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

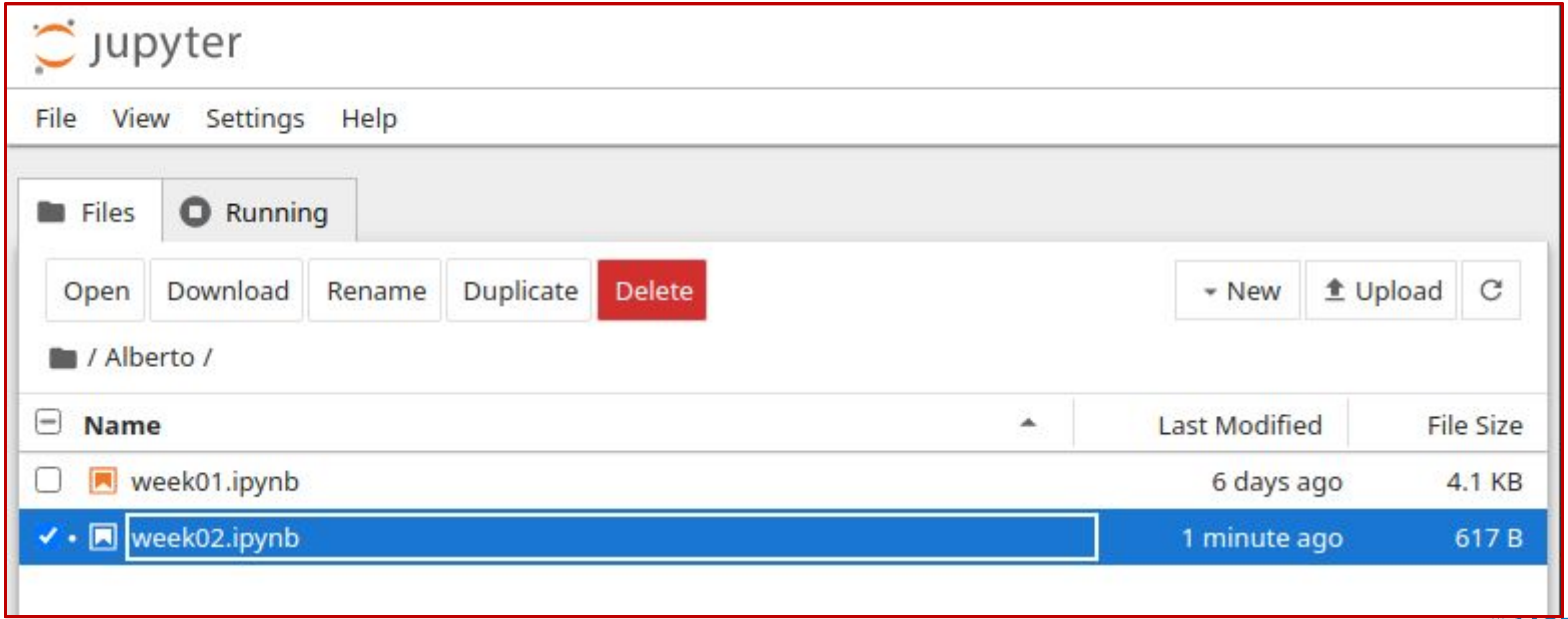
Jupyter Notebook Setup

In a browser:

```
192.168.10.4:8888
```

Password: **ist**

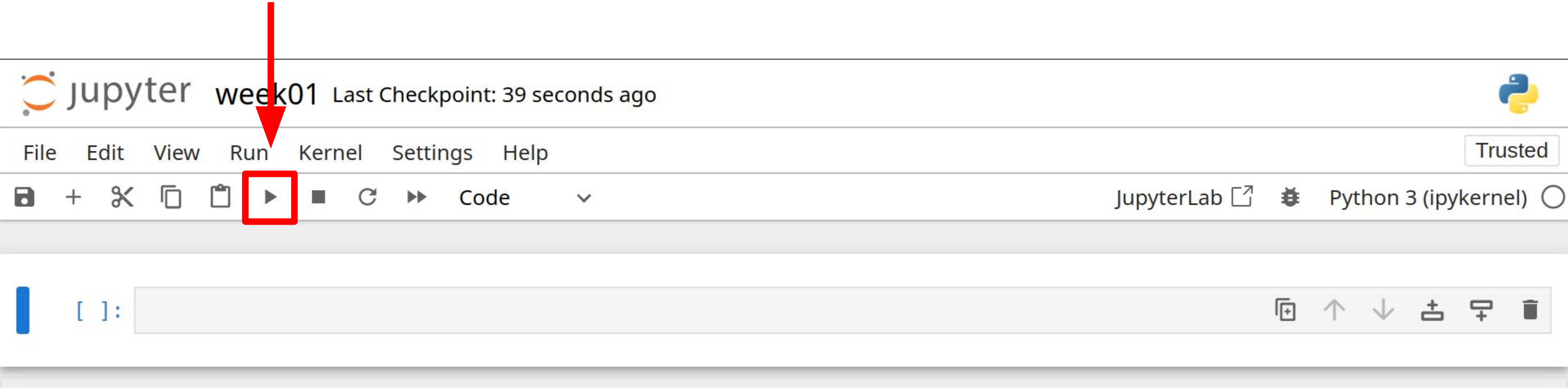
Jupyter Notebook Setup



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



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

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

What is the difference between these two snippets of code?

```python
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")
```

```python
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 < 10:
    print(x)
    x += 1
```

With a **for-loop**:

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

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

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

Recap: 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)
```

Recap: 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)
```

Recap: Accessing List Elements

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

NOTE: List indices start from **0**

index:	**0**	**1**	**2**	**3**	**4**
`nums =`	`[17,`	`28,`	`33,`	`56,`	`6]`
index:	**-5**	**-4**	**-3**	**-2**	**-1**

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

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

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

Recap: 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)
```

Recap: 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)
```

Recap Exercise

Complete the **5.0** program.

Write a program that follows the following steps, what is the output of this program?

1. Create a list `nums` that stores the 3 integers between `22` and `24` (inclusive).
2. Create a list `data` that stores the 2 integers `17` and `55`.
3. Using the `data` list, append 2 additional integers: `20` and `65`
4. Remove the second element of `nums`
5. Insert the integer `34` at Index 1 of `nums`
6. Remove the last element of `data` and insert it at the beginning of `nums`
7. Print `nums` and `data` concatenated in this order (all the values of `nums` first, then all the values of `data`

Exercise 5.0 - Solution

```python
# 1
nums = [22, 23, 24]
# 2
data = [17, 55]
# 3
data.append(20)
data.append(65)
# 4
nums.pop(1)
# 5
nums.insert(1, 34)
# 6
last_elem = data.pop()
nums.insert(0, last_elem)
