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

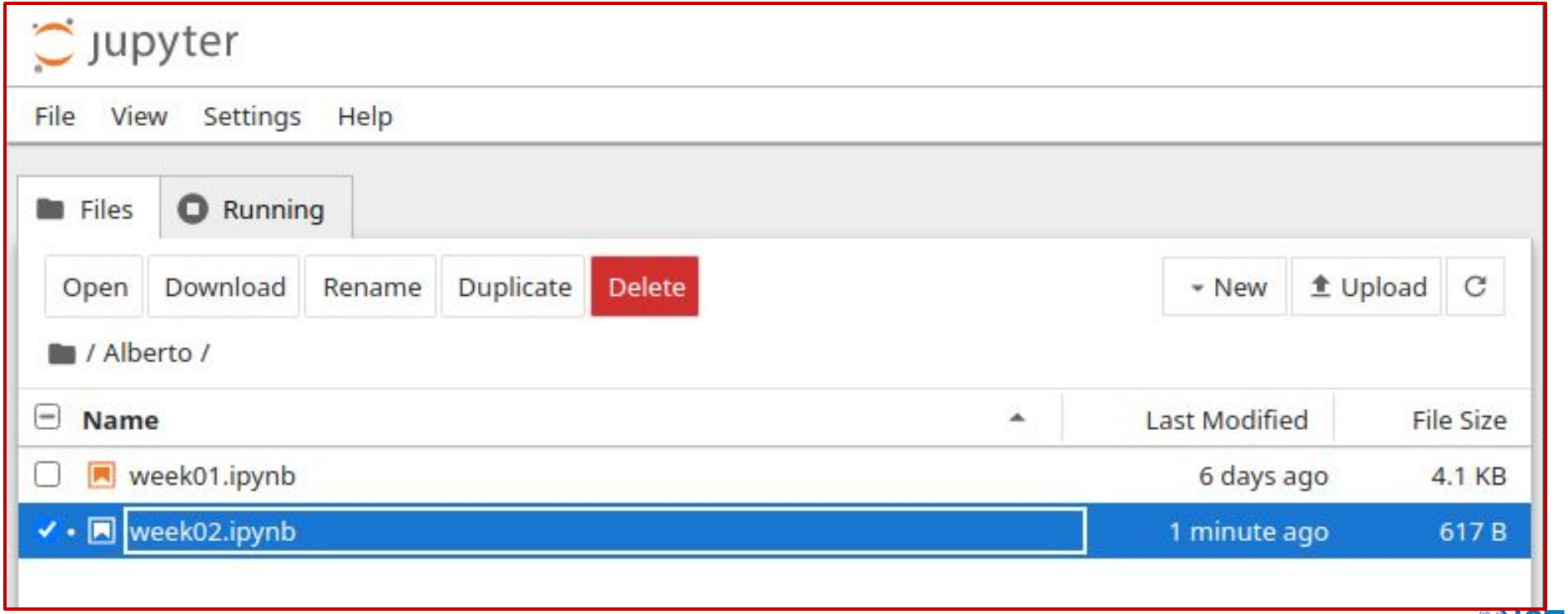
Jupyter Notebook Setup

In a browser:

```
192.168.10.4:8888
```

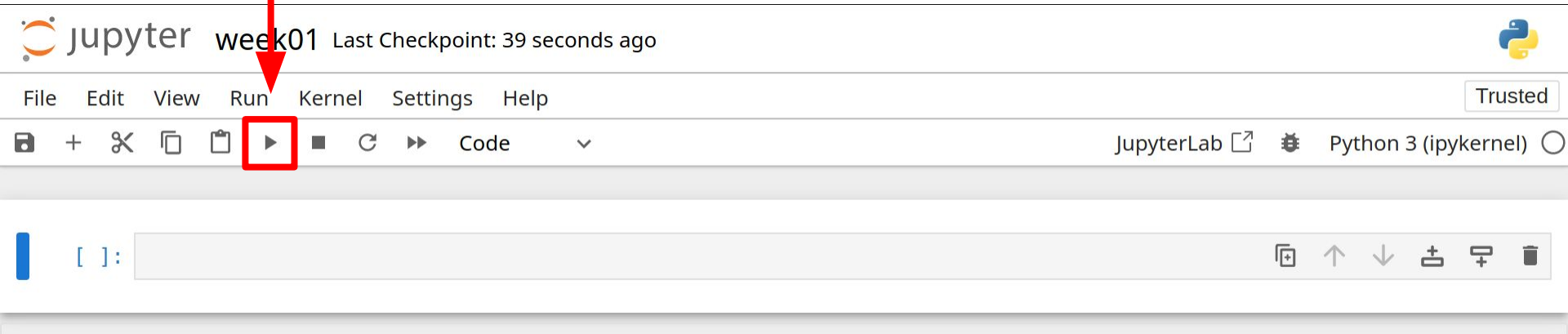
Password: **ist**

Jupyter Notebook Setup



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?

```
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 <u>specific</u> 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: 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])
```

Recap Exercise

Complete the **6.0** program.

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

1. Create a list `nums` that stores the 2 floats `4.5` and `0.2` (inclusive).
2. Prints the length of the list `nums`
3. Append 2 additional floats to `nums`: `1.1` and `20`
4. Remove the second element of `nums` and add it as the last element
5. Print the updated list `nums`
6. Insert the float `3.4` at Index 1 of `nums`
7. Print the updated list `nums`

Exercise 6.0 - Solution

```
# 1
nums = [4.5, 0.2]
# 2
print(nums)
# 3
nums.append(1.1)
nums.append(20)
# 4
elem = nums.pop(1)
nums.append(elem)
# 5
print(nums)
# 6
nums.insert(1, 3.4)
# 7
print(nums)
```

Prints:

```
[4.5, 3.4, 1.1, 20, 0.2]
```

Recap: 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)
```

Recap: 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"])
```

Recap: 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
```

Recap: 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)
```

Recap Exercise

Complete the **6.1** program.

