PIAS
PHENIKAA INSTITUTE
FOR ADVANCED STUDY

PHENIKAA
UNIVERSITY
Realizing your potential

PYTHON

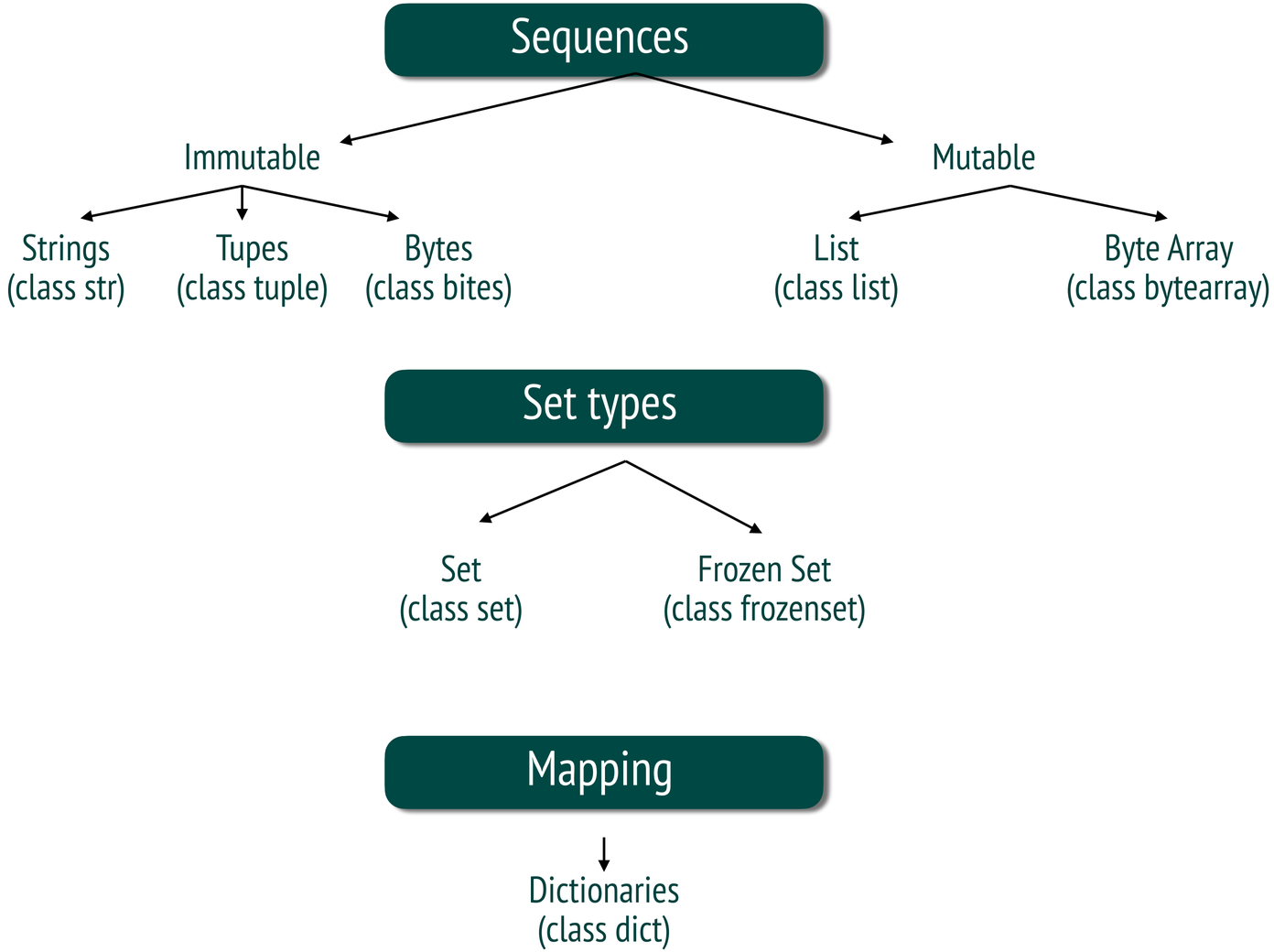
DATA TYPES

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

OUTLINE

- List
- String
- Set
- Tuple
- Dictionary

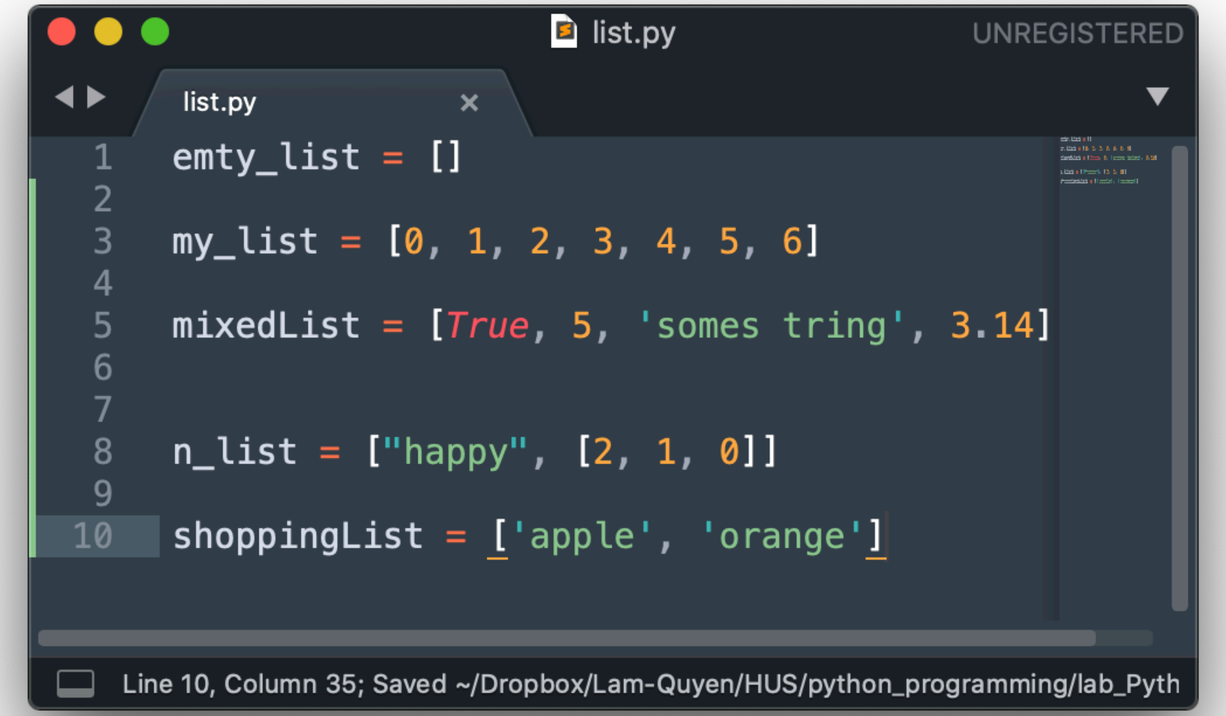
DATA TYPES


Sequences
Immutable
Mutable
Strings
(class str)
Tupes
(class tuple)
Bytes
(class bites)
List
(class list)
Byte Array
(class bytearray)
Set types
Set
(class set)
Frozen Set
(class frozenset)
Mapping
Dictionaries
(class dict)

