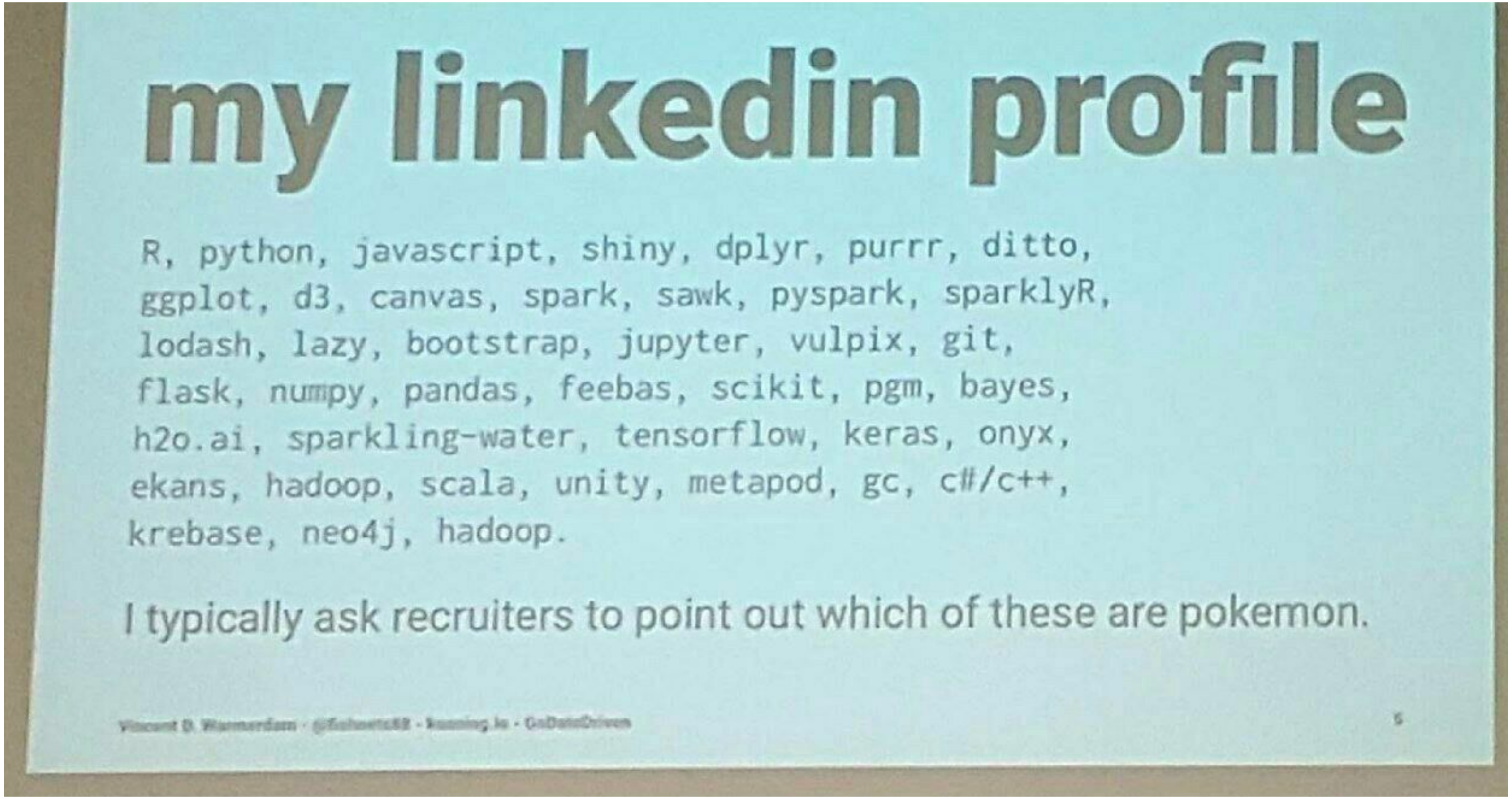
Columbia Business School
AT THE VERY CENTER OF BUSINESS

Introduction to Python

Session 1: Your First Piece of Code!

Hmm...



my linkedin profile

R, python, javascript, shiny, dplyr, purrr, ditto,
ggplot, d3, canvas, spark, sawk, pyspark, sparklyR,
lodash, lazy, bootstrap, jupyter, vulpix, git,
flask, numpy, pandas, feebas, scikit, pgm, bayes,
h2o.ai, sparkling-water, tensorflow, keras, onyx,
ekans, hadoop, scala, unity, metapod, gc, c#/c++,
krebase, neo4j, hadoop.

I typically ask recruiters to point out which of these are pokemon.

Vincent D. Warmerdam · @fishnets88 · koaning.io · GoDataDriven 5

Columbia Business School

Why Learn Coding?

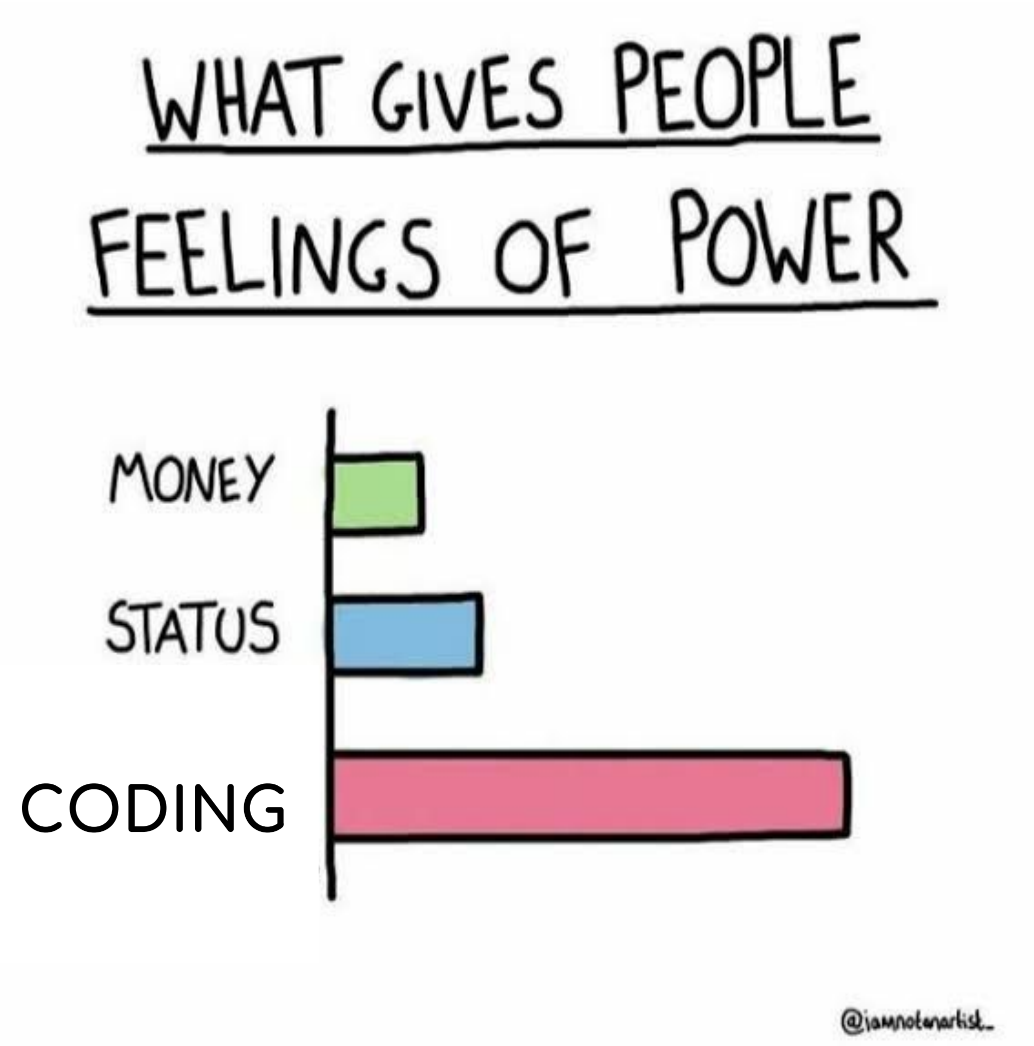
"I think everybody in this country should learn how to program a computer, because it teaches you how to think."

- *Steve Jobs*

"Projects are created with multiple people with varied perspectives, ideas, and skills coming together—and that often involves ***working with engineers****.*

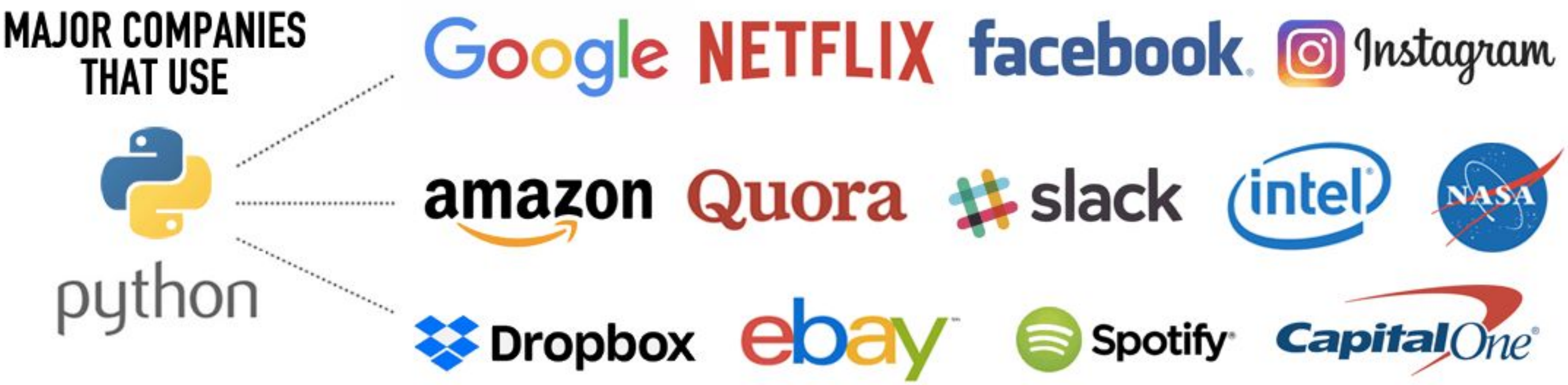
By having some knowledge of coding, you'll have a better sense of ***what's realistic in terms of results, quality, and timeline****, making you a much* ***better teammate and leader****."*

Why Learn Coding?



