

PYTHON

PYTHON VARIABLES

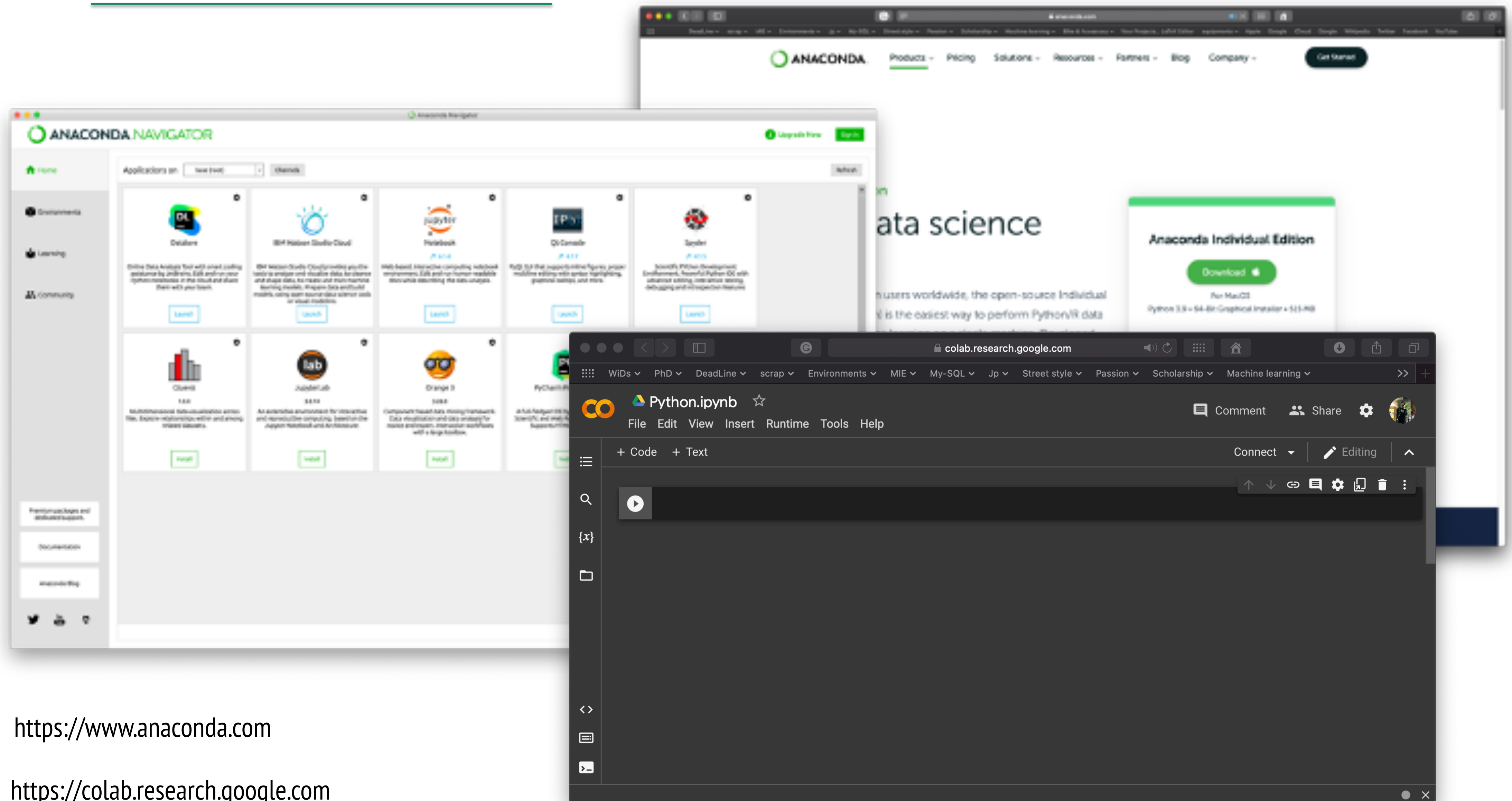
PYTHON OPERATORS

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

OUTLINE

- Getting started
- Python variable and types
- Python operations

LEARNING BY CODING



<https://www.anaconda.com>

<https://colab.research.google.com>

LEARNING BY CODING

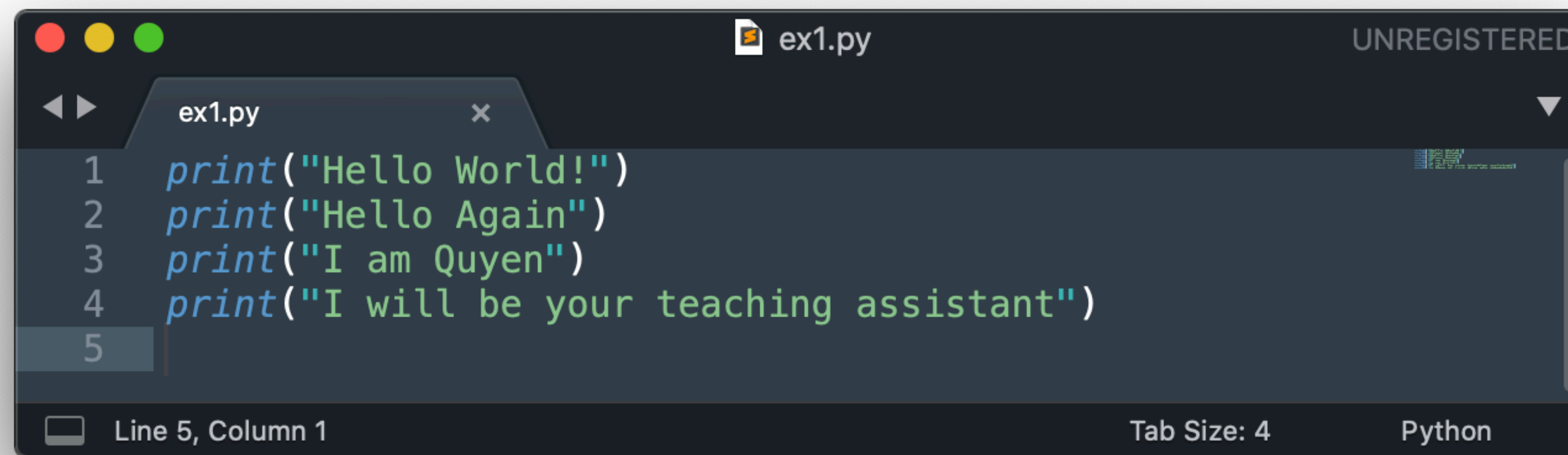
- Alternatice Text Editors

Editor Name	Works On
Visual Studio Code	Linux, macOS, Windows
Notepad++	Linux, macOS, Windows
Textmate	macOS
Sublime Text	Linux, macOS, Windows

.....

HELLO WORLD