LIST

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

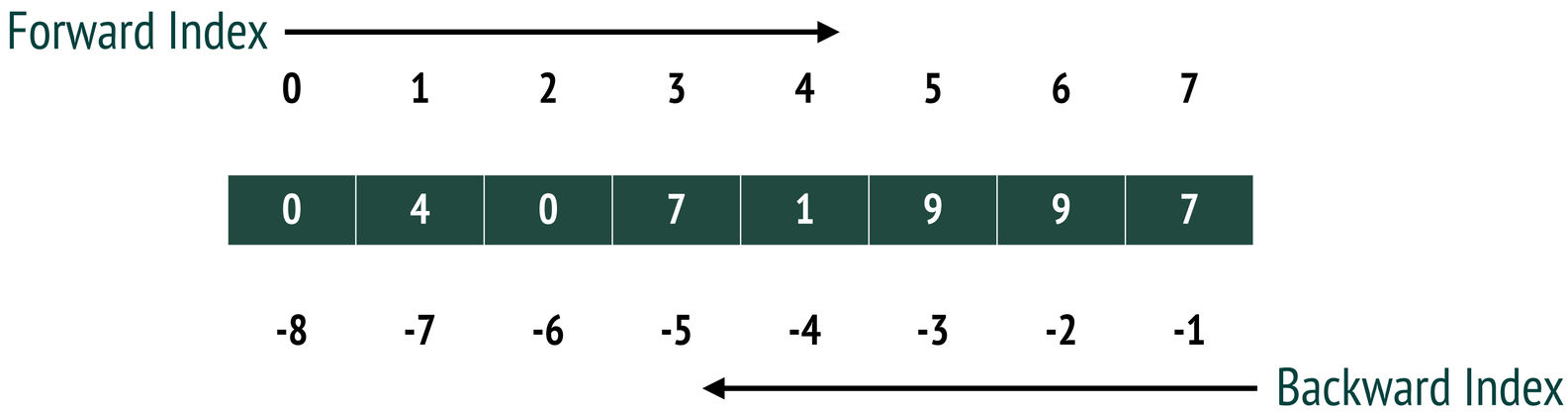
```
list_name = [element_1, …, element_n]
```



LIST

- Index

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

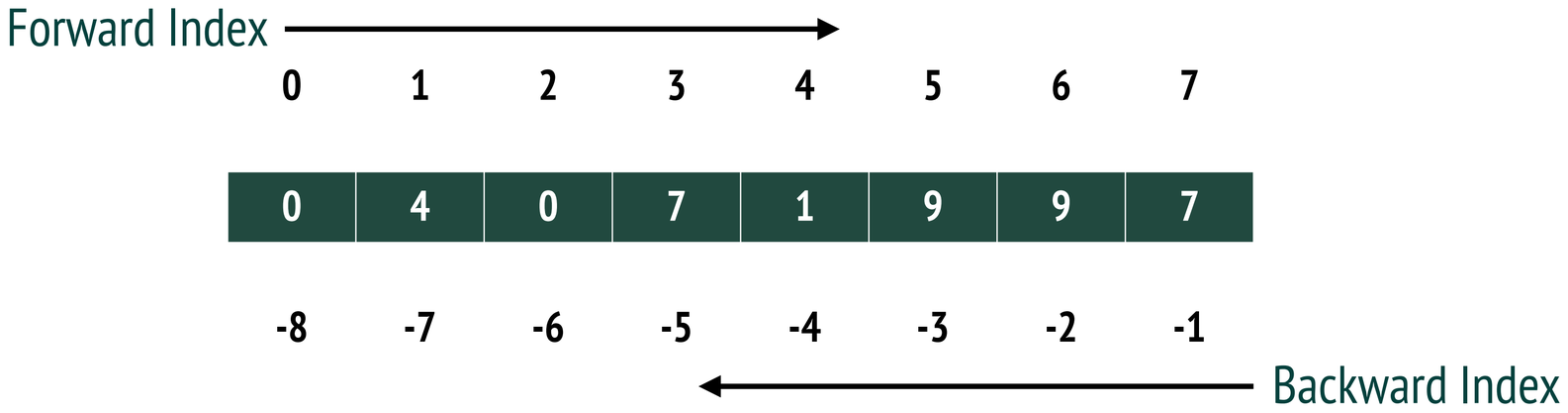



```
my_List[0] -> 0
my_List[3] -> 7
my_List[-4] -> 1
```

