

PYTHON

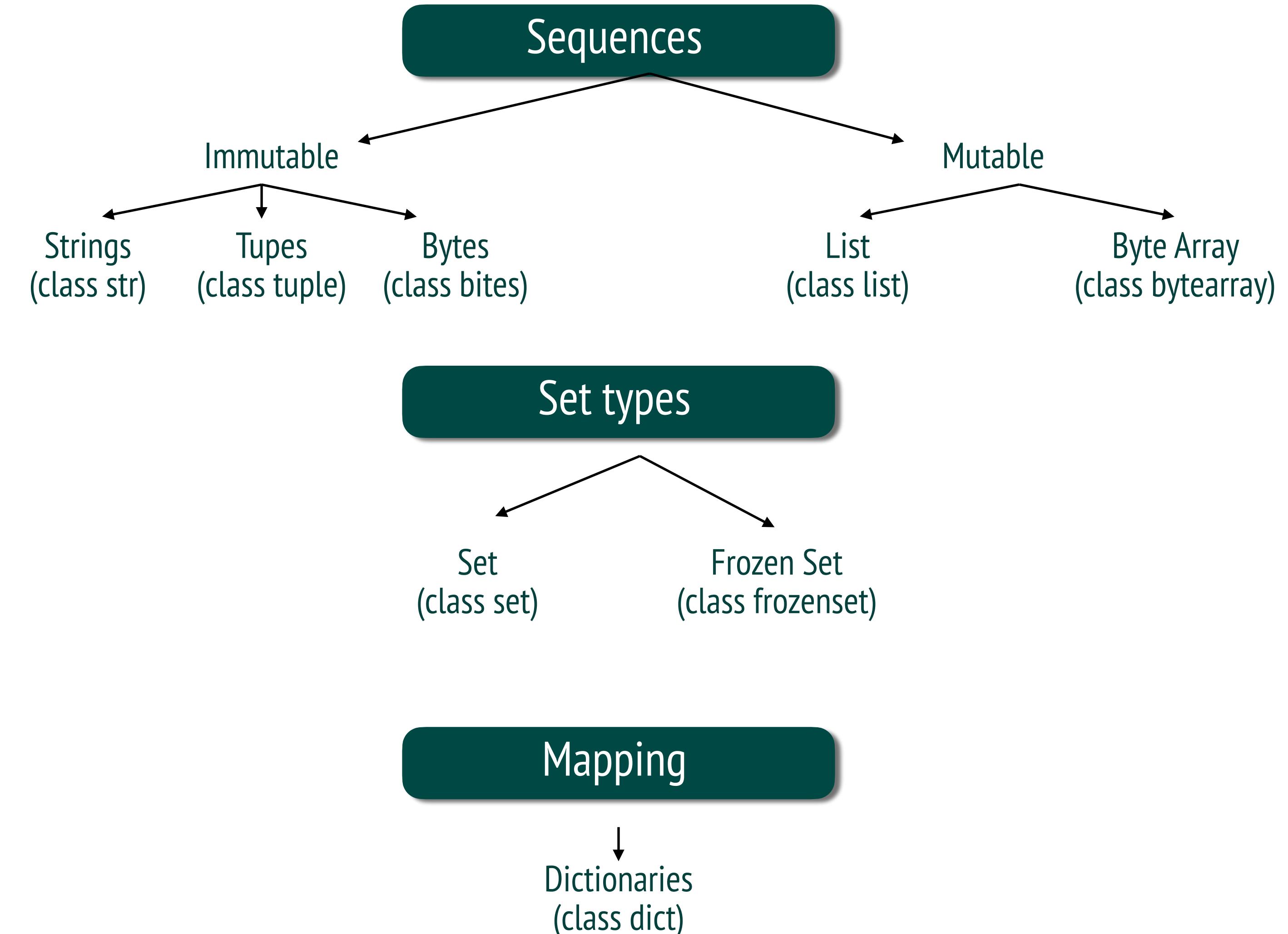
DATA TYPES

Tien-Lam Pham
Anh-Tuan Nguyen
Phenikaa School of Computing

OUTLINE

- List
- String
- Set
- Tuple
- Dictionary

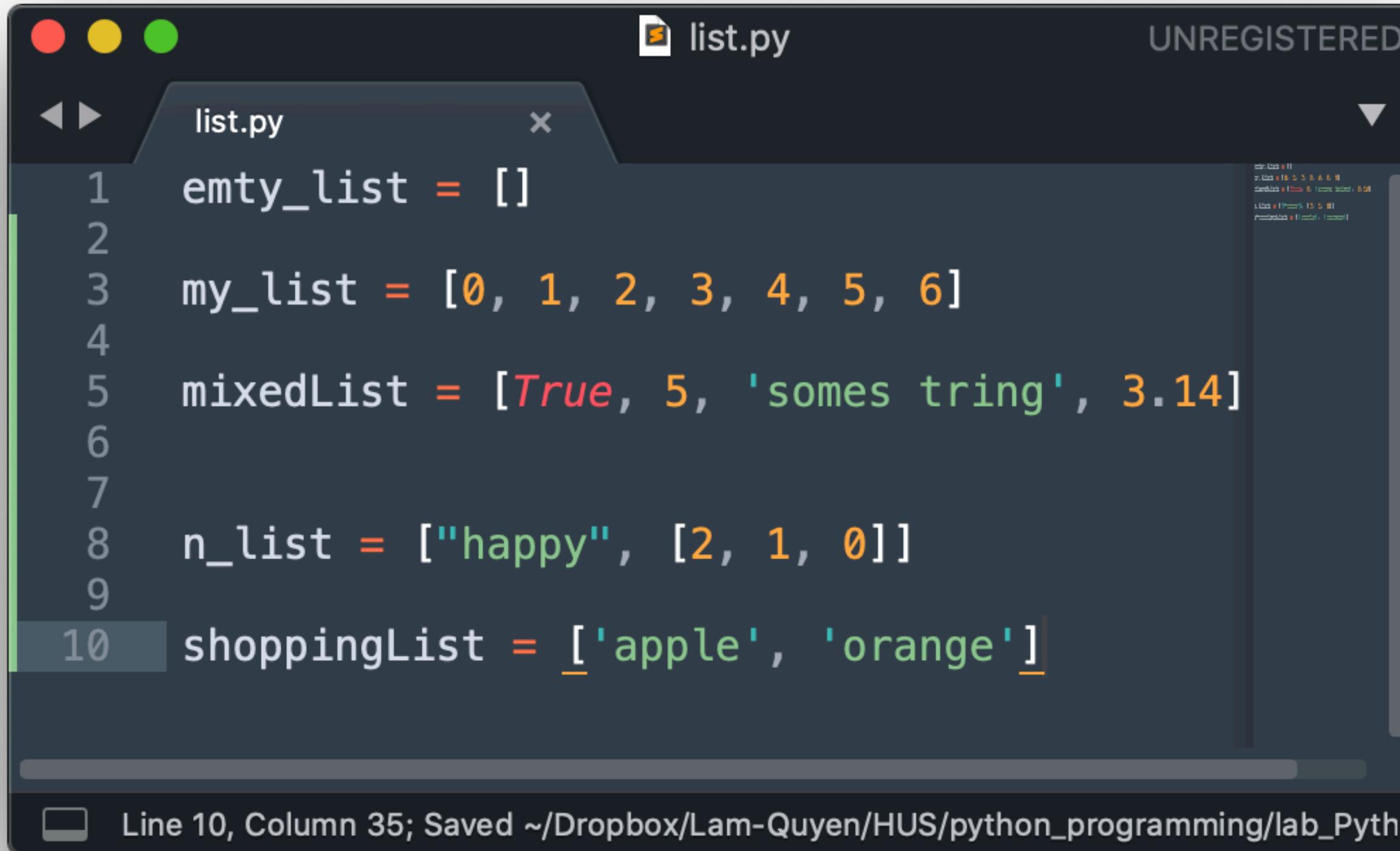
DATA TYPES



LIST

- A container that can contain elements
- Lists are used to store multiple items in a single variable.

```
list_name = [element_1, ..., element_n]
```



A screenshot of a code editor window titled "list.py". The code defines several lists:

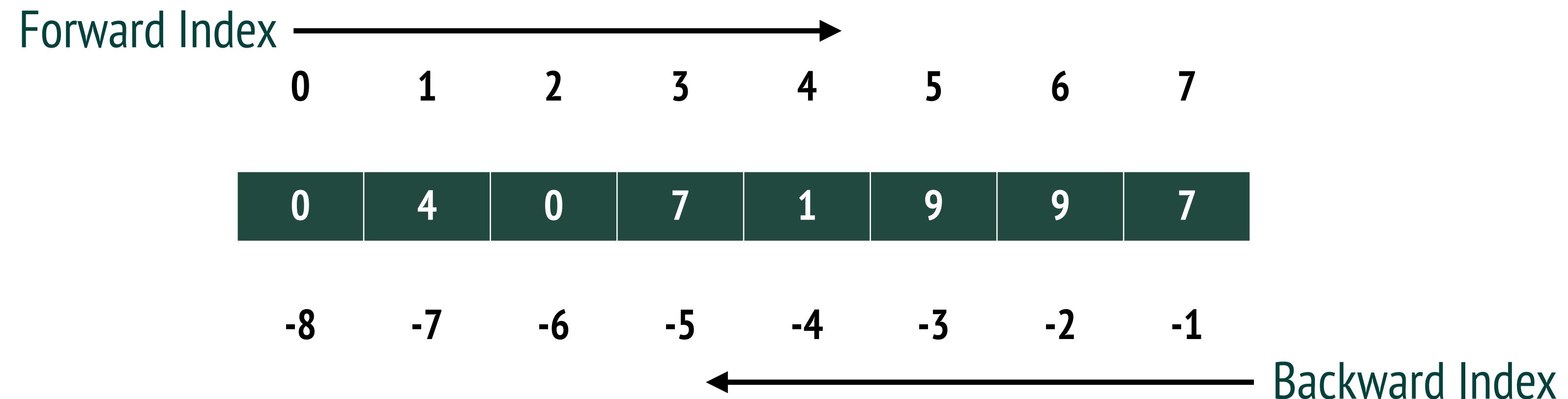