@iamnotanartist_

Why Python?


MAJOR COMPANIES THAT USE
python
Google
NETFLIX
facebook
Instagram
amazon
Quora
slack
intel
NASA
Dropbox
ebay
Spotify
Capital One

Why Python?

Most in-demand programming languages of 2020

Based on LinkedIn job postings in the USA - June 2020

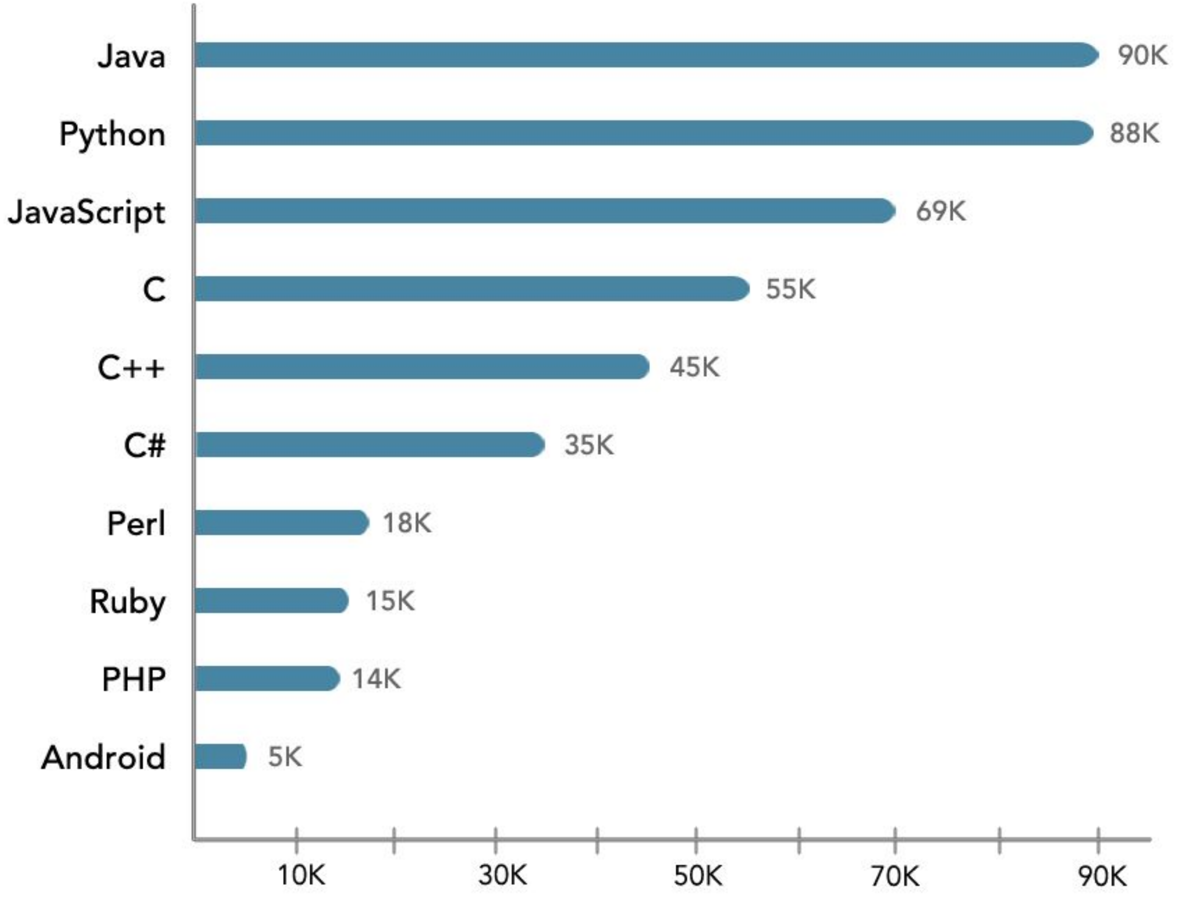
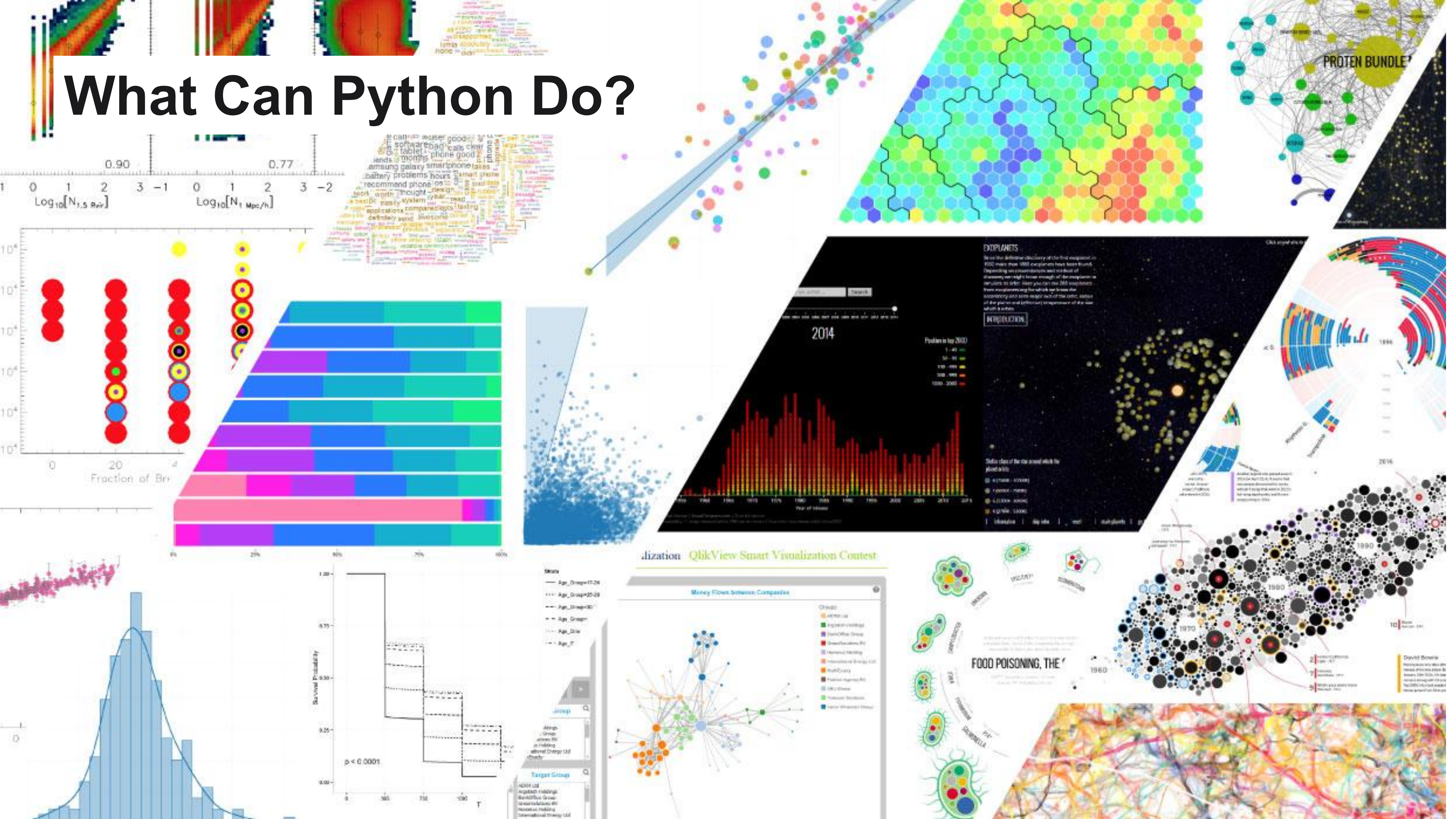
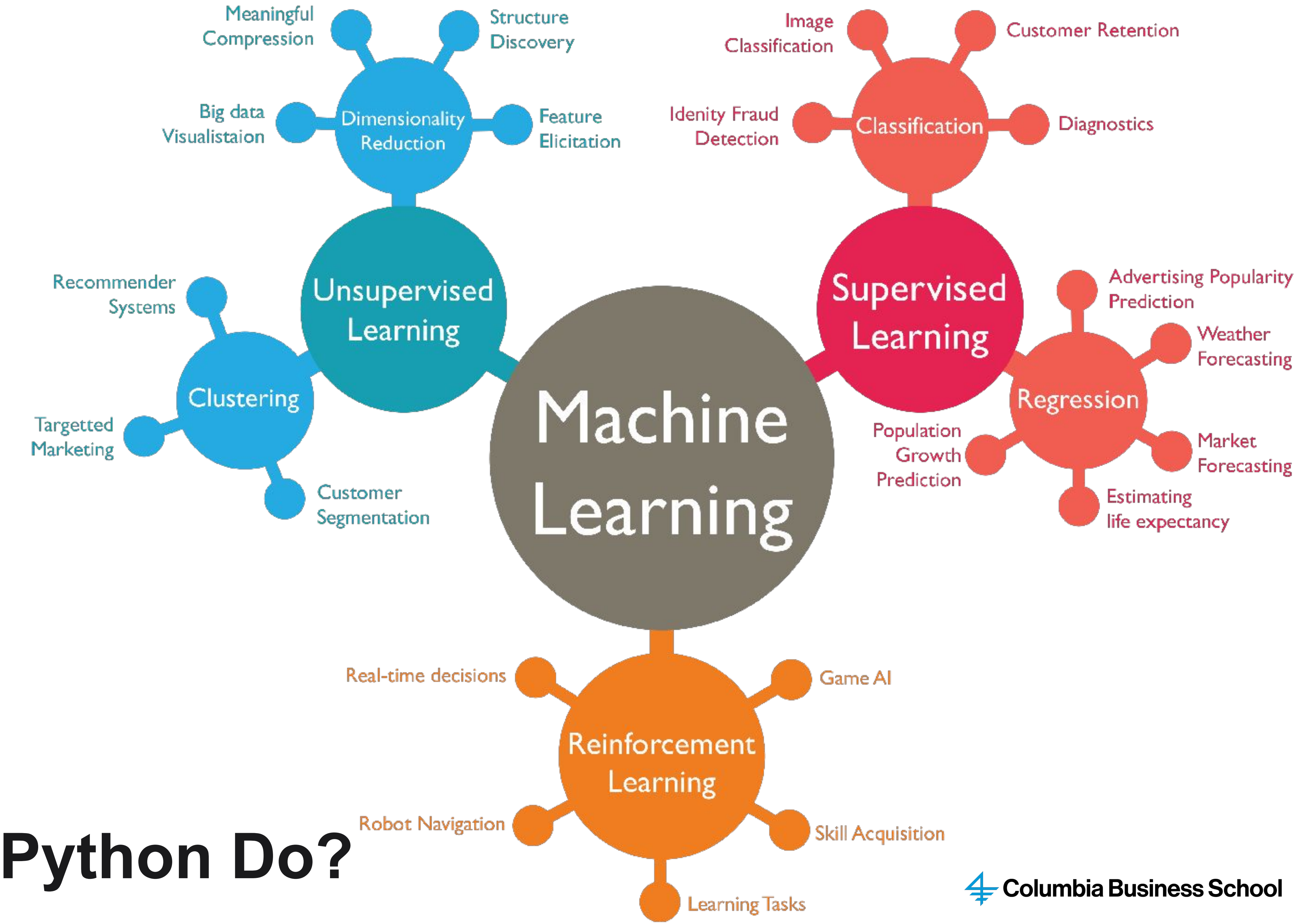