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 =


```
data.sort()
```

>>

data =


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

>>

LIST

- Delete an element

```
data =
```


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


```
data =
```


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

LIST

- Delete elements

```
data =
```


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


```
data =
```


```
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 =

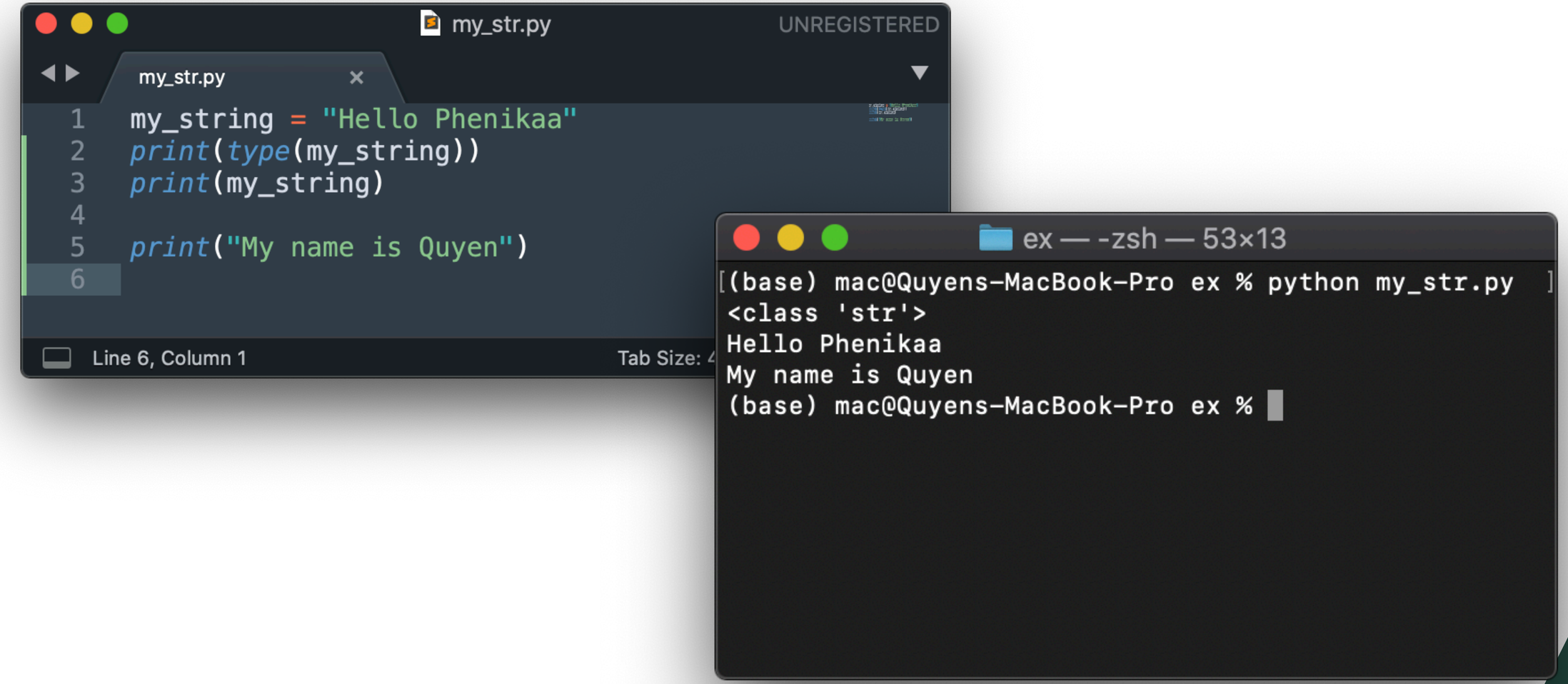
0	4	0	7	1	9	9	7

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

>>

0	4	0	7	1	9	9	7

STRING


my_str.py
UNREGISTERED
my_str.py
1 my_string = "Hello Phenikaa"
2 print(type(my_string))
3 print(my_string)
4
5 print("My name is Quyen")
6
Line 6, Column 1
Tab Size:
ex — -zsh — 53×13