```
empty_list = []
my_list = [0, 1, 2, 3, 4, 5, 6]
mixedList = [True, 5, 'some string', 3.14]
n_list = ["happy", [2, 1, 0]]
shoppingList = ['apple', 'orange']
```

The "shoppingList" assignment is highlighted in red. The status bar at the bottom shows "Line 10, Column 35; Saved ~/Dropbox/Lam-Quyen/HUS/python_programming/lab_Pyth".

LIST

- Index

```
my_List = [0, 4, 0, 7, 1, 9, 9, 7]
```



```
my_List[0] -> 0
```

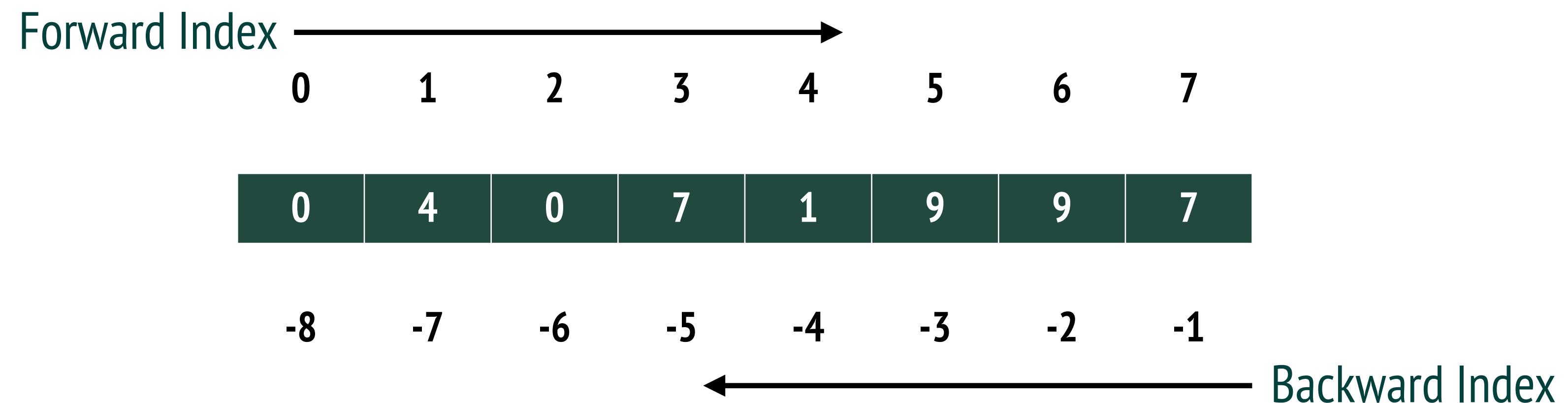
```
my_List[3] -> 7
```

```
my_List[-4] -> 1
```