- Type the following text (you can modify it) into a single file named **ex1.py** (Python works best with files ending in .py)



A screenshot of a code editor window titled 'ex1.py'. The editor contains four lines of Python code: `print("Hello World!")`, `print("Hello Again")`, `print("I am Quyen")`, and `print("I will be your teaching assistant")`. The cursor is at line 5, column 1. The status bar at the bottom indicates 'Line 5, Column 1', 'Tab Size: 4', and 'Python'.

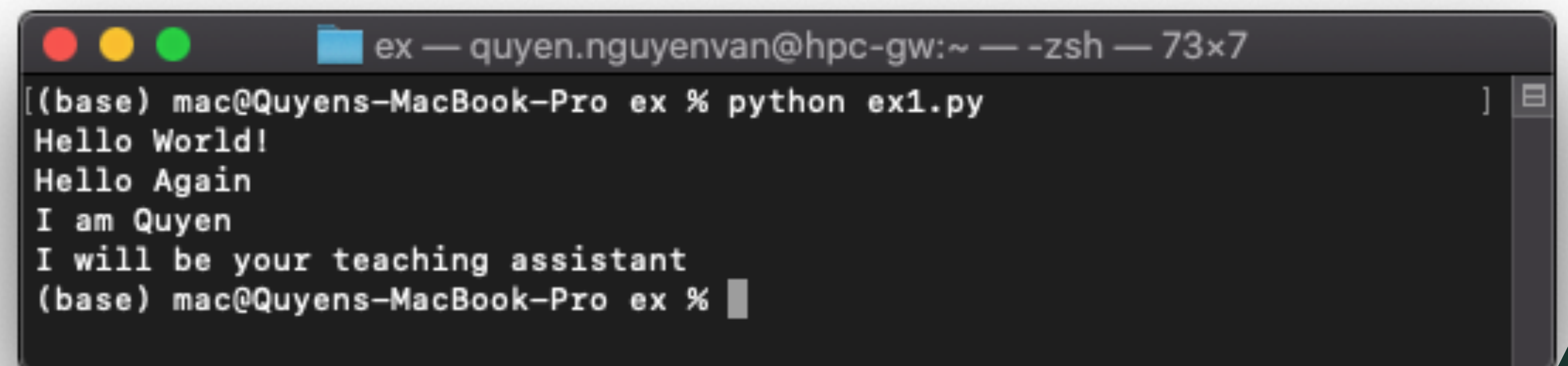
```
1 print("Hello World!")
2 print("Hello Again")
3 print("I am Quyen")
4 print("I will be your teaching assistant")
5
```