(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

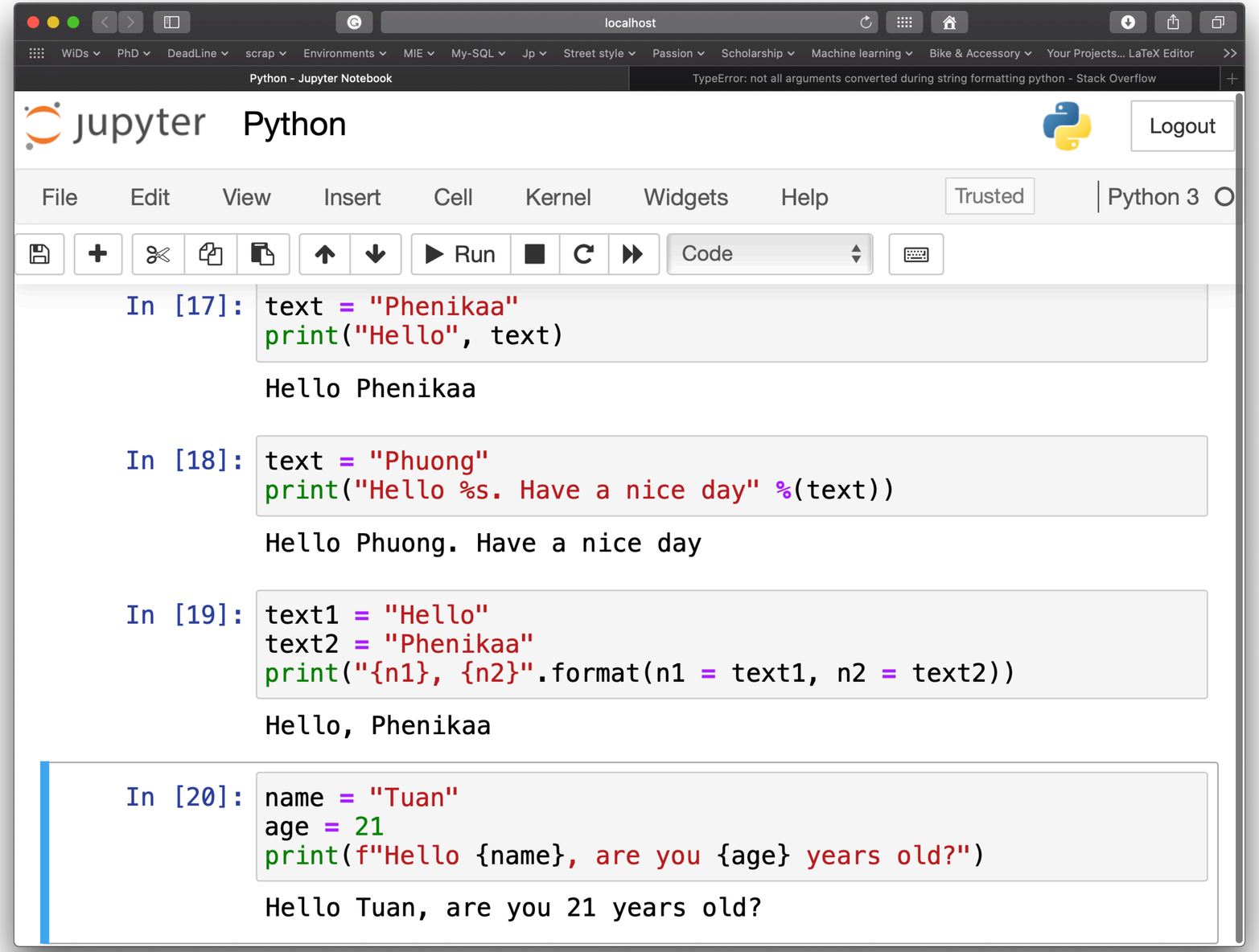
- Specific characters in string

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

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

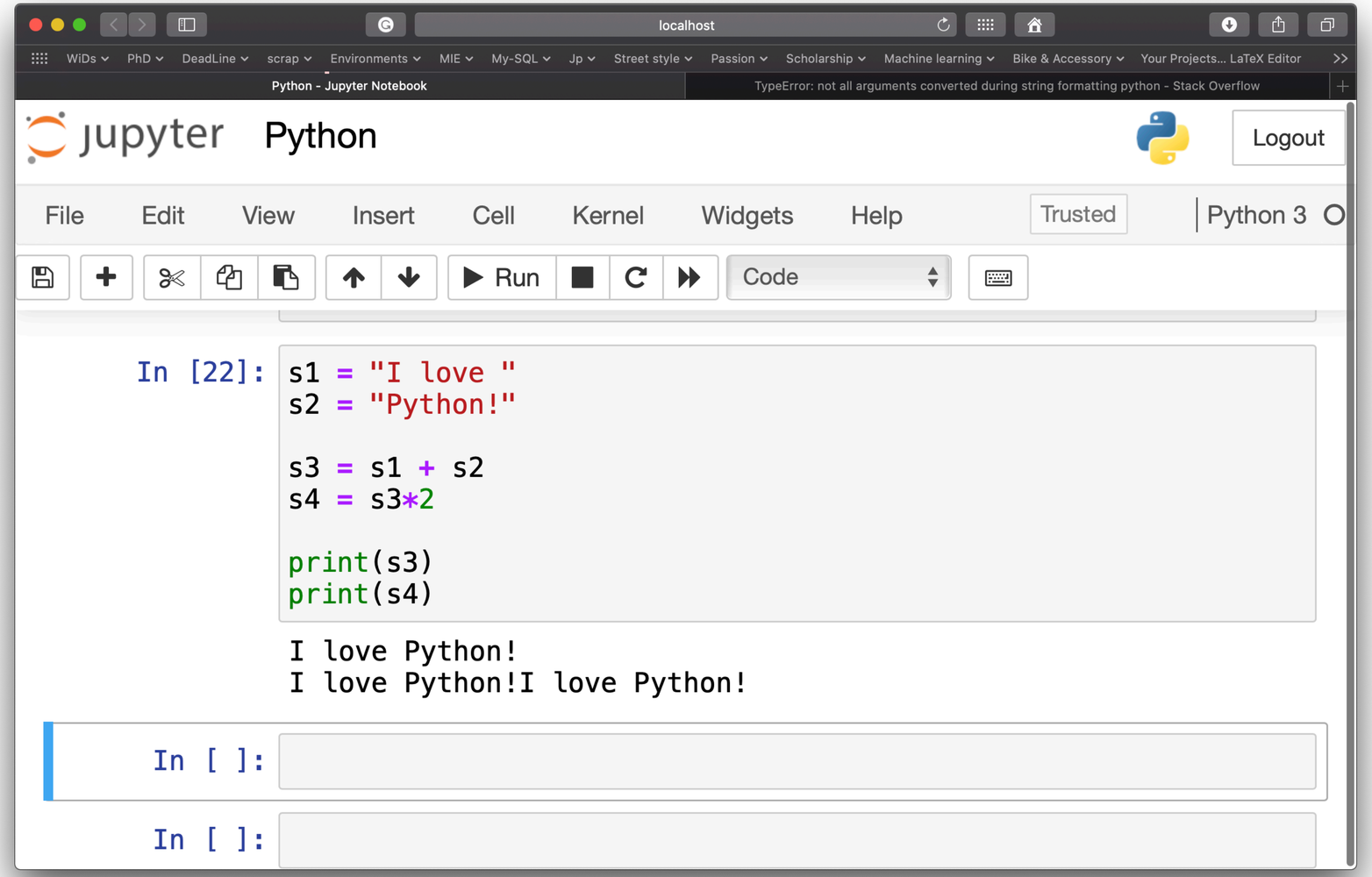
STRING

- Insert to string



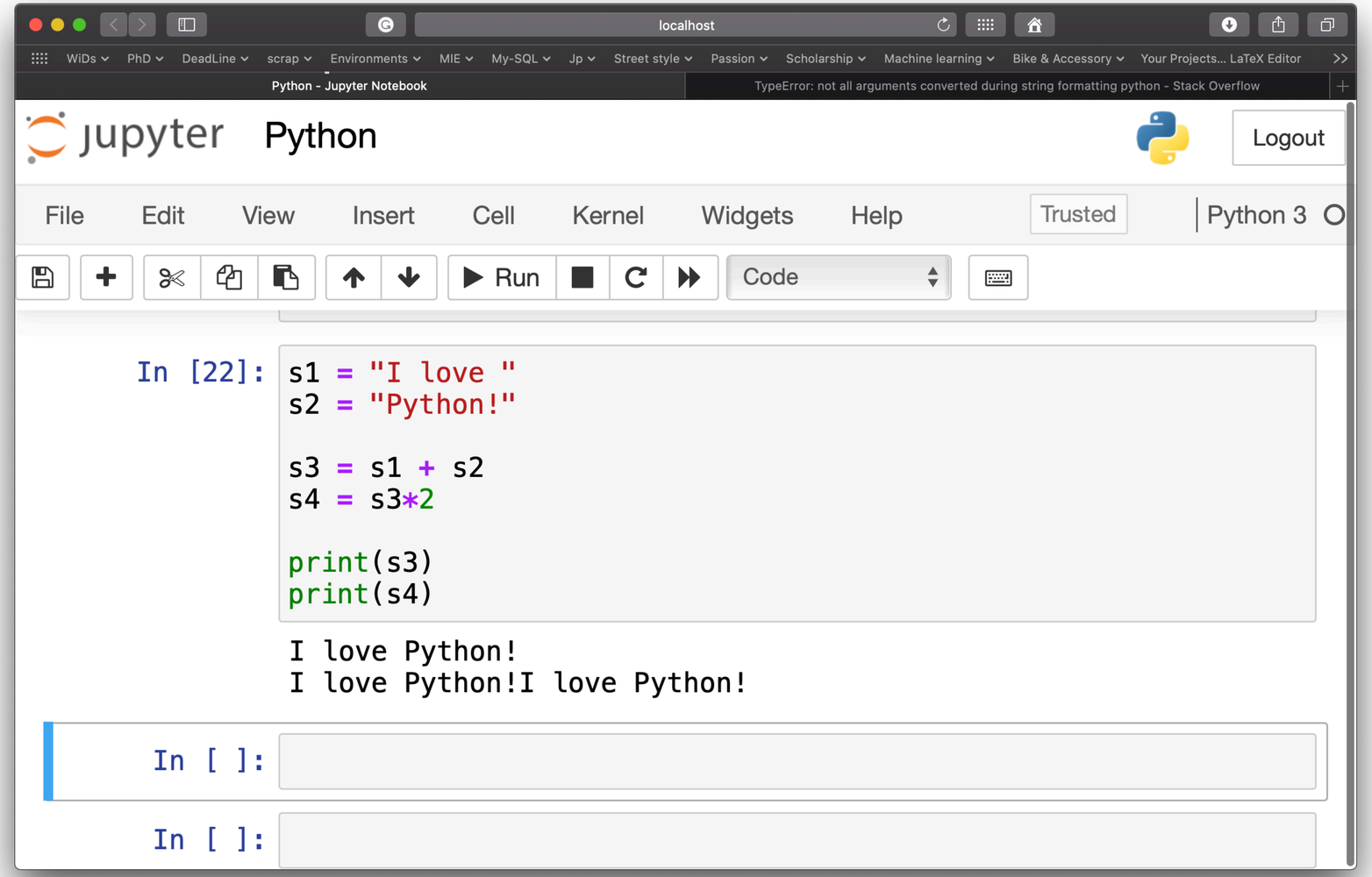
STRING

- + and * operator



STRING

- + and * operator



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 s
#startswith(): Kiểm tra phần đầu của một stri
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

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

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

Insert a set to another set

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

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

Join two sets