LIST

- **Slicing**

```
my_List = [0, 4, 0, 7, 1, 9, 9, 7]
```



“:” get all the elements

“a:b” get the elements from a^{th} to $(b^{th}-1)$

my_List [:3] → [0, 4, 0]

my_List [2:4] → [0, 7]

my_List [6:] → [9, 7]

LIST

- + and * operator

```
data1 = [0, 4, 0, 7]
```

```
data2 = [1, 9, 9, 7]
```

```
data = data1 + data2
```

```
>> [0, 4, 0, 7, 1, 9, 9, 7]
```

```
data3 = [0, 4]
```

```
data = data3 * 3
```

```
>> [0, 4, 0, 4, 0, 4]
```

LIST

- `sort()`

```
data = [0, 4, 0, 7, 1, 9, 9, 7]
```

```
data.sort()
```

```
>> [0, 0, 1, 4, 7, 7, 9, 9]
```

```
data = [0, 4, 0, 7, 1, 9, 9, 7]
```

```
data.sort(reverse = True)
```

```
>> [9, 9, 7, 7, 4, 1, 0, 0]
```

LIST

- Delete an element

```
data = [0, 4, 0, 7, 1, 9, 9, 7]
```

```
data.pop(2) #index
```

```
>> [0, 4, 7, 1, 9, 9, 7]
```

```
data = [0, 4, 0, 7, 1, 9, 9, 7]
```

```
data.remove(7) #the first element
```

```
>> [0, 4, 0, 1, 9, 9, 7]
```

LIST

- Delete elements

```
data = [0, 4, 0, 7, 1, 9, 9, 7]
```

```
del data[1:3]
```

```
>> [0, 7, 1, 9, 9, 7]
```

```
data = [0, 4, 0, 7, 1, 9, 9, 7]
```

```
data.clear()
```

```
>> []
```

LIST

- `index()`, `reverse()`, `count()`, `copy()`

```
data = [0, 4, 0, 7, 1, 9, 9, 7]
```

```
data.index(7)
```

```
>> 3
```

```
data = [0, 4, 0, 7, 1, 9, 9, 7]
```

```
data.reverse()
```

```
>> [9, 9, 7, 7, 4, 1, 0, 0]
```

LIST

- `index()`, `reverse()`, `count()`, `copy()`

```
data = [0, 4, 0, 7, 1, 9, 9, 7]
```

```
data.count(7)
```

```
>> 2
```

```
data = [0, 4, 0, 7, 1, 9, 9, 7]
```

```
data_copy = data.copy()
```

```
data_copy
```

```
>> [0, 4, 0, 7, 1, 9, 9, 7]
```



STRING

The image shows a Mac desktop environment with two windows open. On the left is a code editor window titled "my_str.py" containing Python code. On the right is a terminal window titled "ex -- -zsh -- 53x13" showing the output of running the script.

Code Editor (my_str.py):

```
my_string = "Hello Phenikaa"  
print(type(my_string))  
print(my_string)  
  
print("My name is Quyen")
```

Terminal Output:

```
[base] mac@Quyens-MacBook-Pro ex % python my_str.py  
<class 'str'>  
Hello Phenikaa  
My name is Quyen  
(base) mac@Quyens-MacBook-Pro ex %
```

STRING

- Specific characters in string

Escape character	Result	Description
\'	'	Single quote
\"	"	Double quote
\	\	Backslash

Code	Result
\n	New Line
\r	Carriage Return
\t	Tab
\b	Backspace
\f	Form Feed

STRING

- Insert to string

The screenshot shows a Jupyter Notebook interface running on localhost. The top bar includes a navigation menu with links like WiDs, PhD, DeadLine, scrap, Environments, MIE, My-SQL, Jp, Street style, Passion, Scholarship, Machine learning, Bike & Accessory, Your Projects..., LaTeX Editor, and a Stack Overflow link. The main area displays four code cells:

- In [17]:**

```
text = "Phenikaa"
print("Hello", text)
```

Hello Phenikaa
- In [18]:**

```
text = "Phuong"
print("Hello %s. Have a nice day" %(text))
```

Hello Phuong. Have a nice day
- In [19]:**

```
text1 = "Hello"
text2 = "Phenikaa"
print("{n1}, {n2}".format(n1 = text1, n2 = text2))
```

Hello, Phenikaa
- In [20]:**

```
name = "Tuan"
age = 21
print(f"Hello {name}, are you {age} years old?")
```

Hello Tuan, are you 21 years old?

STRING

- + and * operator

The screenshot shows a Jupyter Notebook interface running on localhost. The title bar indicates it's a Python - Jupyter Notebook. The top right corner shows a Python logo icon and a 'Logout' button. The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, and a 'Trusted' status indicator. Below the menu is a toolbar with icons for file operations like save, new, and delete, and code execution controls like run, cell up, cell down, and cell next.

In [22]:

```
s1 = "I love "
s2 = "Python!"

s3 = s1 + s2
s4 = s3*2

print(s3)
print(s4)
```

I love Python!
I love Python!I love Python!