# 7
print(nums + data)
```

Prints:

```
[65, 22, 34, 24, 17, 55, 20]
```

Additional List Functions

Additional functions that operate on lists

- Get the length of the list: **len**

```
len([4, 8, 10, 12])
```

```
len([-3])
```

```
len([])
```

- Get the max/min elements in a list: **max** and **min**

```
max([4, 8, -2, 0])
```

```
min([4, 8, -2, 0])
```

- Get the sum of all elements in a list: **sum**

```
sum([4, 8, -2, 0])
```

```
sum([-3])
```

Exercise

Complete the **5.1** program.

For the following list `nums` calculate the:

- max value
- min value
- mean value (i.e. the average)

HINT: For the average use a combination of `sum` and `len`

Exercise 5.1 - Solution

```
nums = [4, 8, -17, 23, 55]

max_value = max(nums)
min_value = min(nums)
avg_value = sum(nums) / len(nums)

print(max_value)
print(min_value)
print(avg_value)
```

Dictionaries

Group data together using keys

With **variables**:

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

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

With a **dict**:

```
nums = {"num1": 42, "num2": 100, "num3": 8}

print(nums)
```

Dictionaries

Anatomy of a dictionary:

1. Uses curly brackets **{ }**
2. Elements separated by comma **,**
3. Elements specified with a colon as **key : value**

```
nums = {"num1": 42, "num2": 100,}
```

```
data = {"foo": 8.2, 100: "bar"}
```

Exercise

Complete the **5.2** & **5.3** programs.

- **5.2**: Write a program that creates a new dictionary `letters` containing the three letters `a`, `b`, `c` as keys and assigning them the integer values `1`, `2` and `3`

- **5.3**: Write a program that creates a new dictionary called `pets` which stores the names of my three pets together with their age (as an integer):
  a. `Snowball` is 3 years old
  b. `Flopsie` is 5 years old
  c. `Schnitzel` is 1 year old

Exercise 5.2 - Solution

```python
letters = {"a": 1, "b": 2, "c": 3}

print(letters)
```

Exercise 5.3 - Solution

```python
pets = {"Snowball": 3, "Flopsie": 5, "Schnitzel": 1}

print(pets)
```

Accessing Dictionary Elements

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

NOTE: You can only access keys that exist

```
heights = {"Charles": 175, "Adam": 160, "Florence": 180}
```

```
print(heights["Adam"])
```

```
print(heights["Florence"])
```

ERROR:

```
print(heights["Dan"])
```

Exercise

Complete the **5.4** program.