```
# join two sets
set1 = {"cat", "dog"}
set2 = {"duck", "tiger"}

set3 = set1.union(set2)
print(set3)
```

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

Not allow duplicate values

```
# No duplication
animals = {"cat", "dog", "tiger"}
print(animals)

animals.add("cat")
print(animals)
```

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

SET

difference function

```python
# difference

set1 = {"apple", "banana", "cherry"}
set2 = {"pineapple", "apple"}

set3 = set1.difference(set2)

print(set3)
```

```
{'cherry', 'banana'}
```

symmetric_difference

```python
# symmetric_difference

set1 = {"apple", "banana", "cherry"}
set2 = {"pineapple", "apple"}

set3 = set1.symmetric_difference(set2)

print(set3)
```

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

difference_update function

```python
# difference_update

set1 = {"apple", "banana", "cherry"}
set2 = {"pineapple", "apple"}

set1.difference_update(set2)

print(set1)
```

```
{'cherry', 'banana'}
```

symmetric_difference_update

```python
# symmetric_difference_update

set1 = {"apple", "banana", "cherry"}
set2 = {"pineapple", "apple"}

set1.symmetric_difference_update(set2)

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.

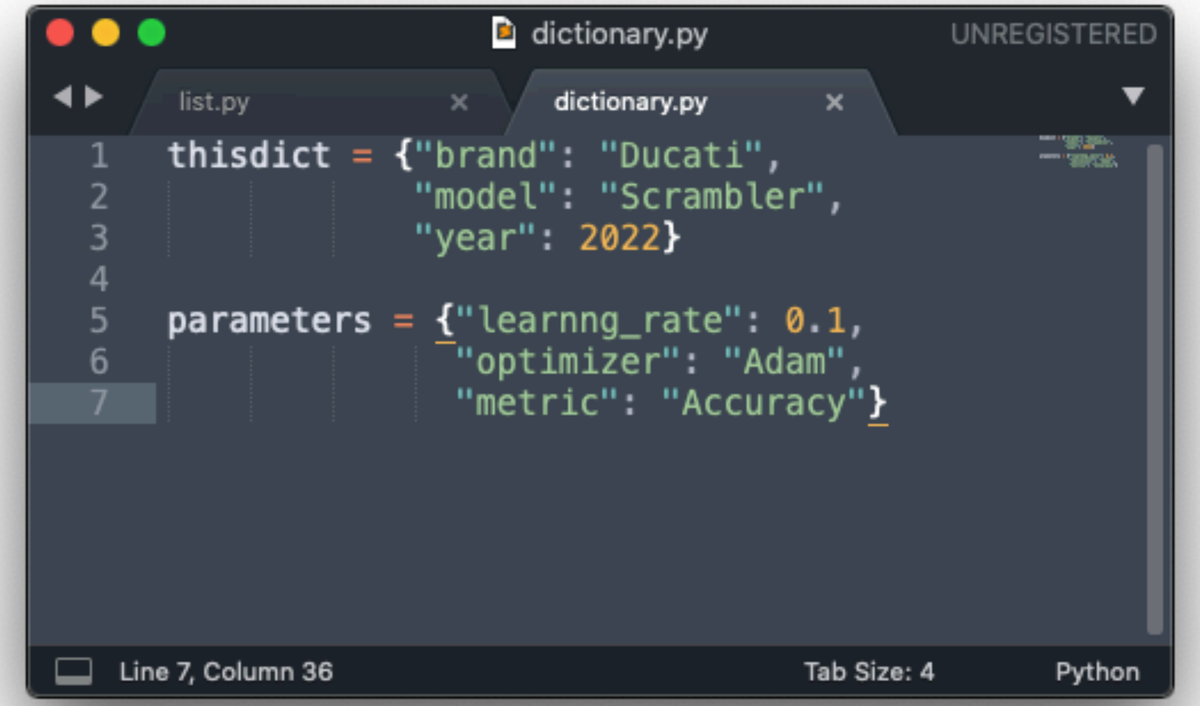
```
1  # remove an item
2  animals = {"cat", "dog", "tiger"}
3  animals.discard("tiger")
4  print(animals)
```

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

DICTIONARY

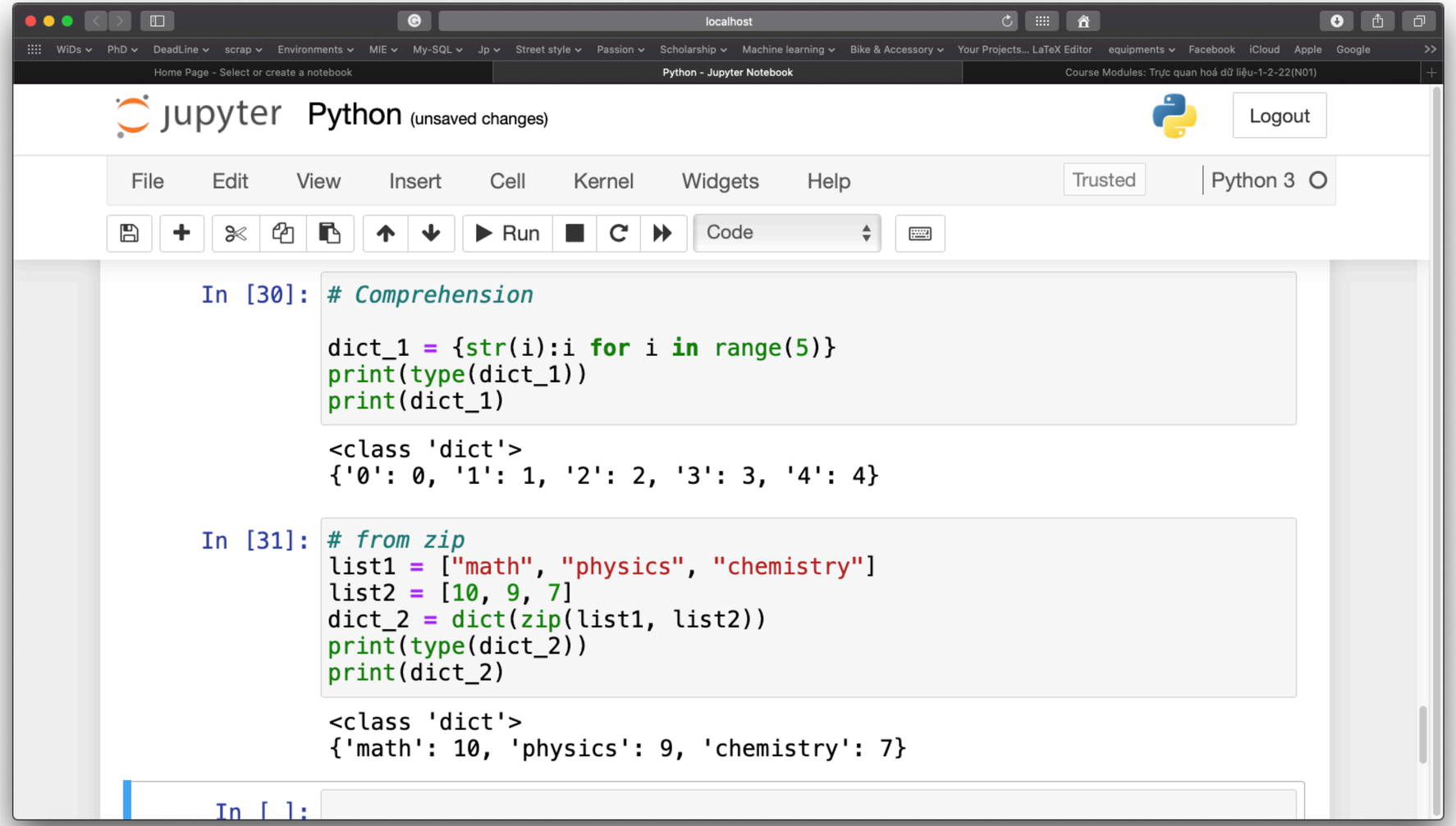
- Dictionaries are used to store data values in `key:value` pairs.

