. Can take any values

```
nums = [42, 100, 8]
```

```
data = [4.2, "cat", 8]
```

Modifying Lists

Adding new elements:

1. To insert at the back: **append**
2. To insert in any position: **insert**

```
nums = [42, 100]

nums.append(8)
nums.insert(0, 200)
nums.append(51)

print(nums)
```

Exercise

Complete the **4.2** & **4.3** programs.

- **4.2**: Initialise a list called **nums** with three numbers inside:
  - 6, 90 and 43

- **4.3**: Add the following numbers to the **nums** list: 3, 21, 17
  - **HINT**: Use **append** or **insert**!

Exercise 4.2 & 4.3 - Solution

4.2

```
nums = [6, 90, 43]

print(nums)
```

4.3

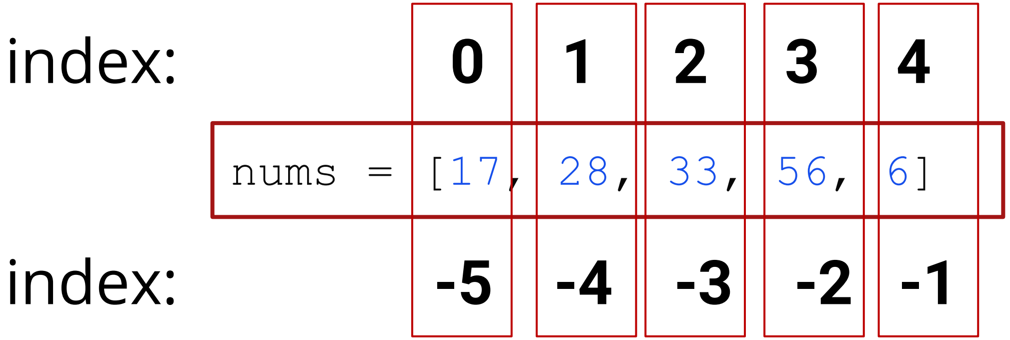
```
nums.append(3)
nums.append(21)
nums.append(17)

print(nums)
```

Accessing List Elements

To access list elements you can use the **[index]** operator.

NOTE: List indices start from **0**



```
nums = [17, 28, 33, 56, 6]
```

```
print(nums[0])
```

```
print(nums[3])
```

```
print(nums[-2])
```

Exercise

Complete the **4.4** & **4.5** programs.

- **4.4**: Given the **nums** list defined in previous exercises, print the first, third, last and second-last elements.

- **4.5**: Write a program that generates a list **data** with all the numbers from 1 to 20.
  Then print the 5th, 10th, 15th and last element.

Exercise 4.4 & 4.5 - Solution

4.4

```
print(nums[0])
print(nums[2])
print(nums[-1])
print(nums[-2])
```

4.5

```
data = []

for x in range(1, 21):
    data.append(x)

print(data[4])
print(data[9])
print(data[14])
print(data[-1])
```

Concatenating Lists

You can concatenate lists with the **extend** function.

Otherwise you can also use addition.

```
left = [1, 2, 3]
right = [4, 5, 6]

left.extend(right)
print(left)
```

```
left = [1, 2, 3]
right = [4, 5, 6]

new = left + right
print(new)
```

Exercise

Complete the **4.6** program.

- Create list **numsL** with all numbers from 0 to 9 (inclusive)
- Create list **numsM** with a single number: 10
- Create list **numsR** with all numbers from 11 to 19 (inclusive)

Concatenate the three lists and print the output list out

Exercise 4.6 - Solution

```
numsL = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
numsM = [10]
numsR = [11, 12, 13, 14, 15, 16, 17, 18, 19]

print(numsL + numsM + numsR)
```

Removing List elements

You can remove elements in a list with the **pop** function.

You may optionally pass an index, default is **-1**.

```
data = [4, 8, 12, 16, 20]
data.pop()
print(data)
```

```
data = [4, 8, 12, 16, 20]
data.pop(2)
print(data)
```

```
data = [4, 8, 12, 16, 20]
num1 = data.pop(2)
num2 = data.pop(-2)
print(num1 + num2)
print(data)
```

Exercise

Complete the **4.7** program.

1. Create a list `data` that stores the 6 integers between 10 and 15 (inclusive).
2. Using the `data` that list, appending 2 additional integers: `20` and `65`
3. Then create a blank list `blank`
4. Insert the integer `34` at Index 0 of `data`
5. Remove the last element of `data` and insert it at the beginning of `blank`
6. Print the two lists concatenated

What is the output of the program?

Exercise 4.7 - Solution

```python
# 1
data = [10, 11, 12, 13, 14, 15]
# 2
data.append(20)
data.append(65)
# 3
blank = []
# 4
data.insert(0, 34)
# 5
last_element = data.pop()
blank.insert(0, last_element)
# 6
print(blank + data)
```

Output:

```
[65, 34, 10, 11, 12, 13, 14, 15, 20]
```

End of Class

See you all next week!