Image Source: CodingNomads.co

What Can Python Do?





What Can Python Do?

Columbia Business School

Seminar Roadmap

Sessions	Content
Session 1	How to Use Jupyter Notebook, Python Basics, Strings
Session 2	Integers, Floats, Lists, Tuples
Session 3	Logic Operators, Conditional Statements, Loops
Session 4	Sets, Dictionaries
Session 5	Functions, Python Packages
Session 6	Sample Exam Questions, Revision!

Our Goals Together

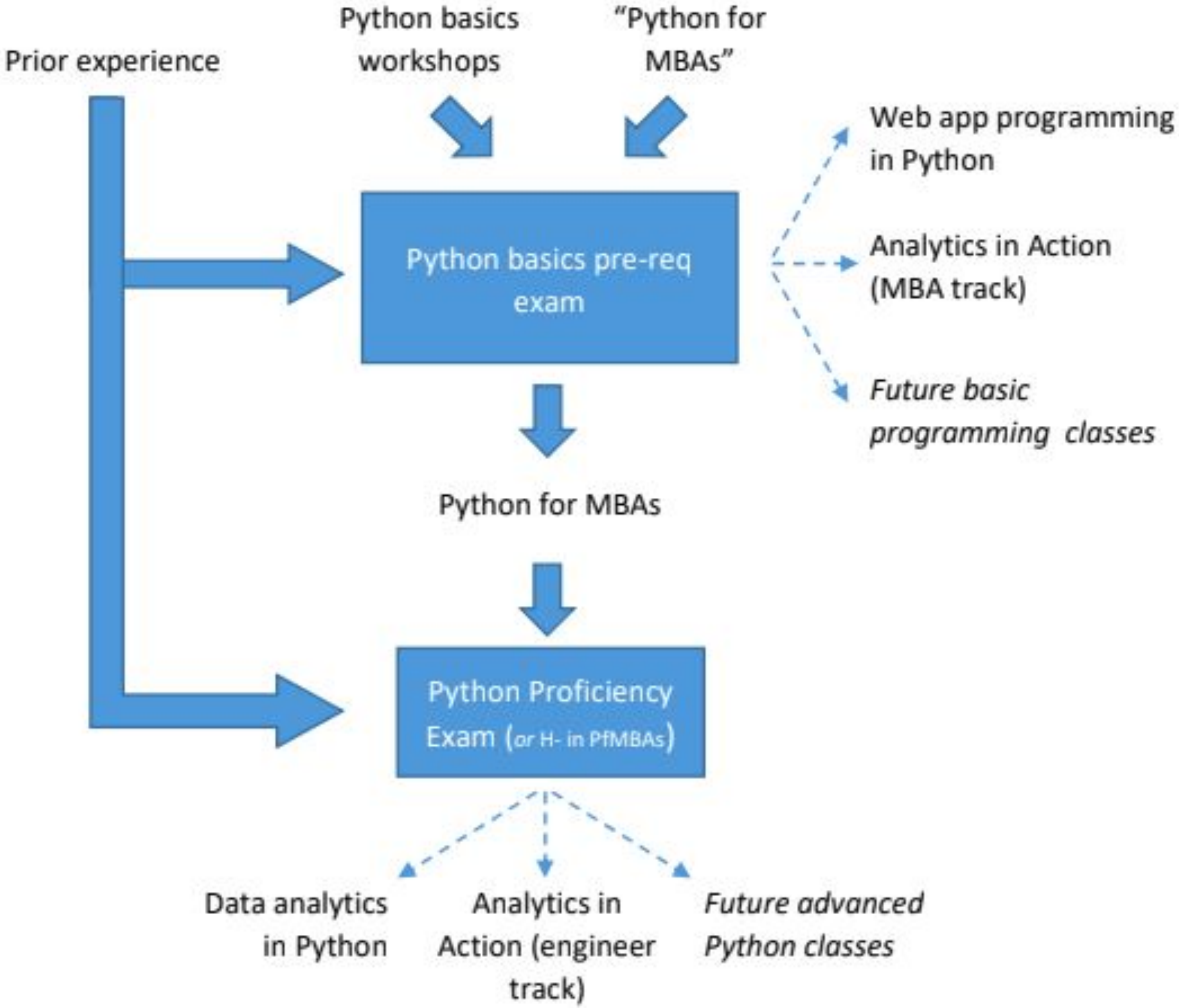
1) Know your Python toolbox
   - Syntax
   - Data structures
   - Packages

2) Know when to use which tool

3) Ace your Python Basics Waiver Exam

4) Have fun coding!

Suggested Readings
Python for MBAs, Mattan Griffel and Daniel Guetta - Chapters 1 to 4
These slides closely follow this excellent book written by your CBS Professors!
Available for FREE to download from CU libraries: https://bit.ly/python_for_mbas



Prior experience

Python basics workshops

"Python for MBAs"

Python basics pre-req exam

Web app programming in Python

Analytics in Action (MBA track)

Future basic programming classes

Python for MBAs

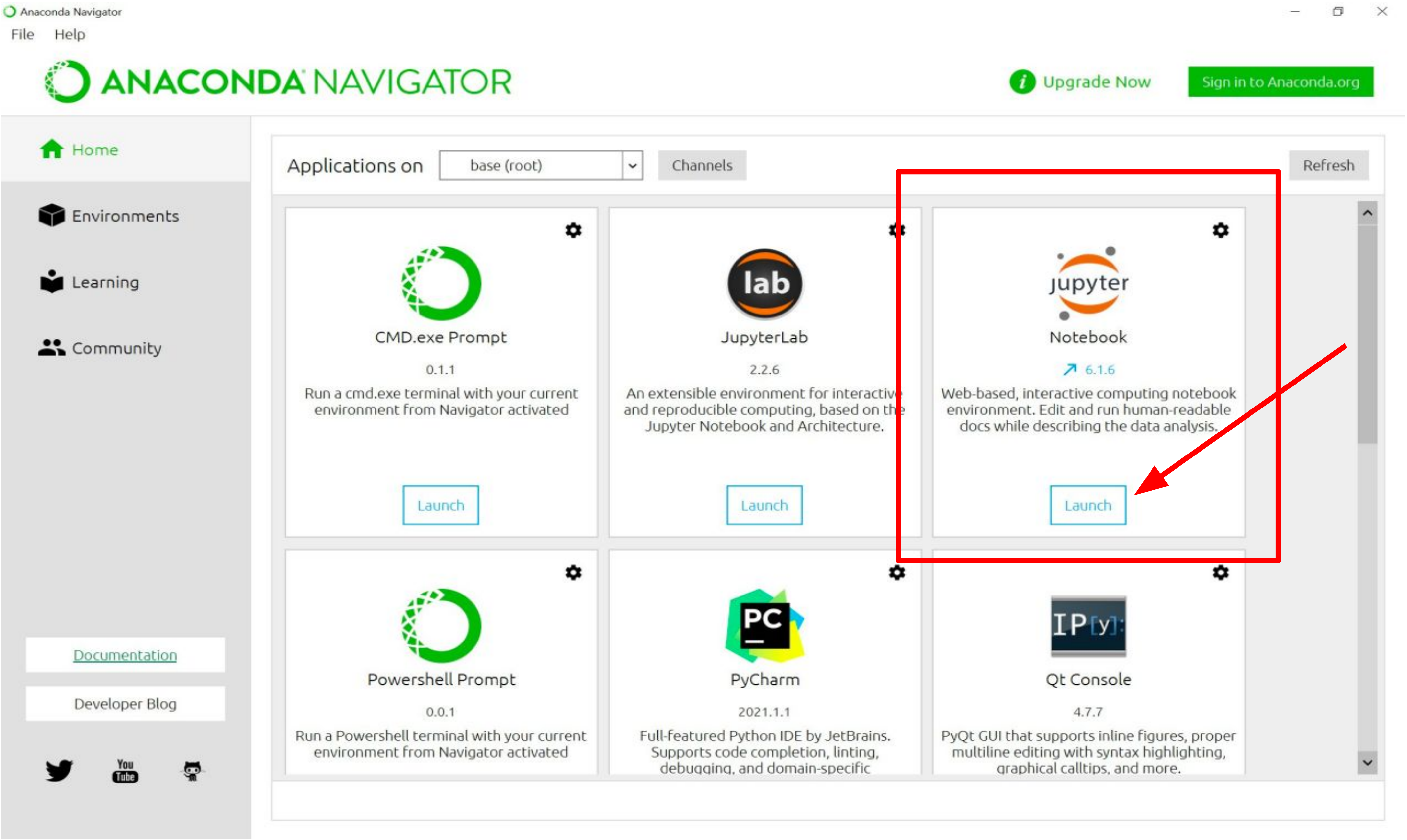
Python Proficiency Exam (*or* H- in PfMBAs)