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



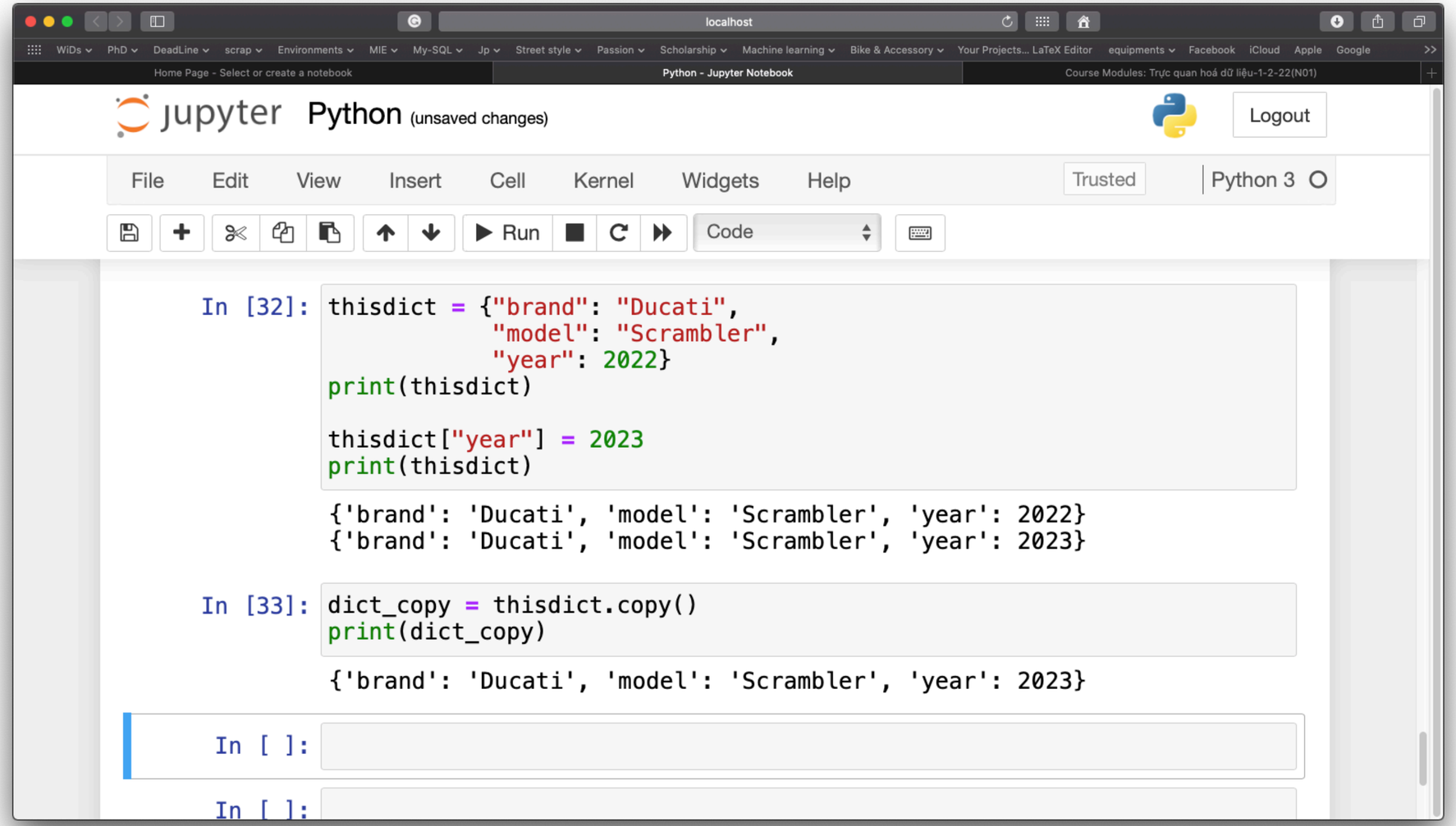
DICTIONARY

- Create Dictionary



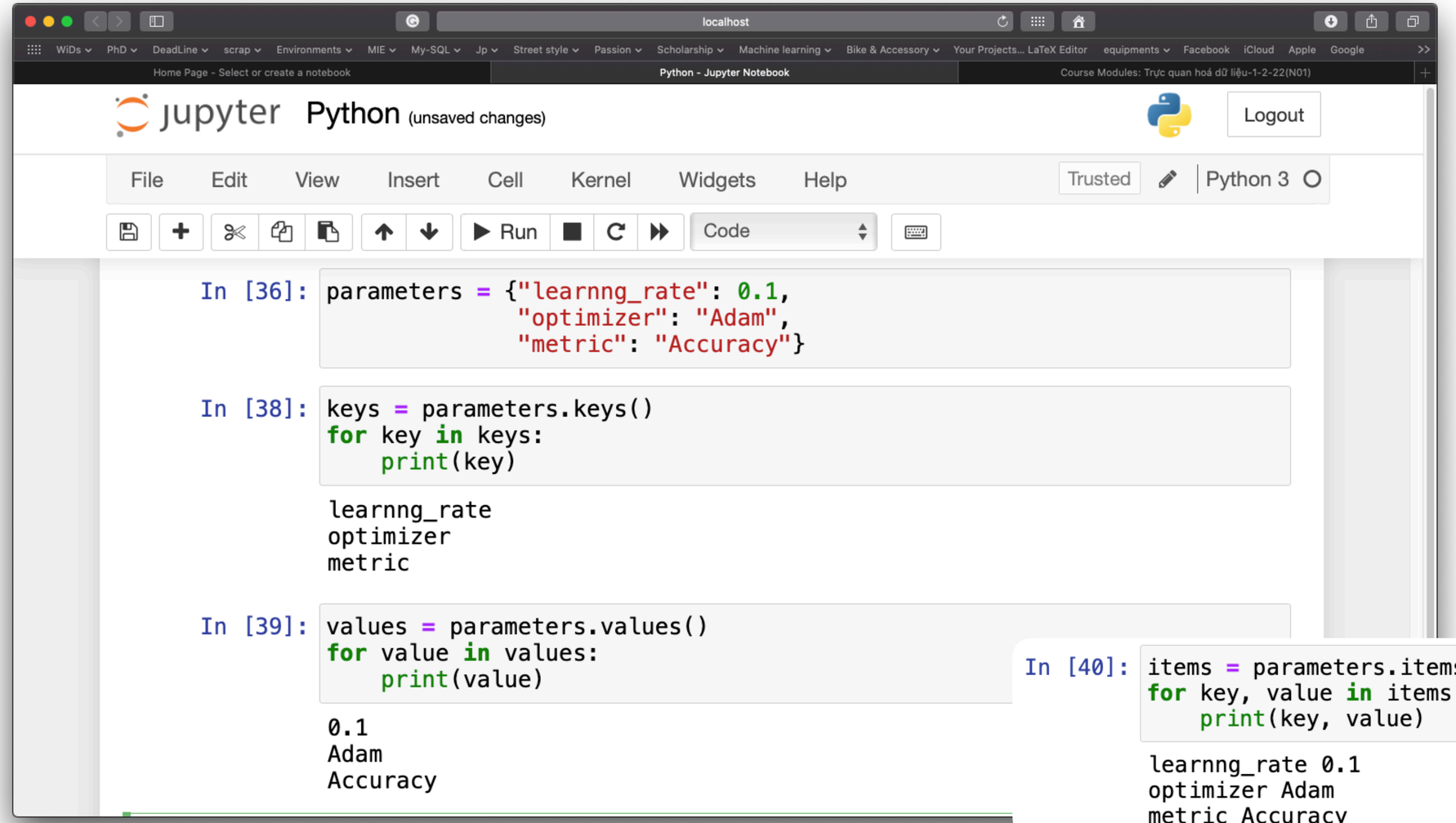