Write a program that given a dictionary `grades` prints out:

1. The grade of student `Bob`
2. The grade of student `Zethus`
3. The sum of the grades of `Bob` and `Alice`
4. The average of the grades of `Alice`, `Bob` and `Charlie`

Exercise 5.4 - Solution

```python
grades = {"Alice": 9.0, "Bob": 7.5, "Charlie": 8.3, "Zethus": 6.0}

# 1
print(grades["Bob"])
# 2
print(grades["Zethus"])
# 3
print(grades["Bob"] + grades["Alice"])
# 4
print((grades["Alice"] + grades["Bob"] + grades["Charlie"]) / 3)
```

Modifying Dictionaries

You can modify dicts in 2 ways:

```
data = {"a": 42, "b": 3}
```

1. To insert a new element you can use a new key

```
data["c"] = 800
```

```
data["d"] = 4.5
```

2. To modify an existing elements you can assign to the key

```
data["a"] = 10
```

```
data["b"] = 3.2
```

Exercise

Complete the **5.5** program.

Write a program that given a dictionary `scores` does the following:

1. Prints out the score of `Rob`
2. Adds a score of 4 for new user `Dan`
3. Prints the score of `Dan`
4. Replaces the score of `Rob` with the number 6
5. Prints the updated score of `Rob`
6. Prints the final `scores` dictionary

Exercise 5.5 - Solution

```python
scores = {"Rob": 10, "Michelle": 2}

# 1
print(scores["Rob"])
# 2
scores["Dan"] = 4
# 3
print(scores["Dan"])
# 4
scores["Rob"] = 6
# 5
print(scores["Rob"])
# 6
print(scores)
```

Removing Dictionary elements

You can remove elements in a dict with the **del** function.

```
data = {"a": 42, "b": 3}
del data["a"]
print(data)
```

```
data = {"a": 42, "b": 3}
del data["b"]
print(data)
```

```
data = {"a": 42, "b": 3}
del data["a"]
del data["b"]
print(data)
```

Exercise

Complete the **5.6** program.

Write a program that given a dictionary `money` does the following:

1. Deletes the entry for `Rob`
2. Prints the updated `money` dictionary
3. Removes `40` euro from `Dan`
4. Prints the updated `money` dictionary
5. Adds `40` euro to `Adam`
6. Prints the updated `money` dictionary

Exercise 5.6 - Solution

```python
money = {"Adam": 100, "Rob": 200, "Dan": 60}

# 1
del money["Rob"]
# 2
print(money)
# 3
money["Dan"] -= 40
# 4
print(money)
# 5
money["Adam"] += 40
# 6
print(money)
```

Consolidation Exercise

Complete the **5.7** program.

Write a program that given a list of `names`, a matching list of `measurements` and an empty dictionary `heights`:

1. Using a `for` loop add each person's name and their corresponding height into `heights`. For example person `Adam` must have a matching height of 175.
2. Calculate (and print) the sum of the heights of `Adam`, `Dan` and `Rob`
3. Add a new entry in `heights` for `Charlie` who has a height of 190
4. Print the length of `heights`. HINT: You can use the `len` function
5. Print the name of the tallest person. HINT use `max` and use a `for`-loop!
6. Remove the entry in `heights` for `Dan`
7. Print out the final dictionary `heights`

Exercise 5.7 - Solution

```python
names = ["Dan", "Rob", "Adam", "Matt"]
measurements = [140, 165, 155, 142]
heights = {}

# 1
for i in range(len(names)):
   heights[names[i]] = measurements[i]

print(heights)
# 2
print(heights["Adam"] + heights["Dan"] + heights["Rob"])
# 3
heights["Charlie"] = 190
# 4
print(len(heights))
# 5
max_height = max(measurements)
for i in range(len(names)):
   name = names[i]
   if heights[name] == max_height:
       print(name)
# 6
del heights["Dan"]
# 7
print(heights)
```

End of Class

See you all next week!