Data analytics in Python

Analytics in Action (engineer track)

Future advanced Python classes

Jupyter Notebook



CMD.exe Prompt
0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated
Launch

JupyterLab
2.2.6
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch

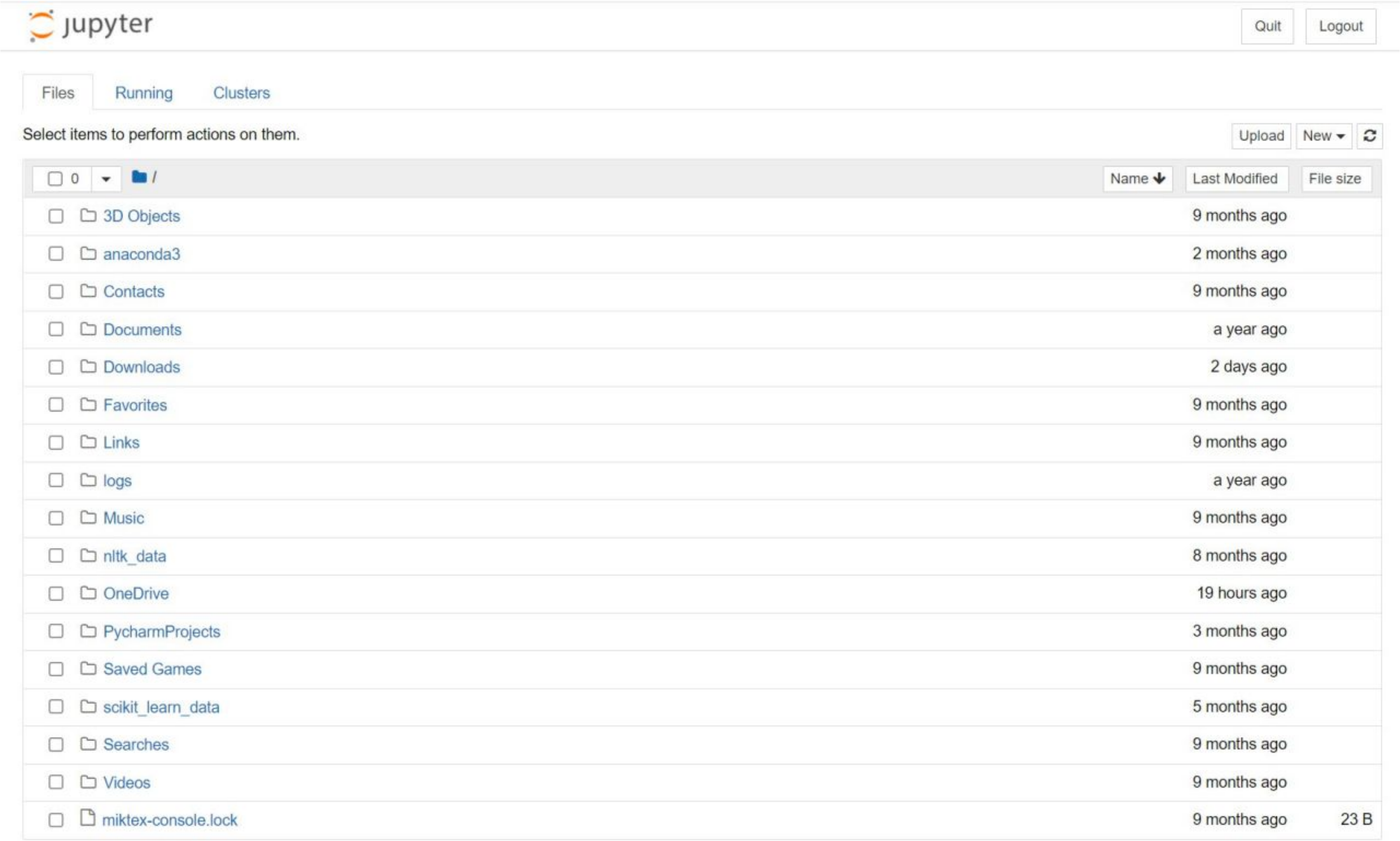
Notebook
↗ 6.1.6
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch

Powershell Prompt
0.0.1
Run a Powershell terminal with your current environment from Navigator activated

PyCharm
2021.1.1
Full-featured Python IDE by JetBrains. Supports code completion, linting, debugging, and domain-specific

Qt Console
4.7.7
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Navigating File Directory

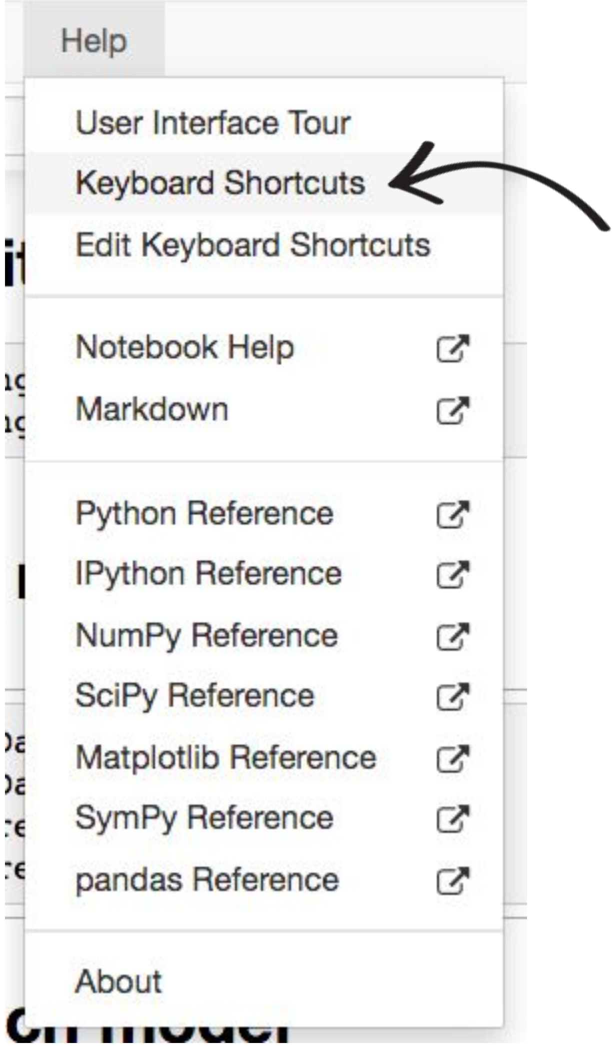


Select items to perform actions on them.

Name ↓	Last Modified	File size
□ 3D Objects	9 months ago	
□ anaconda3	2 months ago	
□ Contacts	9 months ago	
□ Documents	a year ago	
□ Downloads	2 days ago	
□ Favorites	9 months ago	
□ Links	9 months ago	
□ logs	a year ago	
□ Music	9 months ago	
□ nltk_data	8 months ago	
□ OneDrive	19 hours ago	
□ PycharmProjects	3 months ago	
□ Saved Games	9 months ago	
□ scikit_learn_data	5 months ago	
□ Searches	9 months ago	
□ Videos	9 months ago	
□ miktex-console.lock	9 months ago	23 B

How to Use Keyboard Shortcuts

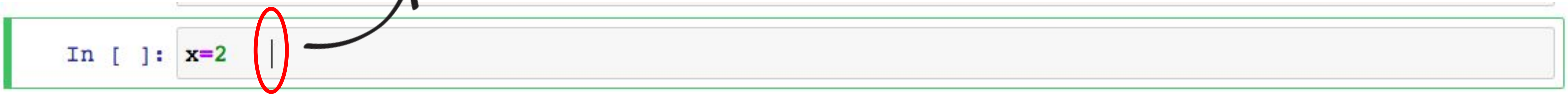
(Trust me, they’re a lifesaver)



How to Use Keyboard Shortcuts

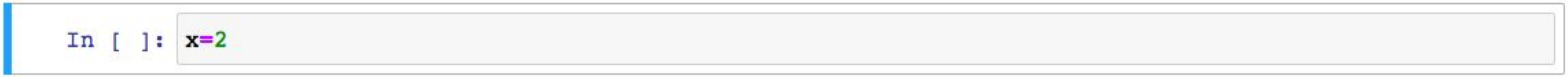
Edit Mode

Your cursor



```
In [ ]: x=2
```

Command Mode



```
In [ ]: x=2
```

Different keyboard shortcuts can be used in each mode

Useful Keyboard Shortcuts In Command Mode

Add a new code cell ABOVE the current cell: Press “a”
Add a new code cell BELOW the current cell: Press “b”