DICTIONARY

- Update value, copy



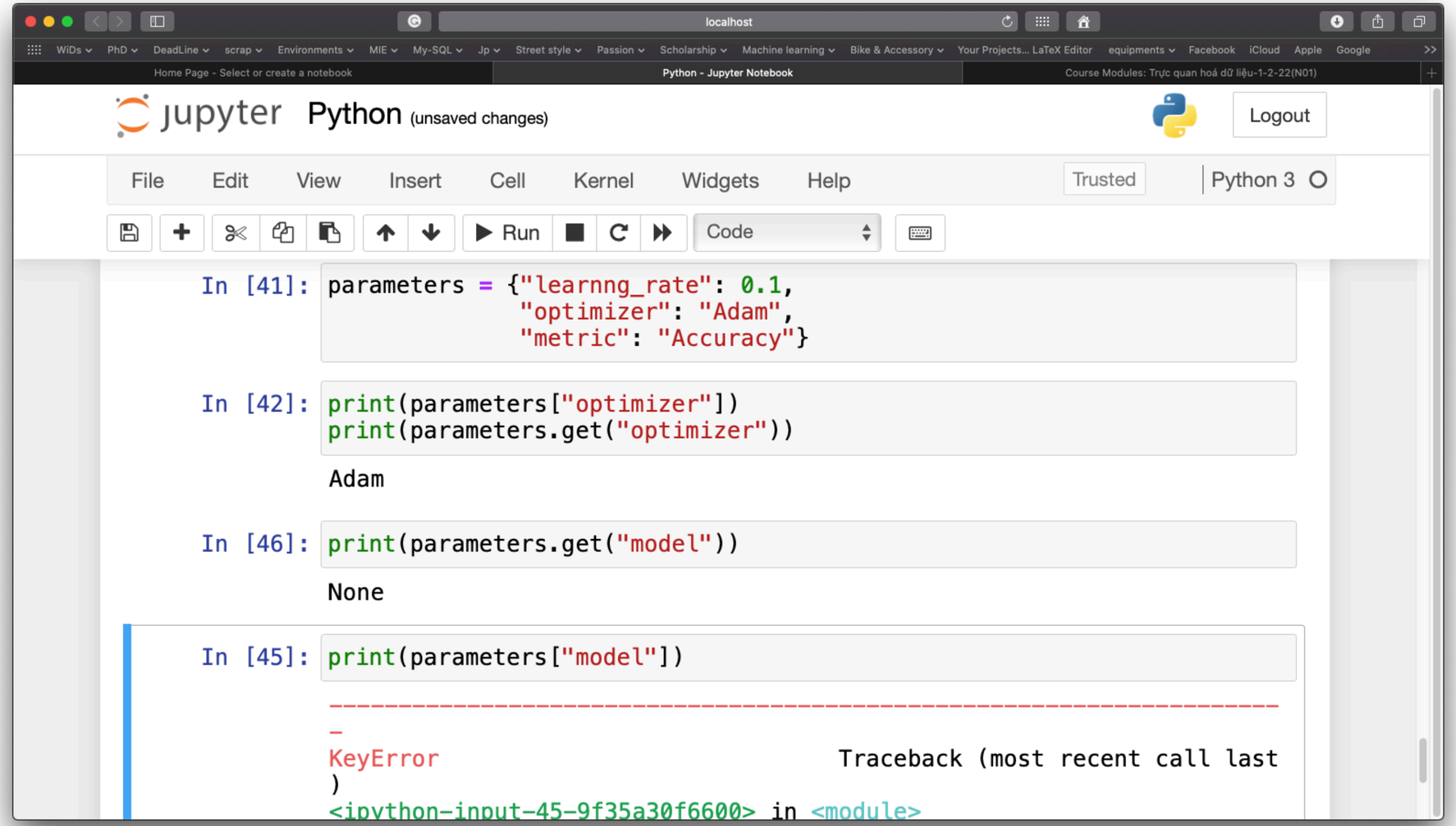
DICTIONARY

- Get keys and values



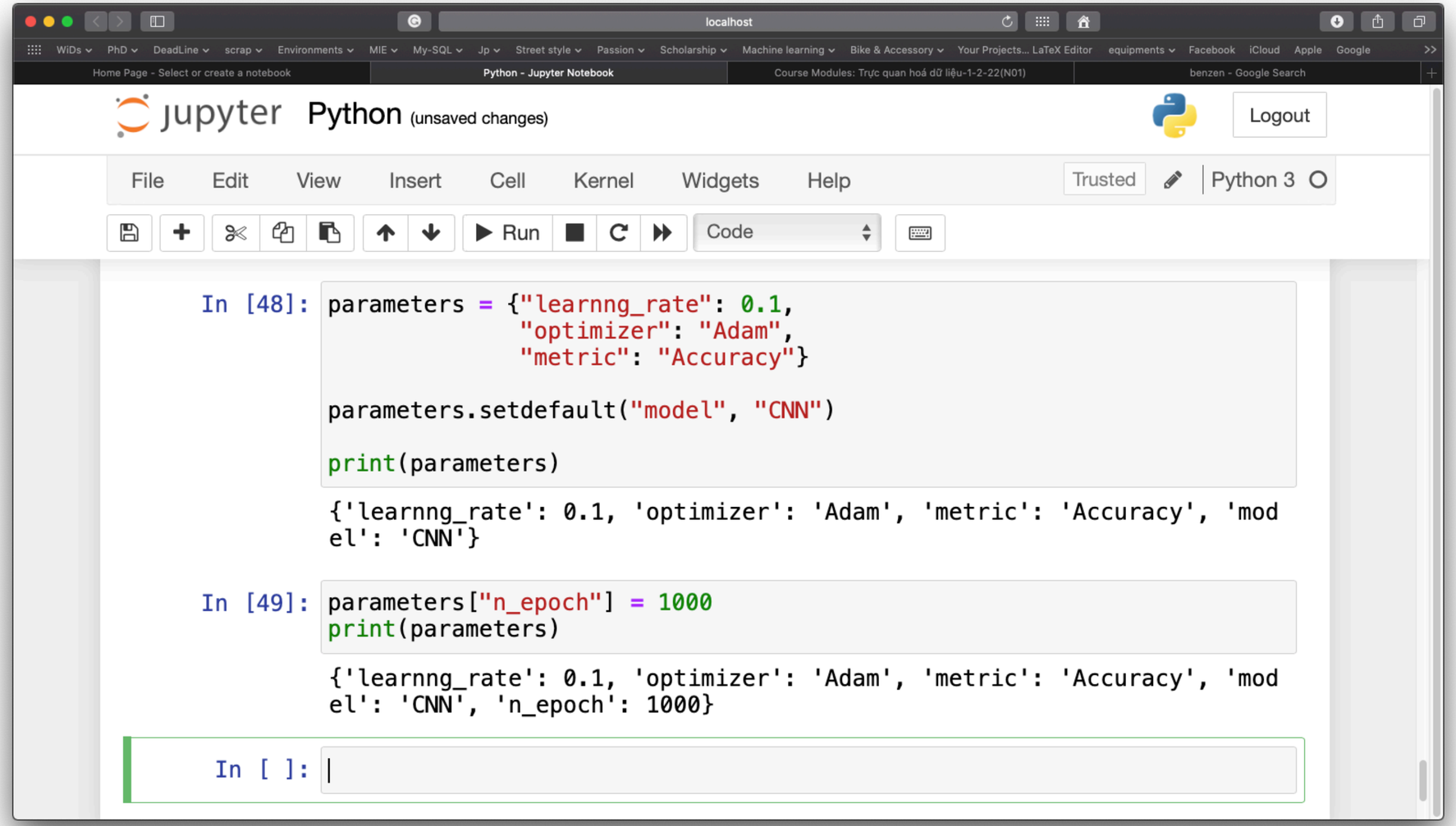
DICTIONARY

