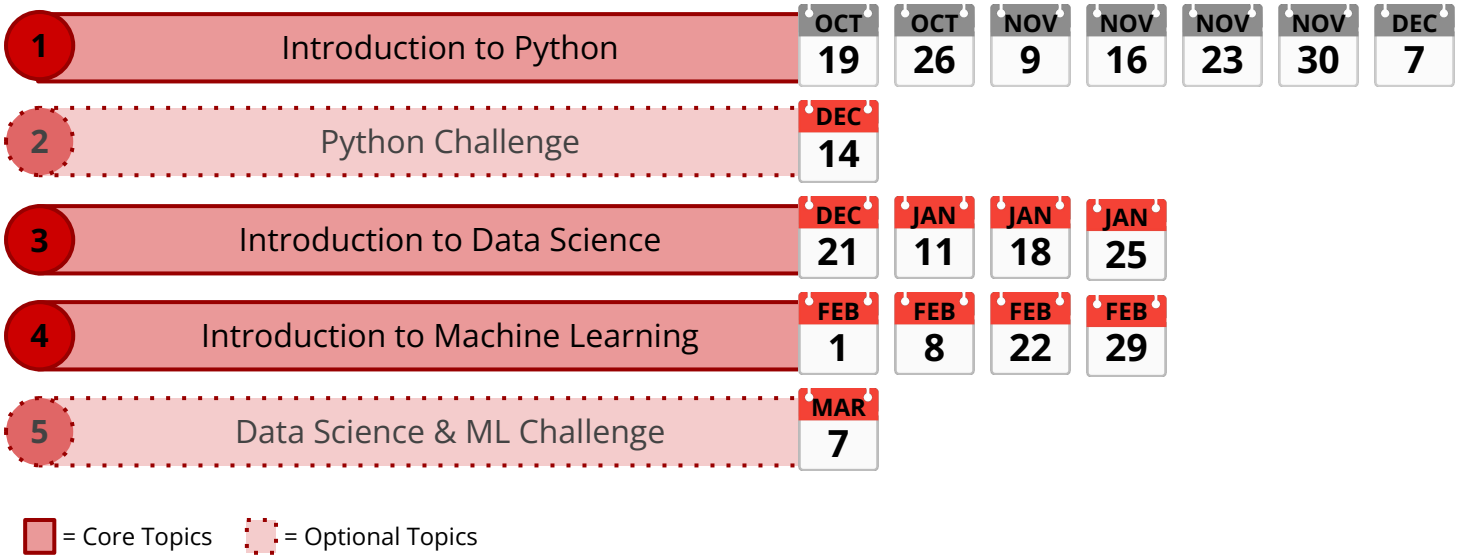
Python for Data Science and Machine Learning

School Year 2023-2024

IST

Course Structure



Jupyter Notebook Setup

In a browser:

```
192.168.10.4:8888
```

Password: **ist**

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 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**:

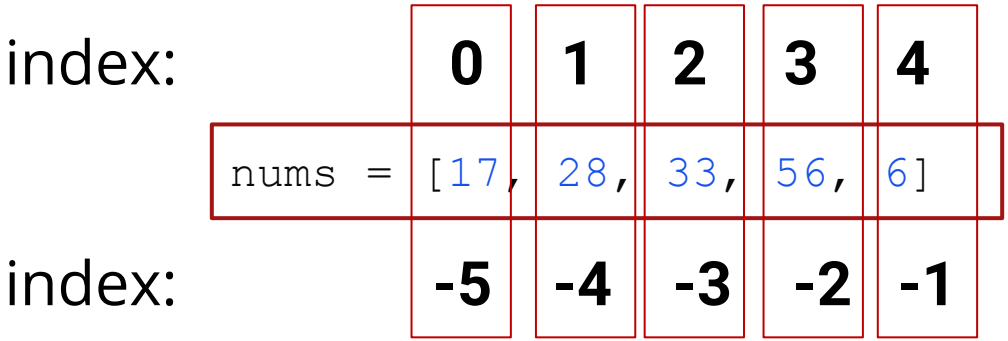
```
nums = [42, 100, 8]

print(nums)
```