In []:

In []:

STRING

- + and * operator

The screenshot shows a Jupyter Notebook interface running on localhost. The top bar includes tabs for WiDs, PhD, DeadLine, scrap, Environments, MIE, My-SQL, Jp, Street style, Passion, Scholarship, Machine learning, Bike & Accessory, Your Projects..., LaTeX Editor, and a Stack Overflow link. The main area displays the following Python code:

```
In [22]: s1 = "I love "
s2 = "Python!"

s3 = s1 + s2
s4 = s3*2

print(s3)
print(s4)
```

The output below the code cell shows the results of the execution:

```
I love Python!
I love Python!I love Python!
```

Two input cells are visible at the bottom, labeled In []:.

STRING

```
#isdigit(): Kiểm tra xem string  
# gồm các kí tự số  
print("10".isdigit())  
print("abc".isdigit())
```

True
False

```
#isalpha(): Kiểm tra xem string  
# chỉ được tạo từ các kí tự chữ cái  
print("10".isalpha())  
print("abc".isalpha())
```

False
True

```
#islower(): Kiểm tra xem string  
# với tất cả các kí tự ở dạng chữ thường  
print("ab".islower())  
print("Ab".islower())
```

True
False

```
#endswith(): Kiểm tra phần kết thúc của một string  
#startswith(): Kiểm tra phần đầu của một string  
my_string = "Đây là trường Phenikaa"  
print(my_string.endswith("Phenikaa"))  
print(my_string.startswith("trường"))
```

True
False

```
#isupper(): Kiểm tra xem string  
# với tất cả các kí tự ở dạng chữ thường  
print("Ab".isupper())  
print("AB".isupper())
```

False
True

```
#istitle(): Kiểm tra xem string  
# có bắt đầu bằng chữ in hoa  
print("hello".istitle())  
print("Hello".istitle())  
print("HELLO".istitle())
```

False
True
False

```
#isspace(): Kiểm tra xem string  
# chỉ là khoảng trắng  
print(" ".isspace())  
print("  ".isspace())  
print("  ".isspace())  
print("PKA".isspace())
```

False
True
True
False

STRING

- modify string

title()	Chuyển đổi kí tự đầu từng từ (word) trong một chuỗi thành kí tự hoa
capitalize()	Chuyển đổi chữ cái đầu tiên của một chuỗi thành chữ hoa và các kí tự sau thành chữ thường
swapcase()	Chuyển đổi các kí tự từ chữ thường sang chữ hoa và ngược lại
upper()	Chuyển đổi tất cả các kí tự trong chuỗi thành chữ hoa
lower()	Chuyển đổi các kí tự sang chữ thường
center()	Chỉnh chuỗi ở trung tâm, và chiều dài của chuỗi là

STRING

- usefull function

```
#count() để đếm số kí tự xuất hiện trong một string,  
#len() để tính chiều dài của một string  
my_str = "Hello, Nice to meet you"  
print(len(my_str))  
print(my_str.count("e"))
```

23
4

```
#strip(): Loại bỏ khoảng trắng ở cả hai đầu của chuỗi  
my_string = " Đây là trường Phenikaa "  
my_string.strip()
```

'Đây là trường Phenikaa'

```
#partition(): Tách chuỗi  
my_string = "Đây là trường Phenikaa"  
my_string.partition("trường")
```

('Đây là ', 'trường', ' Phenikaa')

```
#replace(): Thay thế chuỗi  
my_string = "Đây là trường Phenikaa"  
my_string.replace("Phenikaa", "Đại học Phenikaa")  
  
'Đây là trường Đại học Phenikaa'
```

```
#find(): Tìm vị trí xuất hiện của string s1  
trong string s2. Giá trị -1 được trả về  
#trong trường hợp  
#s1 không được tìm thấy trong s2.  
my_str = "Hello, Nice to meet you"  
print(my_str.find("Nice"))  
print(my_str.find("PKA"))
```

7
-1

TUPLE

```
tuple_name = (element_1, ..., element_n)
```

```
my_tuple = (1, 2, 3)  
print(my_tuple)
```

```
(1, 2, 3)
```

```
my_tuple = 1, 2  
print(my_tuple)
```

```
(1, 2)
```

```
#unpacking  
my_tuple = (1, 2, 3)  
x, y, z = my_tuple  
print(x)  
print(y)  
print(z)
```

```
1  
2  
3
```

- Immutable

```
t = (1, 2, 3, 4, 5)  
t[2] = 9  
print(t)
```

```
TypeError          Traceback (most recent call last)  
<ipython-input-45-b1f85e7d332a> in <module>  
      1 t = (1, 2, 3, 4, 5)  
----> 2 t[2] = 9  
      3 print(t)
```

```
TypeError: 'tuple' object does not support item assignment
```

```
t = (1, 2, [4, 9])  
t[2][1] = 10  
print(t)
```

```
(1, 2, [4, 10])
```

SET

```
set_name = {element_1, ..., element_n}
```

Using curly brackets

```
1 # create a set
2 animals = {"cat", "dog", "tiger"}
3
4 print(type(animals))
5 print(animals)
```

```
<class 'set'>
{'dog', 'cat', 'tiger'}
```

Items with different data types

```
1 # create a set
2 a_set = {"cat", 5, True, 40.0}
3
4 print(type(a_set))
5 print(a_set)
```

```
<class 'set'>
{40.0, 'cat', 5, True}
```

Set comprehension

```
1 # set comprehension
2
3 a_set = {i*i for i in range(10)}
4 print(a_set)
```

```
{0, 1, 64, 4, 36, 9, 16, 49, 81, 25}
```

SET

Add an item

```
1 # add an item
2 animals = {"cat", "dog", "tiger"}
3 animals.add("bear")
4 print(animals)
```

{'dog', 'bear', 'cat', 'tiger'}

Insert a set to another set

```
1 # insert a set to another set
2 animals = {"cat", "dog", "tiger"}
3 animals.update({"chicken", "Duck"})
4 print(animals)
```