- In macOS Terminal or (maybe) Linux

run the file by typing:

python ex1.py

- What you should see:

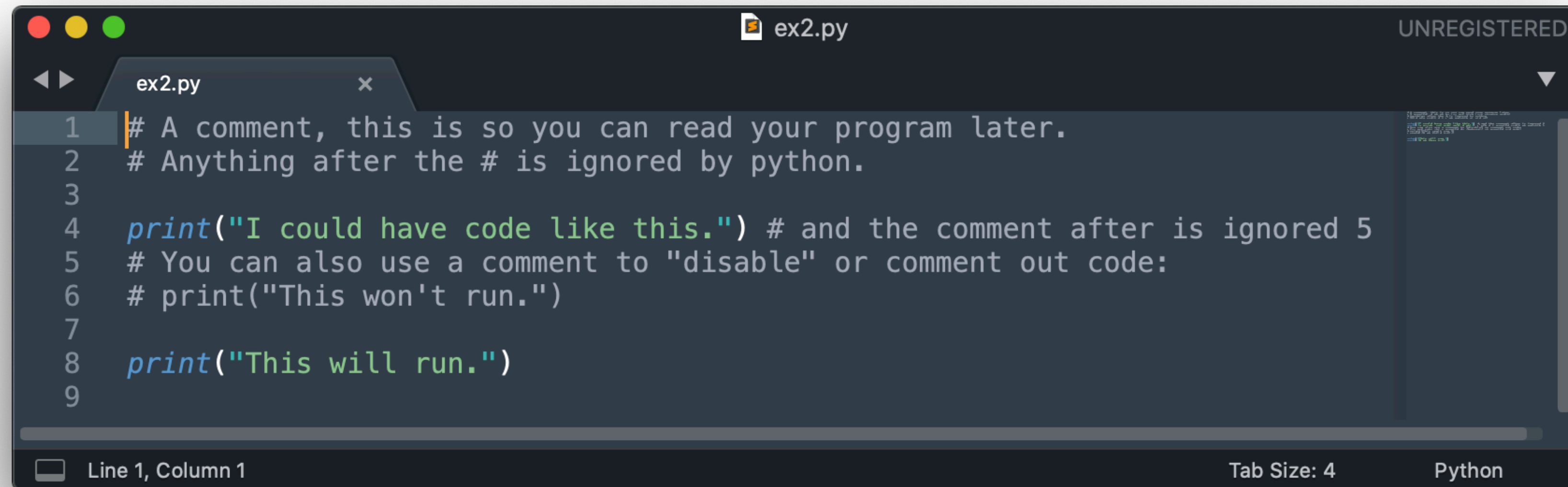


A screenshot of a macOS Terminal window. The title bar shows 'ex — quyen.nguyenvan@hpc-gw:~ — -zsh — 73x7'. The terminal output shows the execution of `python ex1.py` resulting in four lines of text: 'Hello World!', 'Hello Again', 'I am Quyen', and 'I will be your teaching assistant'. The prompt `(base) mac@Quyens-MacBook-Pro ex %` is visible at the bottom.

```
(base) mac@Quyens-MacBook-Pro ex % python ex1.py
Hello World!
Hello Again
I am Quyen
I will be your teaching assistant
(base) mac@Quyens-MacBook-Pro ex %
```


COMMENT

- Here is how you use comments in Python:

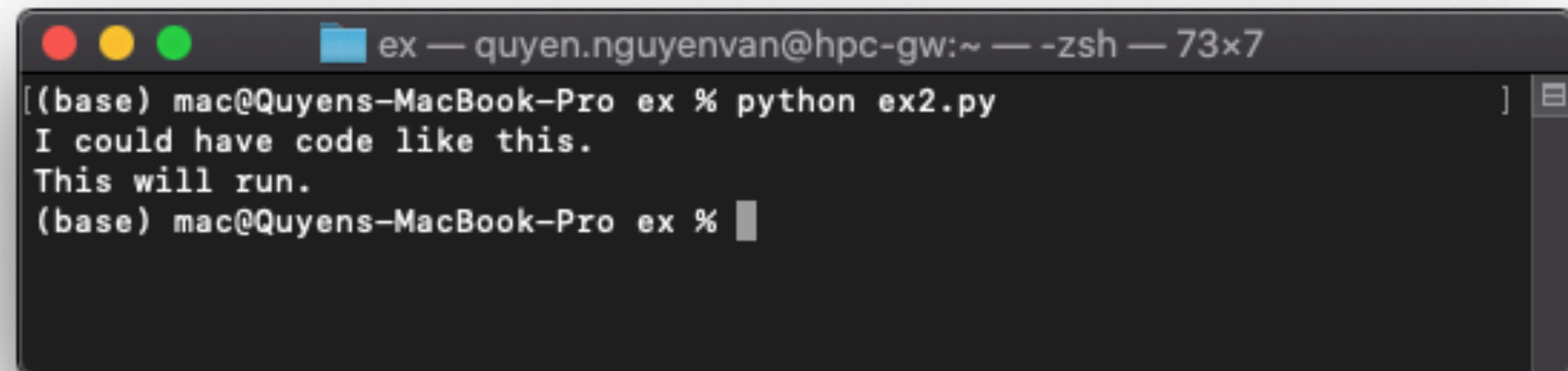


A screenshot of a code editor window titled 'ex2.py' with a 'UNREGISTERED' label in the top right corner. The editor shows the following Python code:

```
1 # A comment, this is so you can read your program later.
2 # Anything after the # is ignored by python.
3
4 print("I could have code like this.") # and the comment after is ignored 5
5 # You can also use a comment to "disable" or comment out code:
6 # print("This won't run.")
7
8 print("This will run.")
9
```

The status bar at the bottom indicates 'Line 1, Column 1', 'Tab Size: 4', and 'Python'.

- What you should see:



A screenshot of a terminal window titled 'ex — quyen.nguyenvan@hpc-gw:~ — -zsh — 73x7'. The terminal shows the execution of the Python script:

```
(base) mac@Quyens-MacBook-Pro ex % python ex2.py
I could have code like this.
This will run.
(base) mac@Quyens-MacBook-Pro ex %
```