Recap: Accessing List Elements

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

NOTE: List indices start from **0**




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

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

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

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)
elem = nums.pop(1)

print(nums)
```

Removing elements:

1. To an element: **pop**

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

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: 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

Recap: Dictionaries

Group data together using keys

With **variables**:

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

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

With a **dict**:

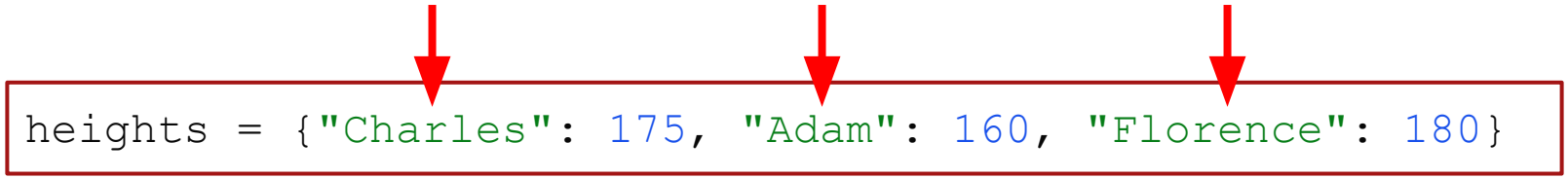
```
nums = {"num1": 42, "num2": 100, "num3": 8}

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

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

1. To insert 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
```

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

```
del data["a"]
```

```
del data["c"]
```

Recap: 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

Recap: 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)
```

Recap: Anatomy of a Set

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"}
```

Recap: 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)
```

Recap: Set Theory

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"}`

Recap: Iterating Sets

Python provides one way to **iterate over sets**.

This makes set and list iteration very similar:

For-each loop:

```
nums = {40, 10, 30, 20}
for num in nums:
    print(num)
```

Remember sets are unordered (so no ordering guarantees!)

Recap: Data-Structure Membership

You can use the **in** keyword to check if an element is in a given data structure. This applies to **lists**, **sets** and **dictionaries**.

```
data1 = ["a", "b", "c"]
x = "b"

print(x in data1)
```

```
data2 = {"a", "b", "c"}
y = "b"

print(y in data2)
```

```
data3 = {"a": 10, "b": 20}
z = "b"

print(z in data3)
```

Functions

Repeatable snippets of code

With **variables**:

```
num1 = 42
num2 = 10

x = num1 + 100
y = num2 + 100
```

With a **function**:

```
def add_100(a):
    return a + 100


num1 = 42
num2 = 10

x = add_100(num1)
y = add_100(num2)
```

Anatomy of a Function

Anatomy of a function:

1. Begins with the **def** keyword
2. Arguments are in brackets **( )** separated by comma **,**
3. Uses the **return** keyword to give output

```
def add_100(a):
    return a + 100

add_100(42)
```

```
def multiply(a, b):
    return a * b

multiply(4, 5)
```

Calling a Function

To call a function you must use the **function name** followed by all the **parameters** within **brackets**.

```
def is_even(n):
    return n % 2 == 0
```

```
x = is_even(2)
y = is_even(5)

print(x)
print(y)
```

```
def create_list(a, b, c):
  return [a, b, c]
```

```
list1 = create_list(1, 2, 3)
list2 = create_list(4, 5, 6)
```

Calling a Function

Functions are not required to take arguments.

```
def create_list():
    my_list = []
    for i in range(1, 4):
        my_list.append(i)
    return my_list
```

```
data1 = create_list()
data1.append(50)

data2 = create_list()

print(data1)
print(data2)
```

Exercise

Complete the **8.0** , **8.1** & **8.2** programs.

- **8.0**: Create a function `multiply_by_five` that takes an integer as an argument and returns the product of that integer and 5.

- **8.1**: Write a function `add_two_numbers` that takes two numbers as arguments and returns their sum.

- **8.2**: Develop a function `is_odd` that takes an integer as input and returns `True` if the number is odd, and `False` otherwise.

Exercise

Complete the **8.3** , **8.4** & **8.5** programs.

- **8.3**: Write a function `max_of_two` that takes two numbers as arguments and returns the larger of the two.

- **8.4**: Create a function `square_number` that takes an integer as an argument and returns the square of that number.

- **8.5**: Write a function `sum_of_list` that takes a list of integers as an argument and returns the sum numbers in the list.

  NOTE: You *must* use a `for-each` loop for this exercise

Competition Time!

In a browser:

```
192.168.10.4:8421
```

Username: **<team-color>**

End of Class

See you all next week!