{'Duck', 'tiger', 'dog', 'cat', 'chicken'}

Join two sets

```
1 # join two sets
2 set1 = {"cat", "dog"}
3 set2 = {"duck", "tiger"}
4
5 set3 = set1.union(set2)
6 print(set3)
```

{'duck', 'dog', 'cat', 'tiger'}

Not allow duplicate values

```
1 # No duplication
2 animals = {"cat", "dog", "tiger"}
3 print(animals)
4
5 animals.add("cat")
6 print(animals)
```

{'tiger', 'cat', 'dog'}
{'tiger', 'cat', 'dog'}

SET

difference function

```
1 # difference
2
3 set1 = {"apple", "banana", "cherry"}
4 set2 = {"pineapple", "apple"}
5
6 set3 = set1.difference(set2)
7
8 print(set3)
```

{'cherry', 'banana'}

symmetric_difference

```
1 # symmetric_difference
2
3 set1 = {"apple", "banana", "cherry"}
4 set2 = {"pineapple", "apple"}
5
6 set3 = set1.symmetric_difference(set2)
7
8 print(set3)
```

{'pineapple', 'cherry', 'banana'}

difference_update function

```
1 # difference_update
2
3 set1 = {"apple", "banana", "cherry"}
4 set2 = {"pineapple", "apple"}
5
6 set1.difference_update(set2)
7
8 print(set1)
```

{'cherry', 'banana'}

symmetric_difference_update

```
1 # symmetric_difference_update
2
3 set1 = {"apple", "banana", "cherry"}
4 set2 = {"pineapple", "apple"}
5
6 set1.symmetric_difference_update(set2)
7
8 print(set1)
```

{'pineapple', 'cherry', 'banana'}

SET

- Remove an item

remove(item)

Remove an item from the set.

```
1 # remove an item
2 animals = {"cat", "dog", "tiger"}
3 animals.remove("dog")
4 print(animals)
```

```
{'cat', 'tiger'}
```

discard(item)

Remove an item from the set if it is present.

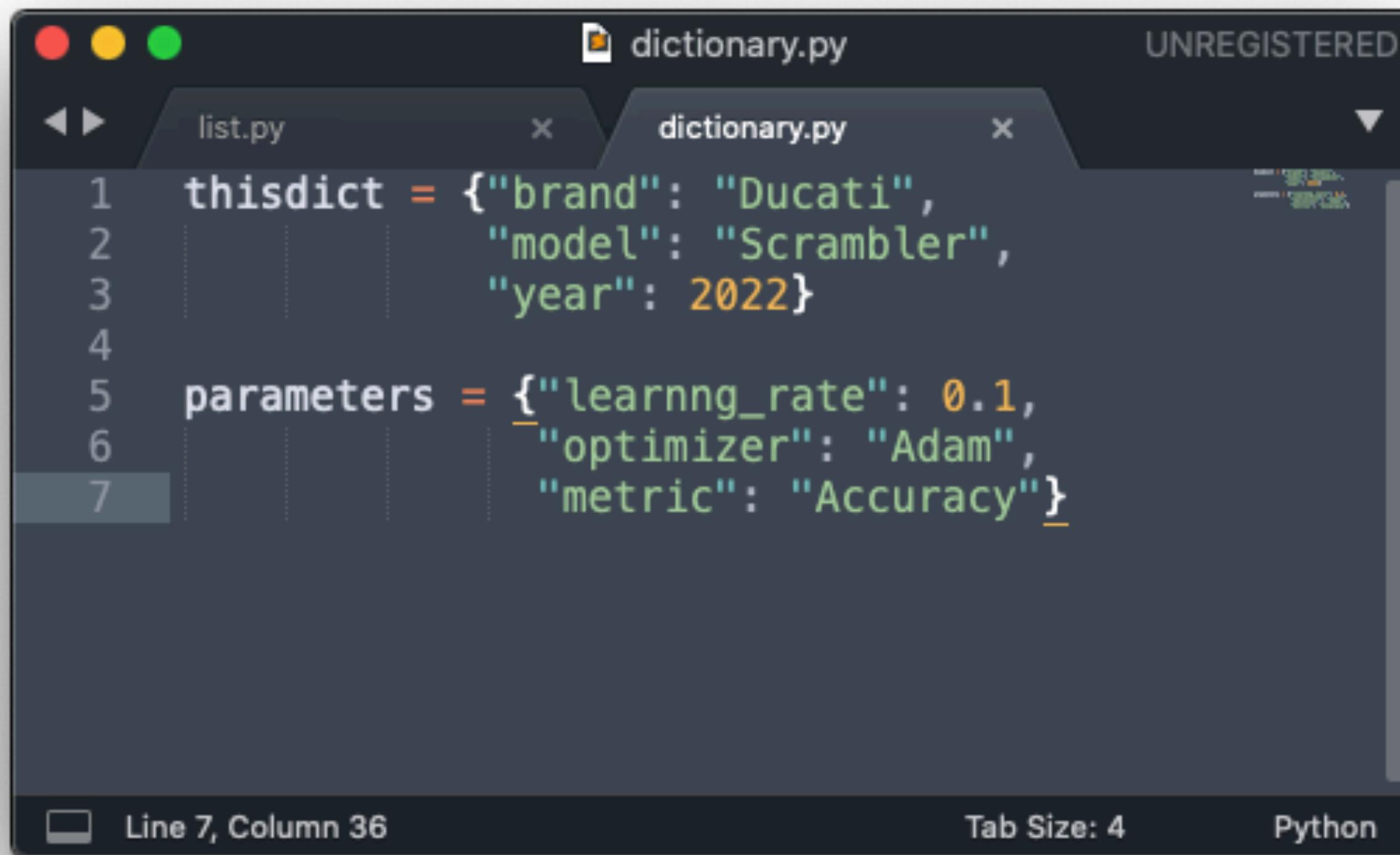
```
1 # remove an item
2 animals = {"cat", "dog", "tiger"}
3 animals.discard("tiger")
4 print(animals)
```

```
{'dog', 'cat'}
```