How to Enter Command Mode

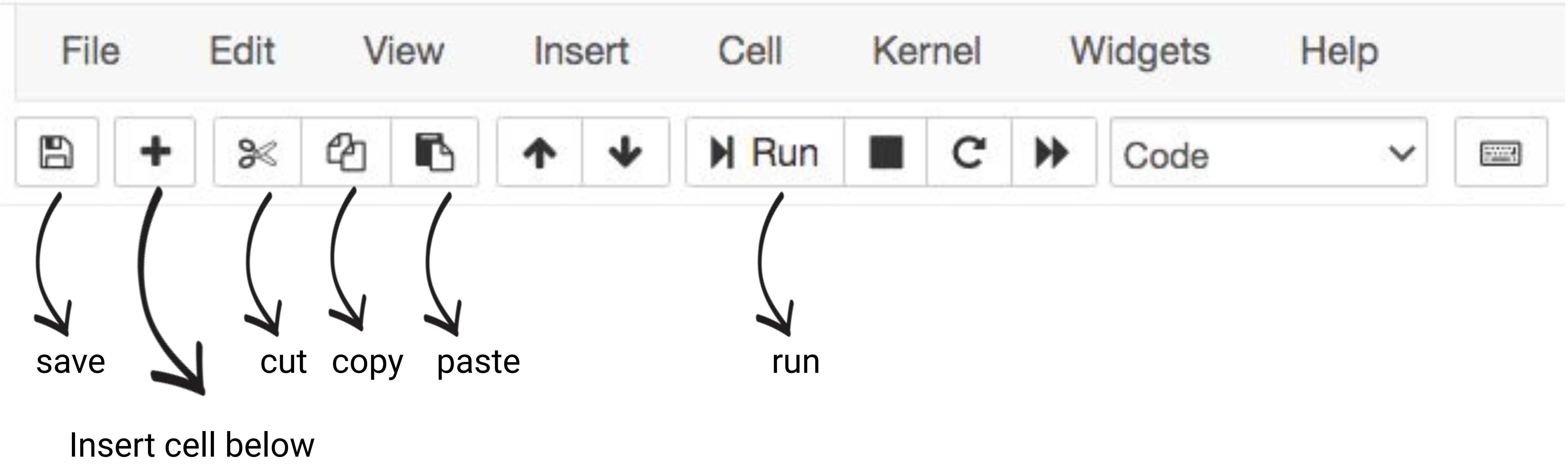
Press “Esc” on your keyboard
OR
Click anywhere outside of the cell you’re typing in

Useful Keyboard Shortcuts In Both Modes

Run a code cell: “Control” + “Enter”

Run a code cell and select cell below: “Shift” + “Enter”

Keyboard shortcuts too confusing?



save

cut

copy

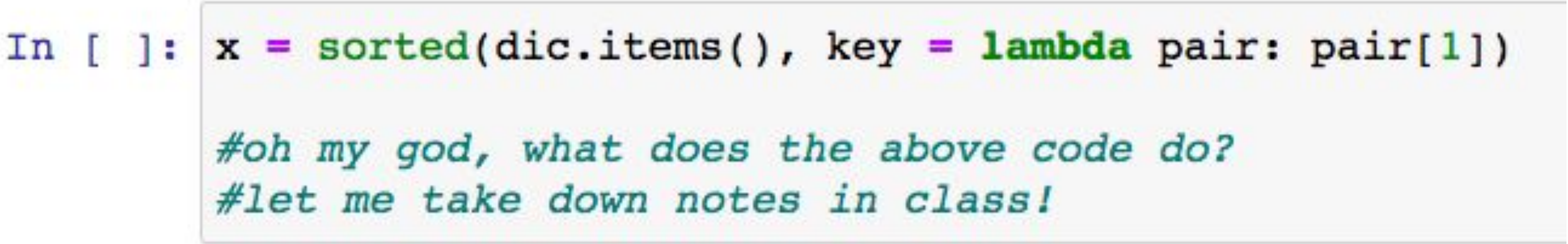
paste

run

Insert cell below

Adding Comments

= How to note-take on Jupyter Notebook!



```
In [ ]: x = sorted(dic.items(), key = lambda pair: pair[1])

        #oh my god, what does the above code do?
        #let me take down notes in class!
```

Printing

Definition

Displays the value of a variable

More on this later!

```python
print('Hello World')
```

```
Hello World
```

Printing in Jupyter Notebook

Jupyter Notebook automatically prints the last line of the code.
If you want an output to be printed, best to explicitly call `print()`.

```
print('Hello World again')
'Hello World'
```

```
Hello World again
'Hello World'
```

Printing in Jupyter Notebook

Jupyter Notebook automatically prints the last line of the code.
If you want an output to be printed, best to explicitly call `print()`.

```
'Hello World'
print('Hello World again')
```

```
Hello World again
```

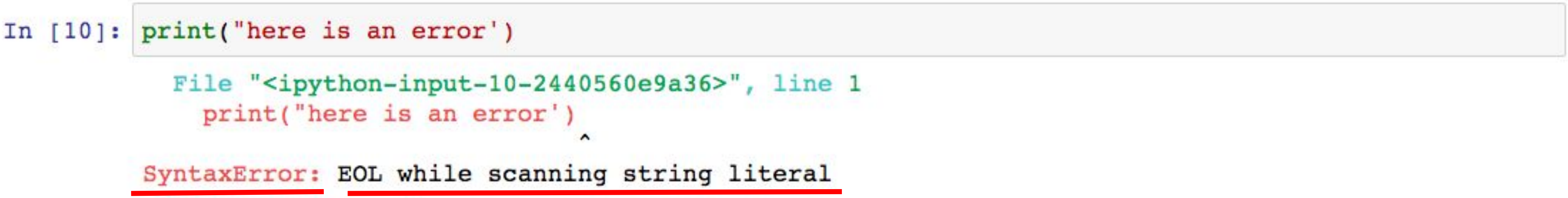
Why Quotes?

Anything in quotes is treated by Python as “human text” - It won't try to understand it as code!

```
print('Hey!')
```

Using single or double quotes is up to you , but you have to open and close the string with the same quote type

Debugging Errors

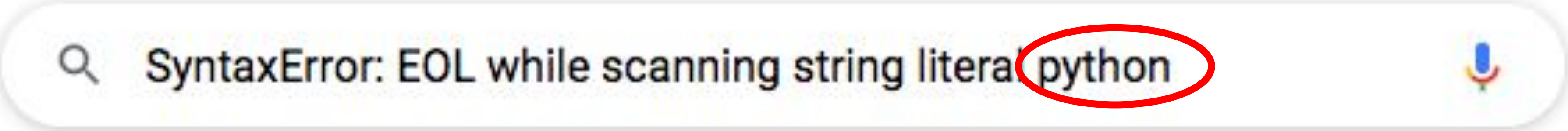


```
In [10]: print("here is an error')

           File "<ipython-input-10-2440560e9a36>", line 1
             print("here is an error')
                                      ^

         SyntaxError: EOL while scanning string literal
```

Error Category

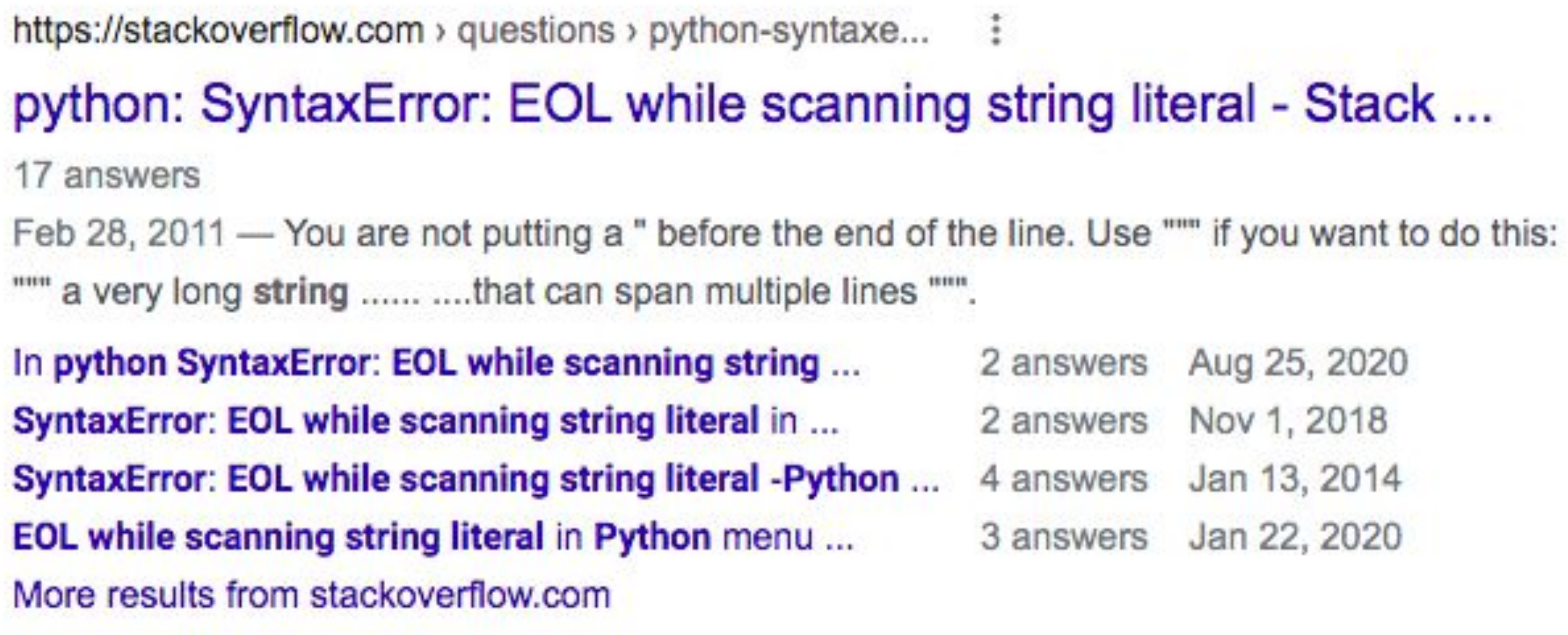
Explanation of Error

Still confused? **GOOGLE THE ERROR!**



SyntaxError: EOL while scanning string literal python

Debugging Errors: Use StackOverflow



https://stackoverflow.com › questions › python-syntaxe...

python: SyntaxError: EOL while scanning string literal - Stack ...