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

1. Prints the balance for `Rob` (i.e. prints how much money `Rob` has)
2. Transfers £50 from `Rob` to `Dan` (i.e removes £50 from `Rob` and adds £50 to `Dan`)
3. Prints the updated balance for `Rob` and `Dan`
4. Adds a new user `Pat` with a balance of £2
5. Removes user `Adam` from the bank (thus removing Adam's balance)
6. Print the dictionary `bank` to show the outcome of all operations

Exercise 6.1 - Solution

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

# 1
print(bank["Rob"])
# 2
bank["Rob"] -= 50
bank["Dan"] += 50
# 3
print(bank["Rob"])
print(bank["Dan"])
# 4
bank["Pat"] = 2
# 5
del bank["Adam"]
# 6
print(bank)
```

Prints:

```
{'Rob': 150, 'Dan': 110, 'Pat': 2}
```

Iterating Lists

Python provides multiple ways to **iterate over lists**.

The most used methodologies are:

Index-iteration:

```
nums = [10, 20, 30, 40]
for i in range(len(nums)):
    print(nums[i])
```

For-each loop:

```
nums = [10, 20, 30, 40]
for num in nums:
    print(num)
```

The output of the two snippets is identical

Exercise

Complete the **6.2** program.

Write a program that given a list of `items` does the following:

1. Prints out every item in the list using **index-based iteration**
2. Prints out every item in the list using a **for-each loop**

Exercise 6.2 - Solution

```python
items = ["apple", "banana", "cherry"]

# 1
for i in range(len(items)):
  print(items[i])

# 2
for item in items:
  print(item)
```

Iterating Dictionaries

Python provides multiple ways to **iterate over dicts**.

The most used methodologies are:

Key-iteration:

```
data = {"a": 4, "f": 1, "z": 8}

for key in data:
    value = data[key]
    print(key, value)
```

For-each loop:

```
data = {"a": 4, "f": 1, "z": 8}

for key, value in data.items():
  print(key, value)
```

The output of the two snippets is identical

Exercise

Complete the **6.3** & **6.4** programs.

```
inventory = {"apples": 30, "bananas": 45, "cherries": 25}
```

- **6.3**: Given the dictionary `inventory`, iterate over the dictionary and print each fruit and its quantity in the format:
  `"There are [quantity] [fruit]"`

- **6.4**: Given the dictionary `inventory`, iterate over the dictionary and calculate the total quantity of fruits. Print the total.

Solution 6.3

```
inventory = {"apples": 30, "bananas": 45, "cherries": 25}

for key, value in inventory.items():
   print("There are " + str(value) + " " + key)
```

Solution 6.4

```python
inventory = {"apples": 30, "bananas": 45, "cherries": 25}

total = 0

for key, value in inventory.items():
    total += value

print(total)
```

Sets

Unordered collections of unique elements

With **variables**:

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

print(num1)
print(num2)

if (num3 != num1) and (num3 != num2):
    print(num3)
```

With a **list**:

```
nums = {42, 100, 42}

print(nums)
```

Sets

Anatomy of a set:

1. Uses curly brackets **{ }**
2. Elements separated by comma **,**
3. Can take any values (will remove duplicates)

```
nums = {42, 100, 42}
```

```
data = {"A", "C", "D"}
```

Modifying Sets

Adding new elements:

1. To insert an element: **add**
2. To remove an element: **remove**

```
nums = {42, 100}

nums.add(8)
nums.remove(100)
nums.add(50)

print(nums)
```

Exercise

Complete the **6.5** & **6.6** programs.

- **6.5**: Write a program that:
  1. Create a set `numbers` with integers 1, 2, and 3.
  2. Add the number 4 to the set.
  3. Remove the number 2 from the set.
  4. Prints the final set `numbers` after all changes

- **6.6**: Write a program that given a set `data` uses a for-loop to add every element of a list `nums` to `data`

Solution 6.5

```python
# 1
numbers = {1, 2, 3}

# 2
numbers.add(4)

# 3
numbers.remove(2)

# 4
print(numbers)
```

Solution 6.6

```
data = {4, 8, 2}

nums = [5, 6, 7, 9]

for num in nums:
    data.add(num)

print(data)
```

Modifying Sets

Set theory operations:

```
set1 = {"A", "B", "C"}
set2 = {"B", "C", "D"}
```

1. Union: **set1 | set2** `{"A", "B", "C", "D"}`
2. Intersection: **set1 & set2** `{"B", "C"}`
3. Difference: **set1 - set2** `{"A"}`

Exercise

Complete the **6.7** program.

Given three schools: `ist`, `ism` and `isb`, write a program that:

1. Finds students who are in both `ist` and `ism`.
2. Finds students who are in `ism` but not in `isb`.
3. Finds students who are in any one of the three schools (union).
4. Finds students who are in all three schools.

Solution 6.7

```
ist_students = {"Alice", "Bob", "Charlie"}
ism_students = {"Bob", "Daisy", "Charlie"}
isb_students = {"Charlie", "Fiona", "Ethan"}

# 1
both_ist_ism = ist_students & ism_students
# 2
ism_not_isb = ism_students - isb_students
# 3
any_school = ist_students | ism_students | isb_students
# 4
all_schools = ist_students & ism_students & isb_students
```

End of Class

See you all next week!