DICTIONARY

- Dictionaries are used to store data values in key:value pairs.

```
Dictionary_name = {key_1:value_1, ..., key_n, value_n}
```



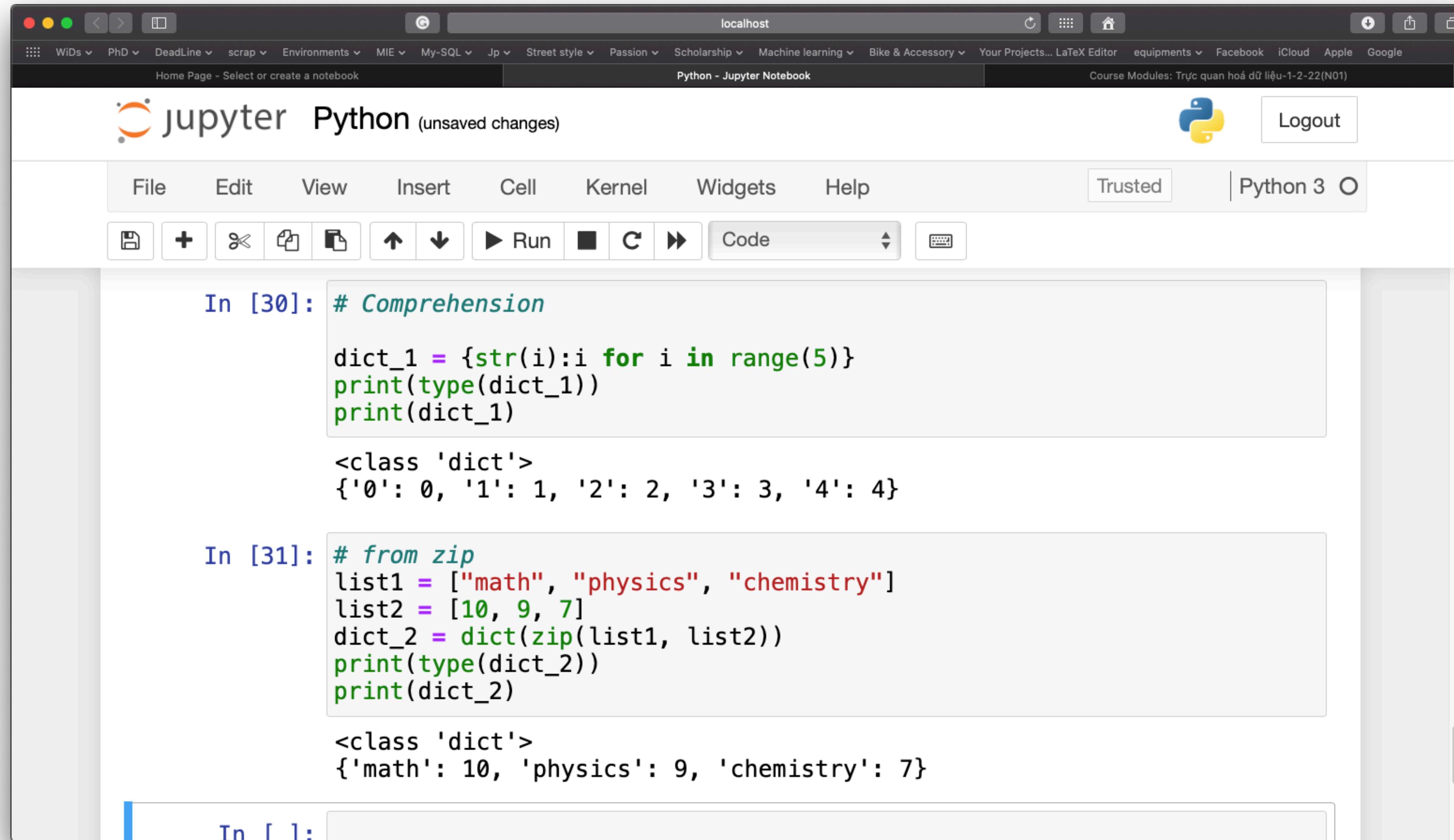
```
thisdict = {"brand": "Ducati",
            "model": "Scrambler",
            "year": 2022}

parameters = {"learning_rate": 0.1,
              "optimizer": "Adam",
              "metric": "Accuracy"}
```

Line 7, Column 36 Tab Size: 4 Python

DICTIONARY

- Create Dictionary



The screenshot shows a Jupyter Notebook interface running on localhost. The top navigation bar includes links for WiDs, PhD, DeadLine, scrap, Environments, MIE, My-SQL, Jp, Street style, Passion, Scholarship, Machine learning, Bike & Accessory, Your Projects, LaTeX Editor, equipments, Facebook, iCloud, Apple, Google, and Course Modules: Trực quan hóa dữ liệu-1-2-22(N01). The main window displays two code cells:

In [30]: # Comprehension

```
dict_1 = {str(i):i for i in range(5)}
print(type(dict_1))
print(dict_1)
```