VARIABLE AND VALUE

- equal sign is an assignment of a value to a variable name

Variable

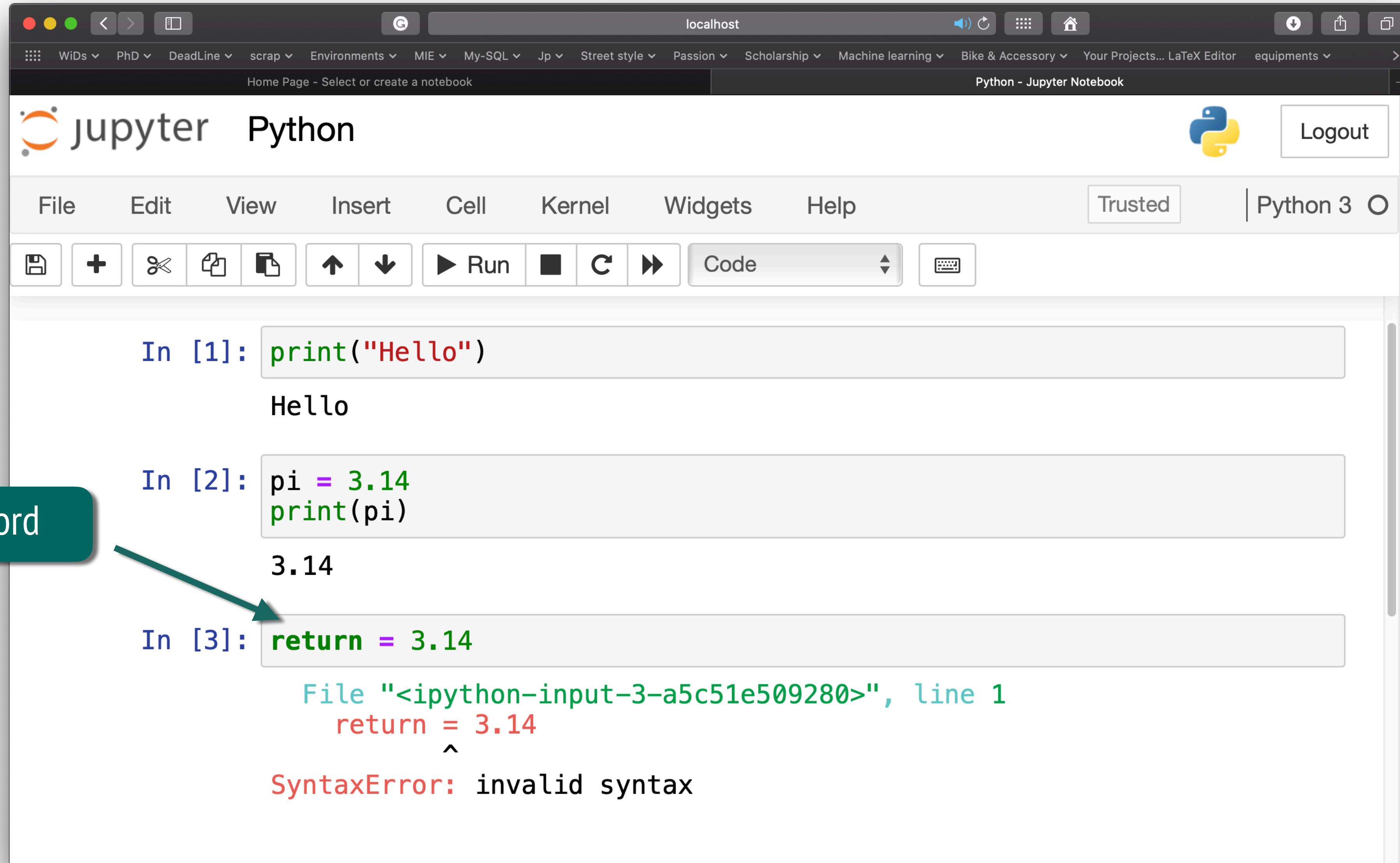
→ `pi = 3.14159` ←
`pi_approx = 22/7`

Value

- Variable name should have a meaning
- cannot use keywords

and	continue	except	global	lambda	print	with
as	def	exec	if	TRUE	raise	yeild
assert	del	finaally	import	not	return	FALSE
break	elif	for	in	or	try	
class	else	from	is	pass	while	

VARIABLE AND VALUE



```
In [1]: print("Hello")
Hello

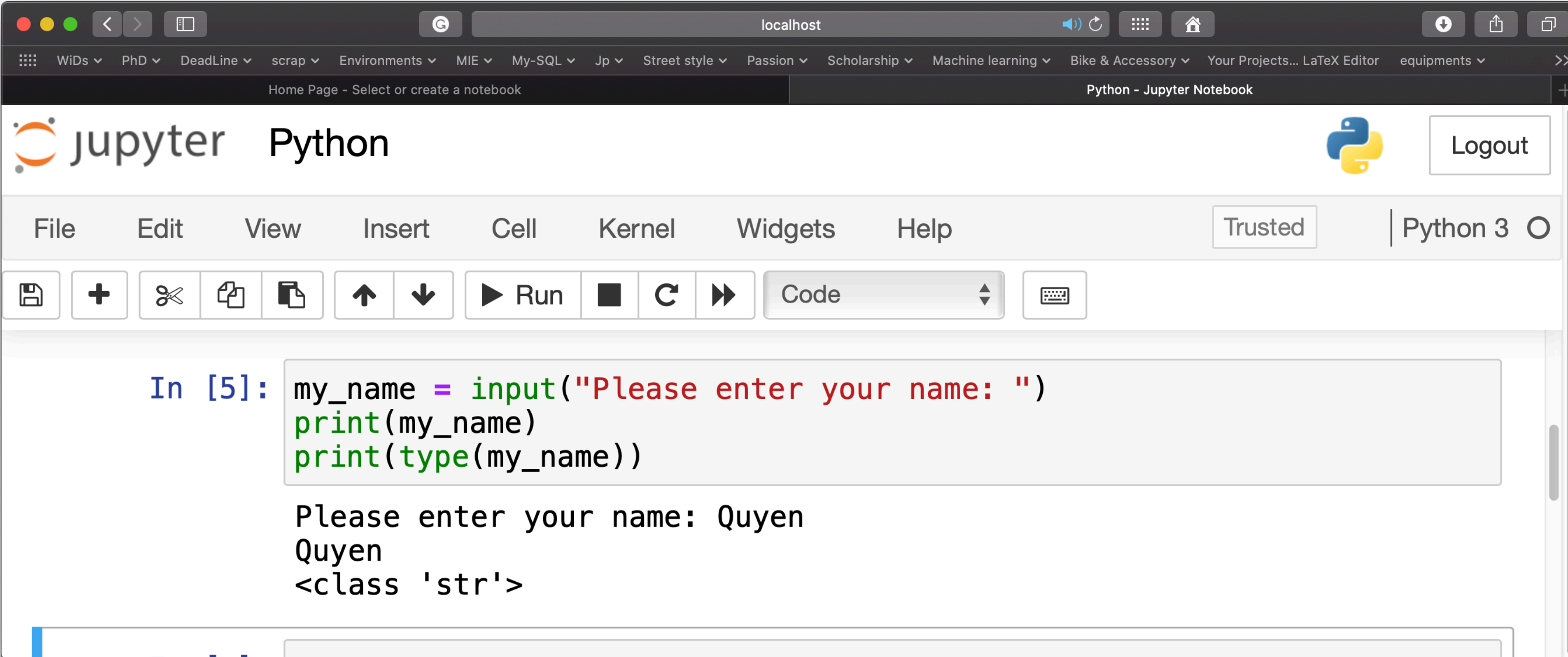
In [2]: pi = 3.14
        print(pi)
3.14

In [3]: return = 3.14
File "<ipython-input-3-a5c51e509280>", line 1
      return = 3.14
             ^
SyntaxError: invalid syntax
```