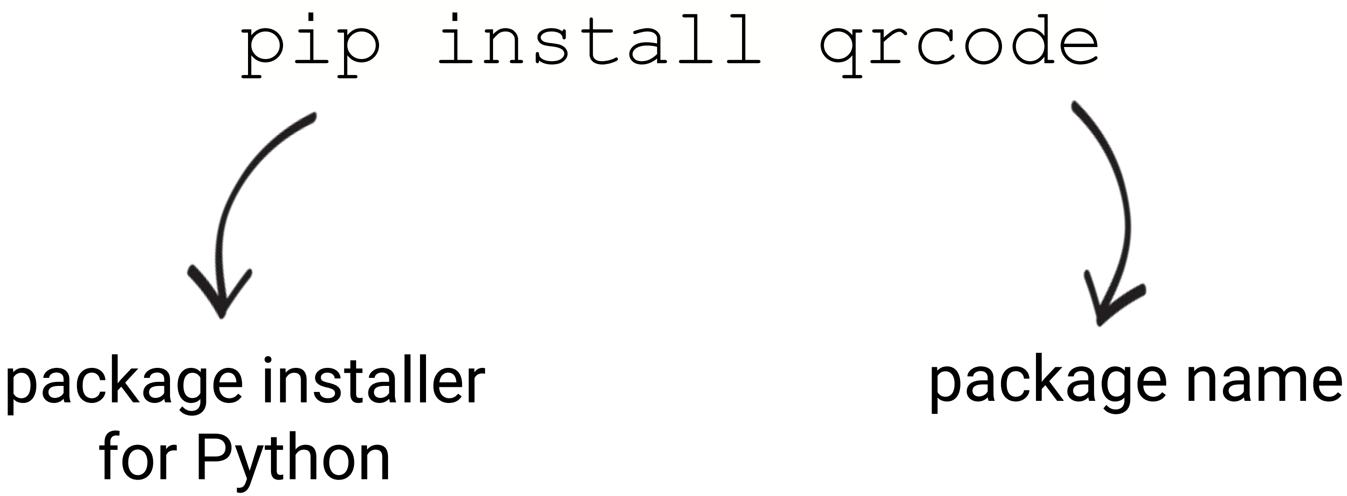
17 answers

Feb 28, 2011 — You are not putting a " before the end of the line. Use """ if you want to do this: """ a very long **string** ....that can span multiple lines """.

In **python SyntaxError: EOL while scanning string** ...	2 answers	Aug 25, 2020
SyntaxError: EOL while scanning string literal in ...	2 answers	Nov 1, 2018
SyntaxError: EOL while scanning string literal -Python ...	4 answers	Jan 13, 2014
EOL while scanning string literal in **Python** menu ...	3 answers	Jan 22, 2020

More results from stackoverflow.com

Using Python Packages



```
pip install qrcode
```

package installer for Python

package name

Note: You only need to install a package once

Using Python Packages

Python requires packages to be explicitly imported before you can use them, using the syntax: `import package_name`

- There are thousands of packages built by Python users online! (another reason to learn Python)

Let’s out try a handy Python package

```
!pip install qrcode
import qrcode
import matplotlib.pyplot as plt
url = (fill_in_your_url_here)
img = qrcode.make(url)
plt.imshow(img)
```

Importing packages using an alias

Variables

What is a Variable?

Definition

Associating a value with a name

```
num_rest_days = 5
num_work_days = 3
```

Variable Assignment

```
print(num_rest_days)
print(num_work_days)
```

```
5
3
```

Variable Assignment

```
num_rest_days = 5
num_work_days = 3
num_days = num_rest_days + num_work_days

print(num_days)
```

```
8
```

Variable Assignment

Variable Re-Assignment

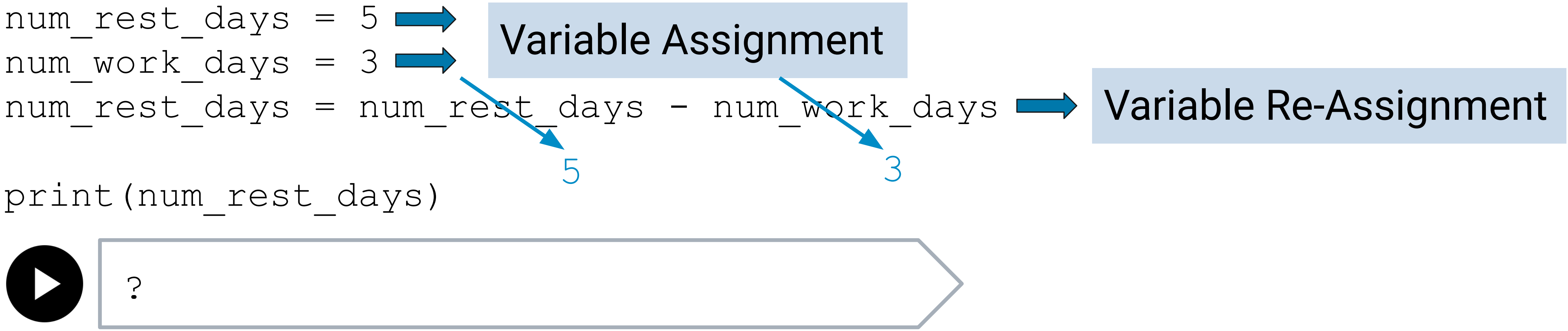
```
num_rest_days = 5
num_work_days = 3
num_rest_days = num_rest_days - num_work_days

print(num_rest_days)
```

Variable Assignment

Variable Re-Assignment

```
?
```

Variable Re-Assignment



```
num_rest_days = 5
num_work_days = 3
num_rest_days = num_rest_days - num_work_days

print(num_rest_days)
```

Variable Assignment

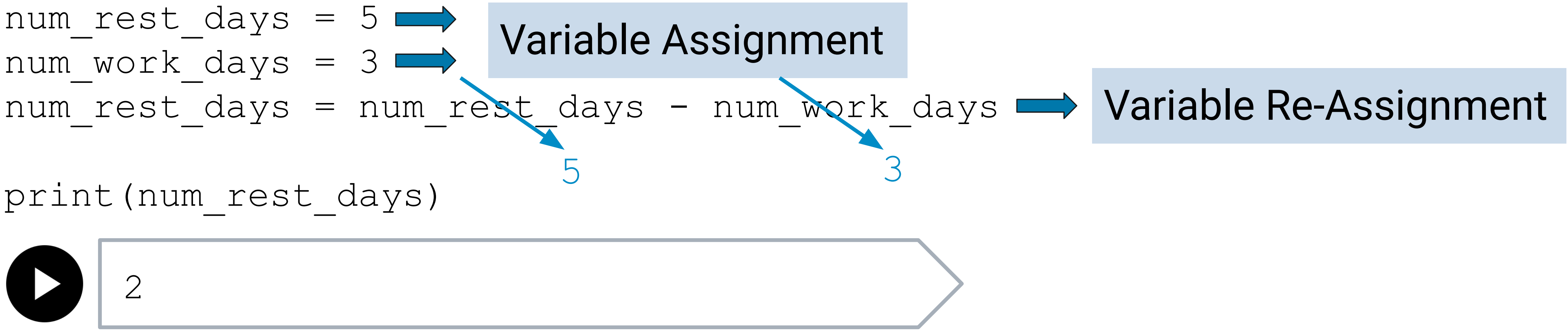
Variable Re-Assignment

5

3

```
?
```

Variable Re-Assignment



```
num_rest_days = 5
num_work_days = 3
num_rest_days = num_rest_days - num_work_days

print(num_rest_days)
```

Variable Assignment

Variable Re-Assignment

5

3

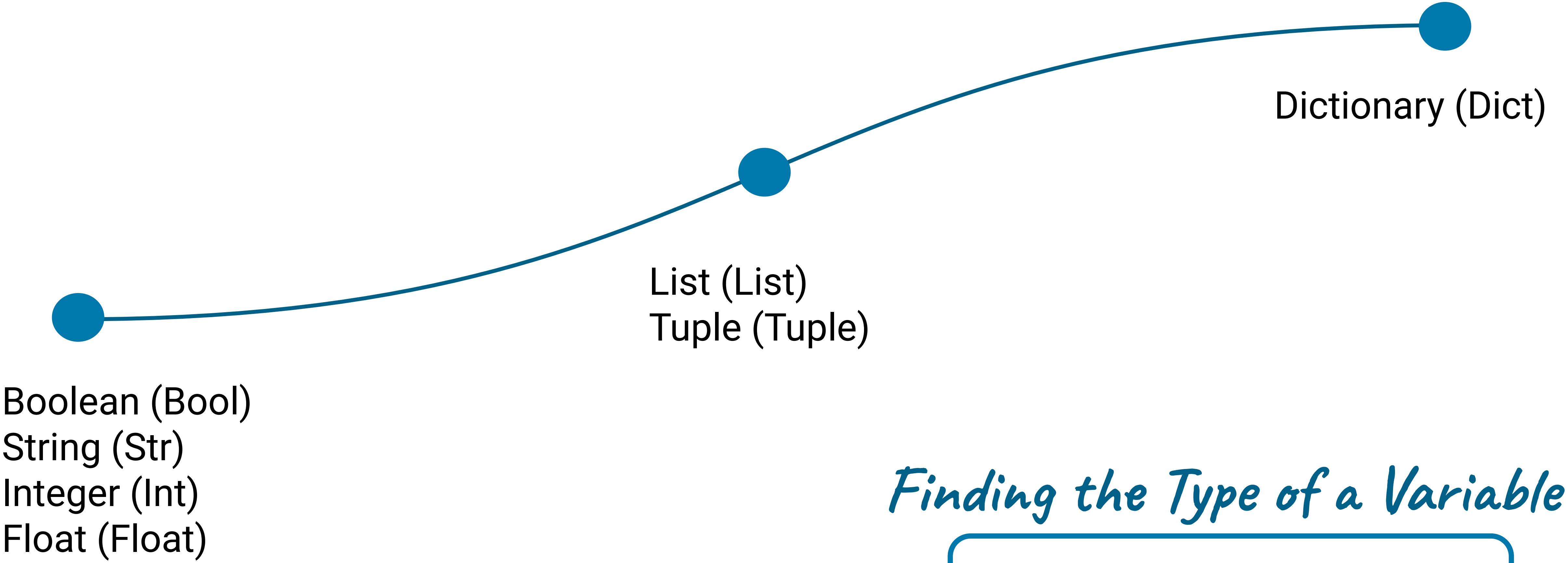
```
2
```

Variable Types

16

16
handles

Variable Types



Boolean (Bool)
String (Str)
Integer (Int)
Float (Float)

List (List)
Tuple (Tuple)

Dictionary (Dict)

Finding the Type of a Variable

```
type(variable_name)
```

What is a String?

Definition

A data type that represents a sequence of characters

Characters can be letters, digits, symbols (@, &, /), etc!