<class 'dict'>
{'0': 0, '1': 1, '2': 2, '3': 3, '4': 4}

In [31]: # from zip

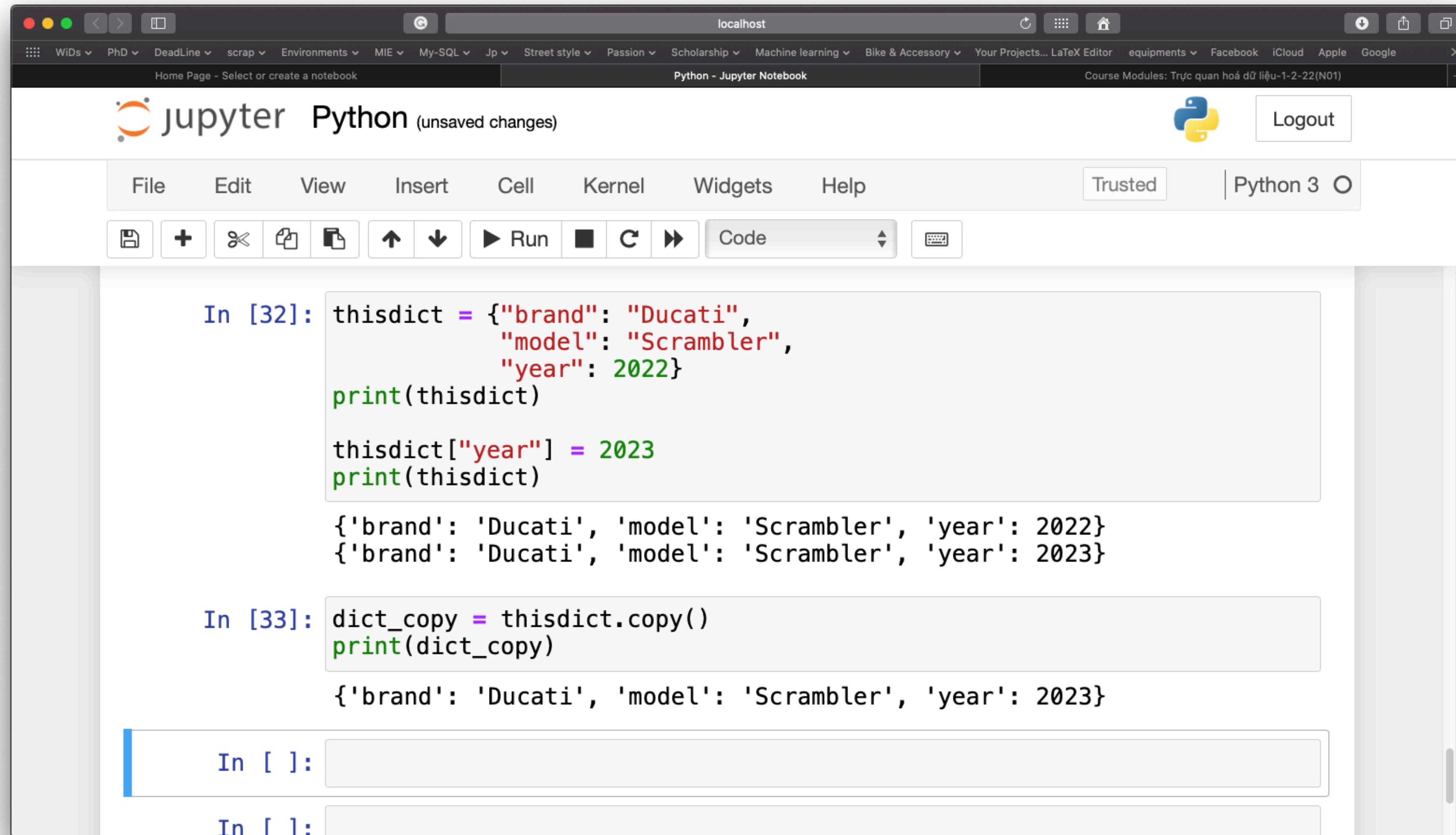
```
list1 = ["math", "physics", "chemistry"]
list2 = [10, 9, 7]
dict_2 = dict(zip(list1, list2))
print(type(dict_2))
print(dict_2)
```

<class 'dict'>
{'math': 10, 'physics': 9, 'chemistry': 7}

In []:

DICTIONARY

- Update value, copy



The screenshot shows a Jupyter Notebook interface running on localhost. The top navigation bar includes links for WiDs, PhD, DeadLine, scrap, Environments, MIE, My-SQL, Jp, Street style, Passion, Scholarship, Machine learning, Bike & Accessory, Your Projects, LaTeX Editor, equipments, Facebook, iCloud, Apple, Google, and Course Modules: Trực quan hóa dữ liệu-1-2-22(N01). The main area displays two code cells:

```
In [32]: thisdict = {"brand": "Ducati",
                  "model": "Scrambler",
                  "year": 2022}
print(thisdict)

thisdict["year"] = 2023
print(thisdict)

{'brand': 'Ducati', 'model': 'Scrambler', 'year': 2022}
{'brand': 'Ducati', 'model': 'Scrambler', 'year': 2023}

In [33]: dict_copy = thisdict.copy()
print(dict_copy)

{'brand': 'Ducati', 'model': 'Scrambler', 'year': 2023}
```

The notebook interface includes a toolbar with file operations like Save, New, and Run, as well as kernel selection (Python 3) and a Trusted button.

DICTIONARY

- Get keys and values

The screenshot shows a Jupyter Notebook interface running on localhost. The notebook has several cells containing Python code and their outputs.

- In [36]:**

```
parameters = {"learning_rate": 0.1,
               "optimizer": "Adam",
               "metric": "Accuracy"}
```
- In [38]:**

```
keys = parameters.keys()
for key in keys:
    print(key)
```

Output:
learning_rate
optimizer
metric
- In [39]:**

```
values = parameters.values()
for value in values:
    print(value)
```