Keyword

BUILT-IN FUNCTION

- `print (parameter)`
- `type (parameter)`
- `input (prompt)`



The screenshot shows a Jupyter Notebook interface in a web browser. The browser's address bar shows 'localhost'. The notebook's top bar includes the Jupyter logo, the word 'Python', a 'Logout' button, and a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. Below the menu bar is a toolbar with icons for saving, creating a new cell, deleting, copying, pasting, and running code. The main area of the notebook displays a code cell with the following Python code:

```
In [5]: my_name = input("Please enter your name: ")
        print(my_name)
        print(type(my_name))
```

The output of the code cell is shown below the code:

```
Please enter your name: Quyen
Quyen
<class 'str'>
```

DATA TYPES

Text Type:	<code>str</code>
Numeric Types:	<code>int, float, complex</code>
Sequence Types:	<code>list, tuple, range</code>
Mapping Type:	<code>dict</code>
Set Types:	<code>set, frozenset</code>
Boolean Type:	<code>bool</code>
Binary Types:	<code>bytes, bytearray,</code>

- Examples

<code>x = "Hello World"</code>	<code>str</code>
<code>x = 20</code>	<code>int</code>
<code>x = 20.5</code>	<code>float</code>
<code>x = 1j</code>	<code>complex</code>
<code>x = ["apple", "banana",</code>	<code>list</code>

DATA TYPES

- Type conversions
 - ➡ can **convert object of one type to another**
 - ➡ `float(3)` converts interger 3 to float 3.0
 - ➡ `int(3.9)` truncates float 3.9 to interger 3

EXPRESSIONS

- Combine objects and operators to form expressions
- an expression has a value, which has a type
- syntax for a simple expression

`<object> <operator> <object>`

PYTHON OPERATORS

• Python Arithmetic Operators

+	Addition	<code>x + y</code>
-	Subtraction	<code>x - y</code>
*	Multiplication	<code>x * y</code>
/	Division	<code>x / y</code>
%	Modulus	<code>x % y</code>
**	Exponentiation	<code>x ** y</code>
//	Floor division	<code>x // y</code>

• Python Identity Operators

<code>is</code>	Returns True if both variables are the same object	<code>x is y</code>
<code>is not</code>	Returns True if both variables are not the same object	<code>x is not y</code>

PYTHON OPERATORS

- Python Comparison Operators

and

Returns True if both statements are true

```
x < 5 and x  
< 10
```

or

Returns True if one of the statements is true

```
x < 5 or x <  
4
```

not

Reverse the result, returns False if the result is true

```
not (x < 5 and  
x < 10)
```

`==, !=, >, <, >=, <=`

PYTHON OPERATORS

- Python Membership Operators

<code>in</code>	Returns True if a sequence with the specified value is present in the object	<code>x in y</code>
<code>not in</code>	Returns True if a sequence with the specified value is not present in the object	<code>x not in y</code>

ASSIGNMENT