- Get a value via a key



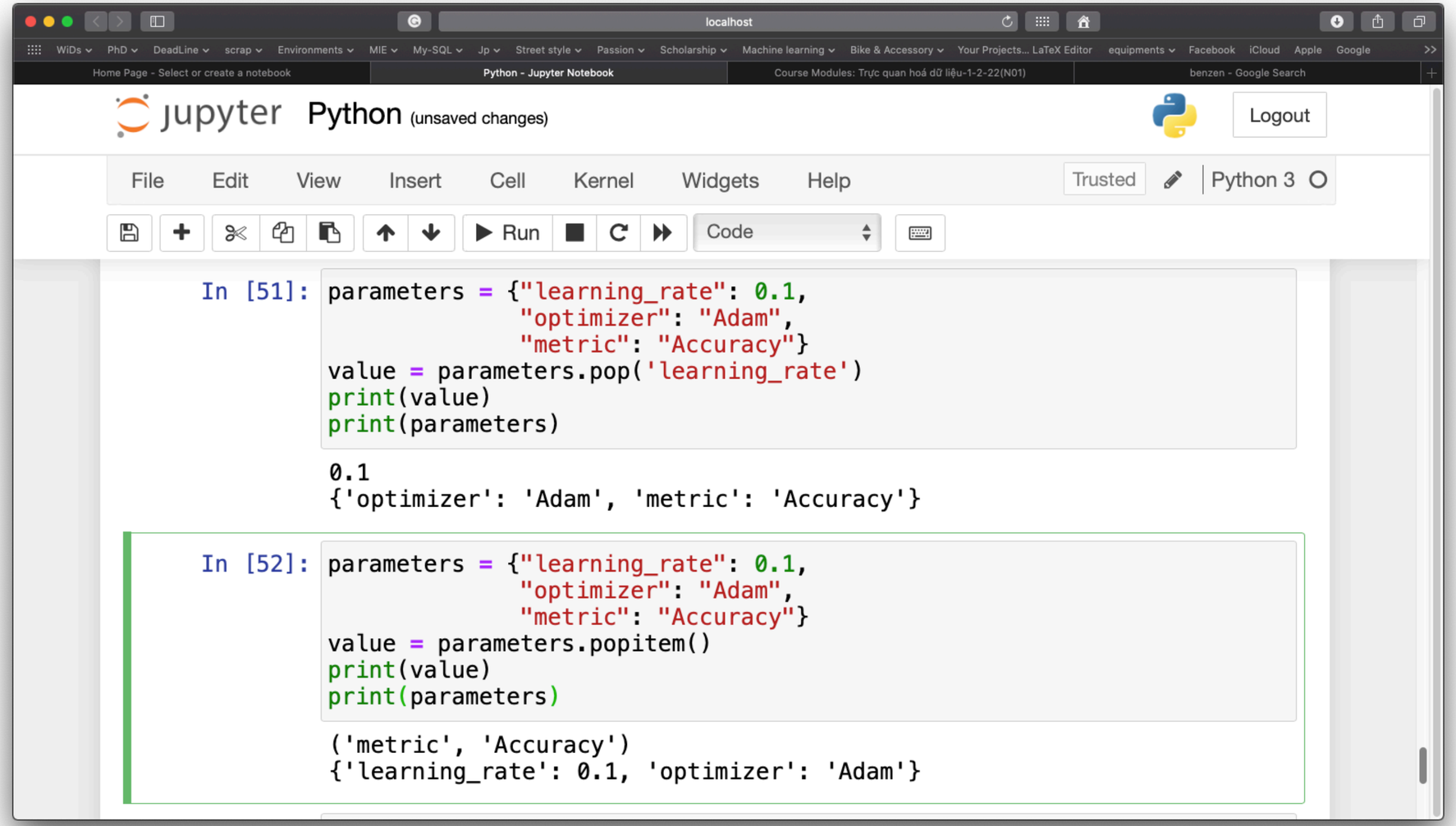
DICTIONARY

- setdefault()



DICTIONARY

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



ASSIGNMENT