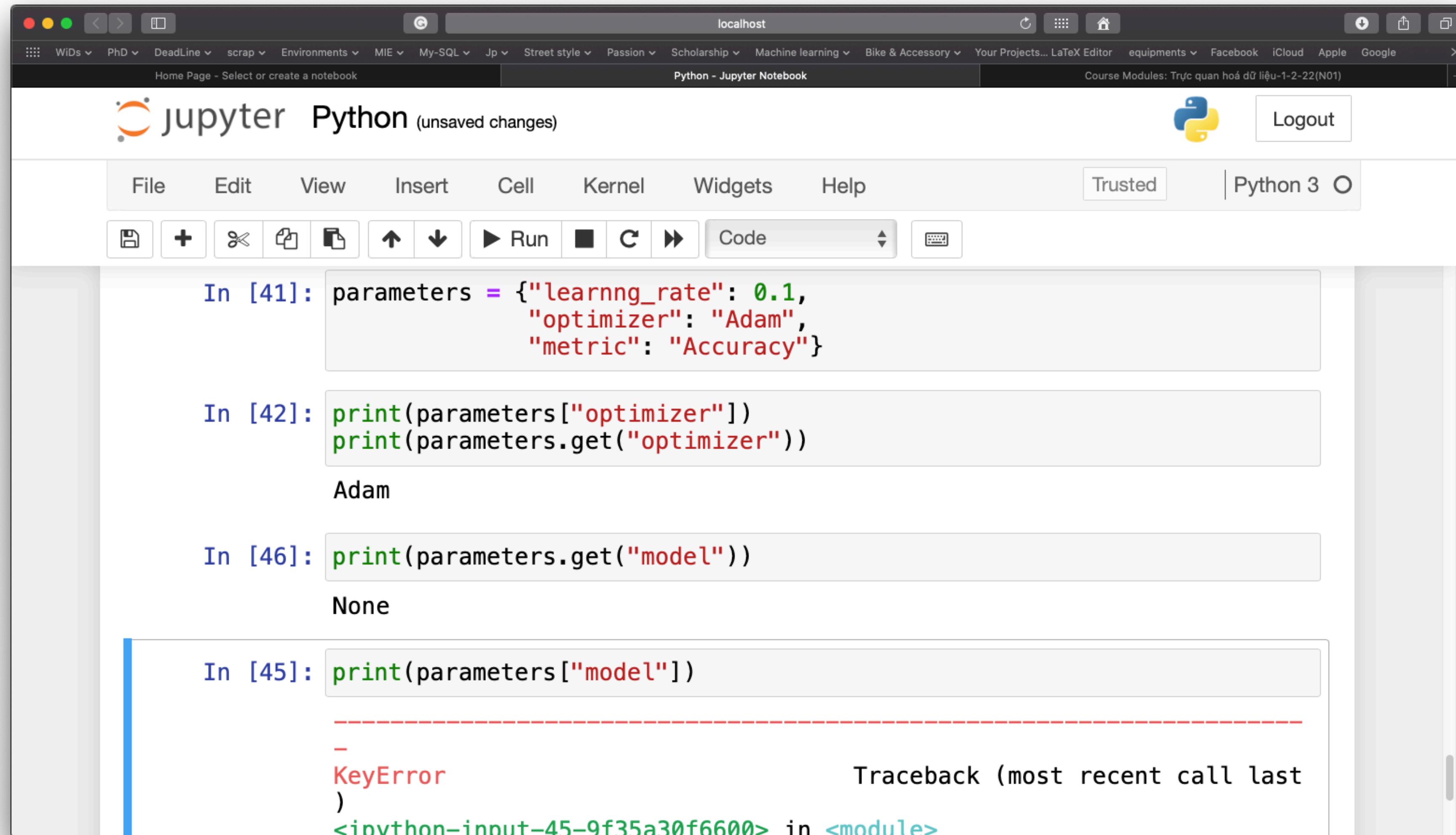
Output:
0.1
Adam
Accuracy
- In [40]:**

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

Output:
learning_rate 0.1
optimizer Adam
metric Accuracy

DICTIONARY

- Get a value via a key



The screenshot shows a Jupyter Notebook interface running on localhost. The notebook has a Python kernel and is set to Python 3. The user has defined a dictionary named 'parameters' in cell 41, which contains keys 'learning_rate', 'optimizer', and 'metric'. In cell 42, they print the value of 'optimizer' using both direct indexing and the get method, both resulting in 'Adam'. In cell 46, they print the value of 'model', which is 'None'. In cell 45, they attempt to print the value of 'model', which triggers a `KeyError`. The error message indicates that the most recent call was from the current module.

```
In [41]: parameters = {"learning_rate": 0.1,  
                      "optimizer": "Adam",  
                      "metric": "Accuracy"}  
  
In [42]: print(parameters["optimizer"])  
print(parameters.get("optimizer"))  
  
Adam  
  
In [46]: print(parameters.get("model"))  
  
None  
  
In [45]: print(parameters["model"])  
  
-----  
-  
KeyError  
)  
<ipython-input-45-9f35a30f6600> in <module>
```

DICTIONARY

- `setdefault()`

The screenshot shows a Jupyter Notebook interface running in a web browser. The title bar indicates it's a Python notebook. The toolbar includes standard options like File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted status, and Python 3 kernel selection. Below the toolbar, there are several icons for file operations (Save, New, Delete, Copy, Paste, Find, Run, Cell, Kernel, Help, Code). The main area contains two code cells:

```
In [48]: parameters = {"learning_rate": 0.1,
                     "optimizer": "Adam",
                     "metric": "Accuracy"}

parameters.setdefault("model", "CNN")

print(parameters)
{'learning_rate': 0.1, 'optimizer': 'Adam', 'metric': 'Accuracy', 'model': 'CNN'}
```

```
In [49]: parameters["n_epoch"] = 1000
print(parameters)

{'learning_rate': 0.1, 'optimizer': 'Adam', 'metric': 'Accuracy', 'model': 'CNN', 'n_epoch': 1000}
```

The first cell runs successfully, and its output is displayed below it. The second cell is partially visible at the bottom.

DICTIONARY

- pop(), popitem(), clear() del()

The screenshot shows a Jupyter Notebook interface running on localhost. The top bar includes tabs for 'Home Page - Select or create a notebook' and 'Python - Jupyter Notebook'. The main area displays two code cells:

In [51]:

```
parameters = {"learning_rate": 0.1,
               "optimizer": "Adam",
               "metric": "Accuracy"}
value = parameters.pop('learning_rate')
print(value)
print(parameters)
```

Output:

```
0.1
{'optimizer': 'Adam', 'metric': 'Accuracy'}
```

In [52]:

```
parameters = {"learning_rate": 0.1,
               "optimizer": "Adam",
               "metric": "Accuracy"}
value = parameters.popitem()
print(value)
print(parameters)
```

Output:

```
('metric', 'Accuracy')
{'learning_rate': 0.1, 'optimizer': 'Adam'}
```

ASSIGNMENT