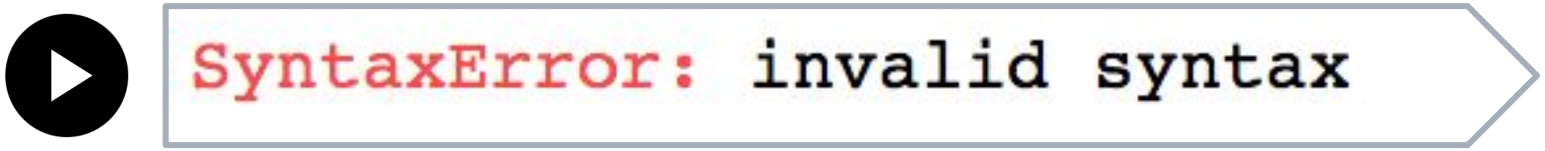
How to Create a String

```
print('Hey!')

print("Hello.")
```

A string is any sequence of characters enclosed in single or double quotes (Interchangeable!)

How to Create a String

```
print('I'm great')
```



```
SyntaxError: invalid syntax
```

We typically use single quotes to enclose strings, unless the string itself contains a single quote, in which case we use double quotes:

```
print("I'm great")
```

“Adding” (aka “concatenating” strings)

```
print("Romeo" + " & " + "Juliet")
```

```
Romeo & Juliet
```

Why types matter...

```
print(6 + 6)
```

```
12
```

```
print('6' + '6')
```

```
66
```

```
print('The number is ' + 6)
```

```
☠️
```

Converting between types

```
int('6')
```

▶

```
6
```

```
str(6)
```

▶

```
'6'
```

```
int("Ceci n'est pas un chiffre")
```

▶ ☠️

Converting between types

```
print('The number is ' + 6)
```

☠️

```
print('The number is ' + str(6))
```

```
'The number is 6'
```

Deep Dive: Strings

How to Create a String: Using f-strings

```
drink = 'Gin and tonic'
print(f'Time for a {drink}!')
```

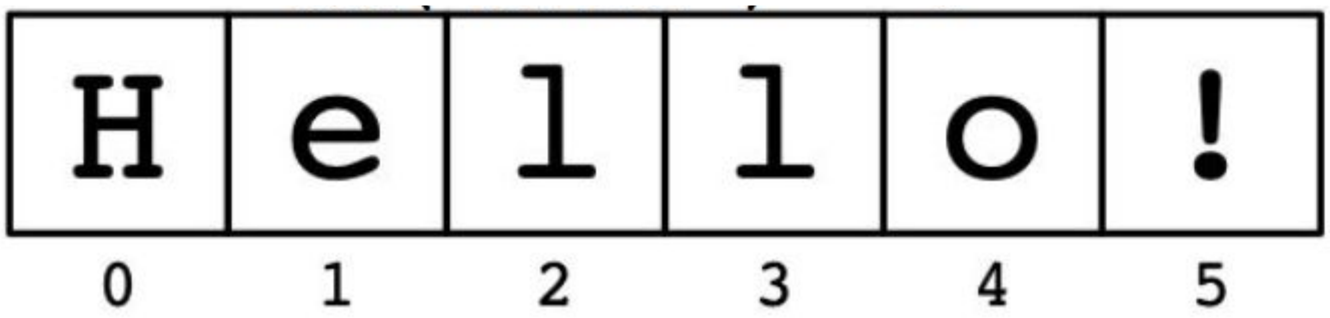
```
Time for a Gin and tonic!
```

String Indexing

Index

An integer representing the location of a character in a string

In Python, indexes start from 0!

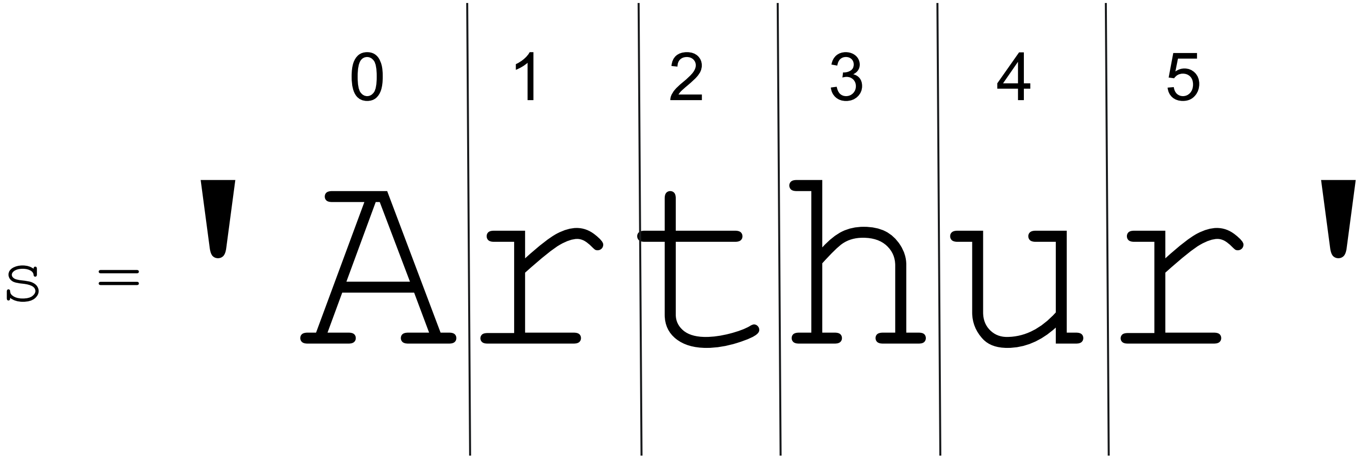


H	e	l	l	o	!
0	1	2	3	4	5

String Indexing

Bracket Notation

A character at index ***i*** of a string named ***s*** can be accessed using bracket notation: ***`s[i]`***



0	1	2	3	4	5
A	r	t	h	u	r

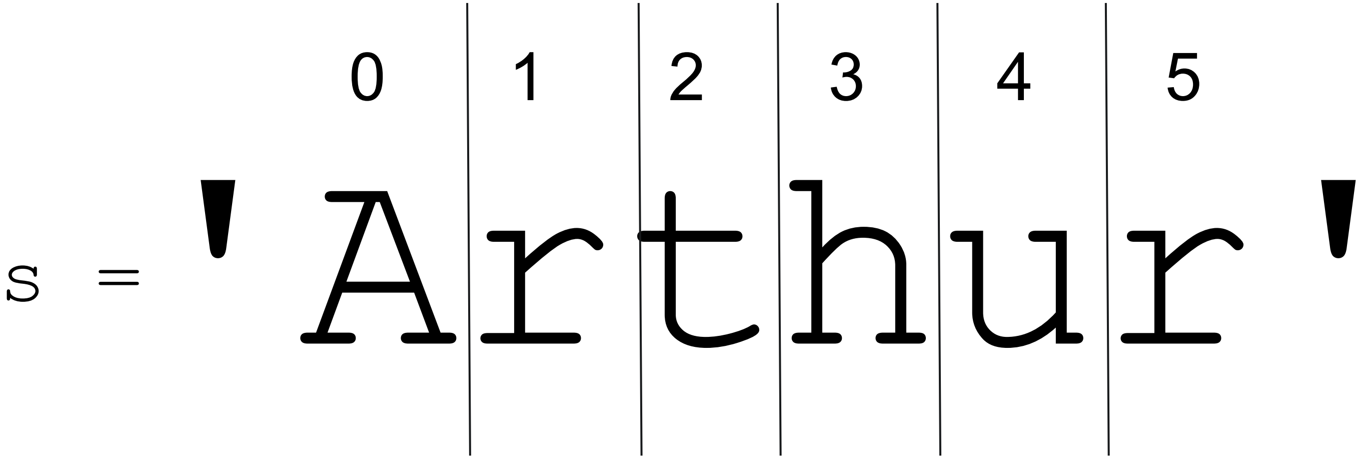
```
s = 'Arthur'
```

```
s[2] = ?
```

String Indexing

Bracket Notation

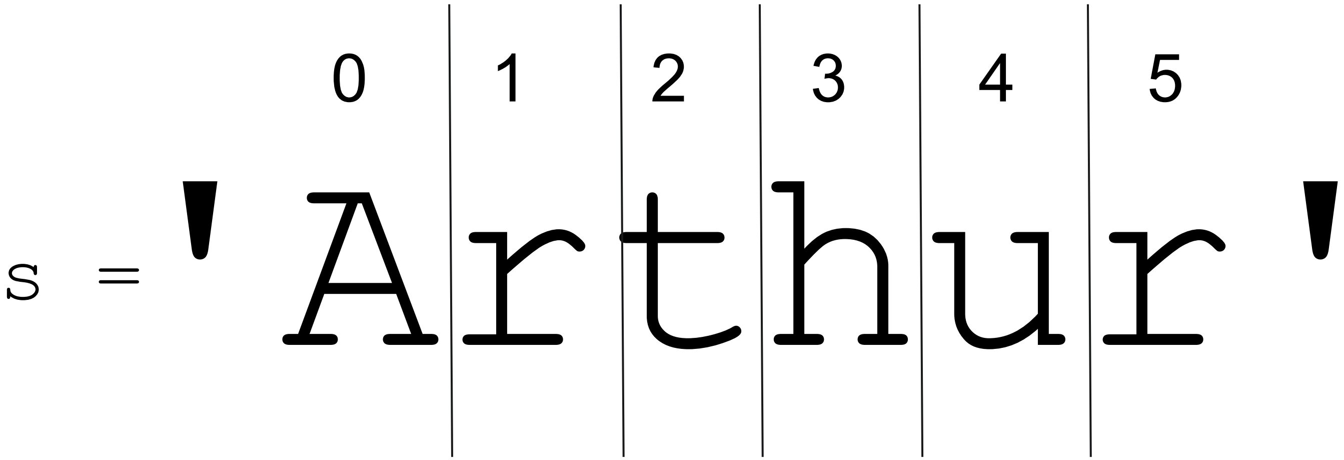
A character at index ***i*** of a string named ***s*** can be accessed using bracket notation: ***`s[i]`***



```
        0  1  2  3  4  5
s = 'Arthur'
```

```
s[2] = 't'
```

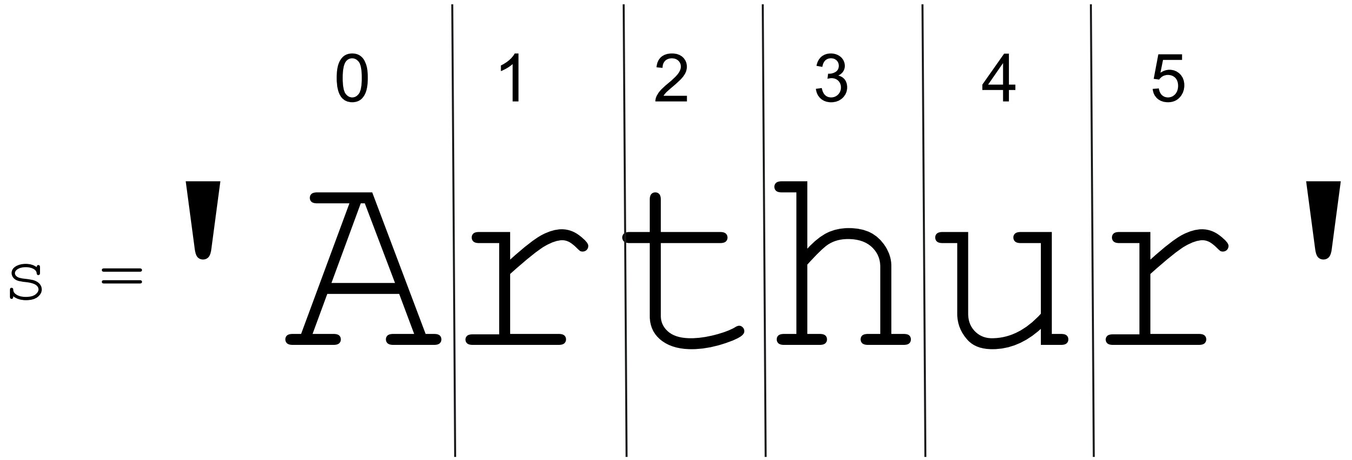
String Indexing



```
s[4] = ?
s[5] = ?
s[-1] = ?
s[-2] = ?
```

We can also index a string backwards, starting from -1, and **decreasing** by 1 each time

String Indexing

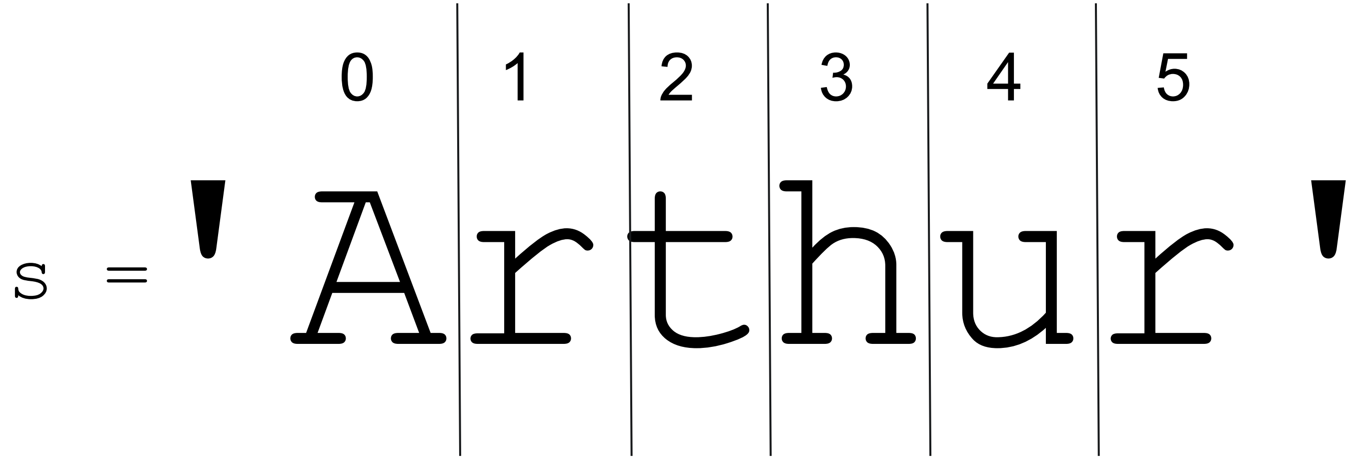


```
       0   1   2   3   4   5
s = ' A | r | t | h | u | r '
```

```
s[4] = 'u'
s[5] = ?
s[-1] = ?
s[-2] = ?
```

We can also index a string backwards, starting from -1, and **decreasing** by 1 each time

String Indexing

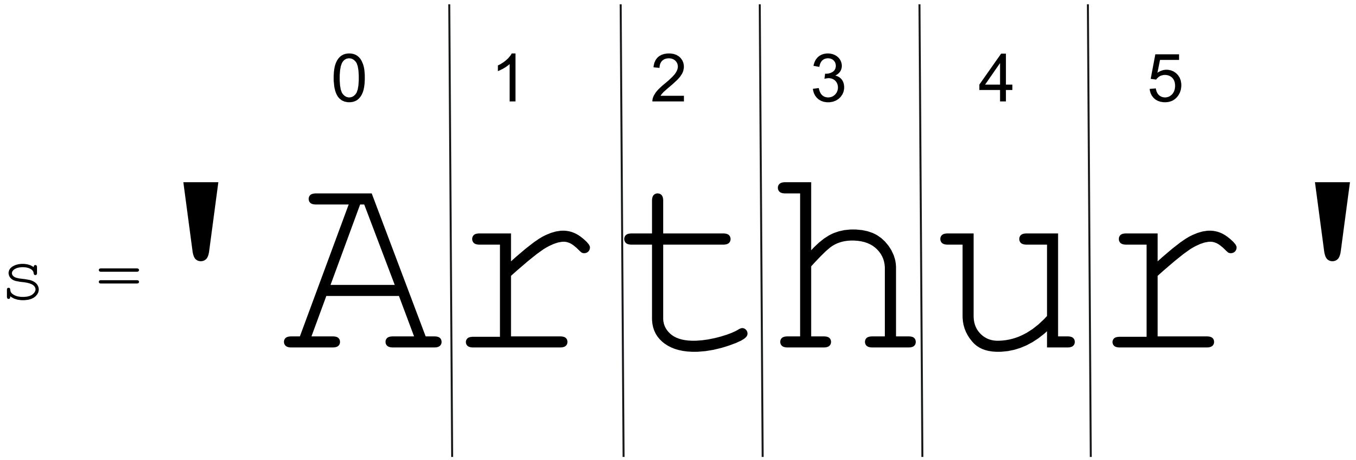


```
     0 1 2 3 4 5
s = 'A r t h u r'
```

```
s[4] = 'u'
s[5] = 'r'
s[-1] = ?
s[-2] = ?
```

We can also index a string backwards, starting from -1, and **decreasing** by 1 each time

String Indexing



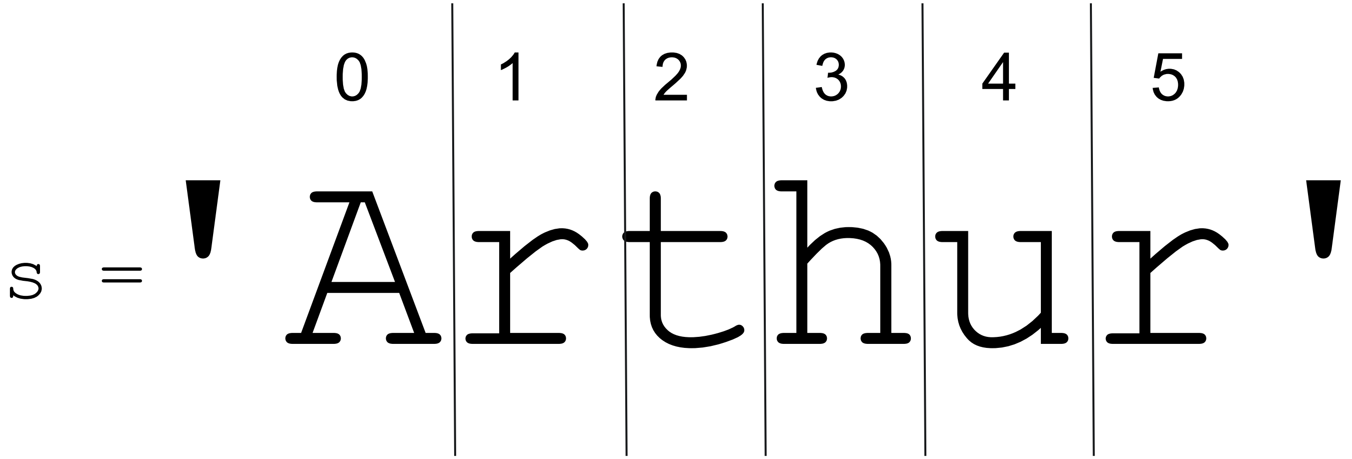
0	1	2	3	4	5
A	r	t	h	u	r

```
s = 'Arthur'
```

```
s[4] = 'u'
s[5] = 'r'
s[-1] = 'r'
s[-2] = ?
```

We can also index a string backwards, starting from -1, and **decreasing** by 1 each time

String Indexing



```
s[4] = 'u'
s[5] = 'r'
s[-1] = 'r'
s[-2] = 'u'
```

We can also index a string backwards, starting from -1, and **decreasing** by 1 each time

String Indexing

Finding the Length of a String

```
len(string_name)
```

= How many characters are in it?

```
string1 = 'hi'
string2 = ''
string3 = 'strings are cool!'

len(string1) = ?
len(string2) = ?
len(string3) = ?
```

String Indexing

Finding the Length of a String

```
len(string_name)
```

```
string1 = 'hi'
string2 = ''
string3 = 'strings are cool!'

len(string1) = 2
len(string2) = ?
len(string3) = ?
```

String Indexing

Finding the Length of a String

```
len(string_name)
```

```
string1 = 'hi'
string2 = ''
string3 = 'strings are cool!'

len(string1) = 2
len(string2) = 0
len(string3) = ?
```

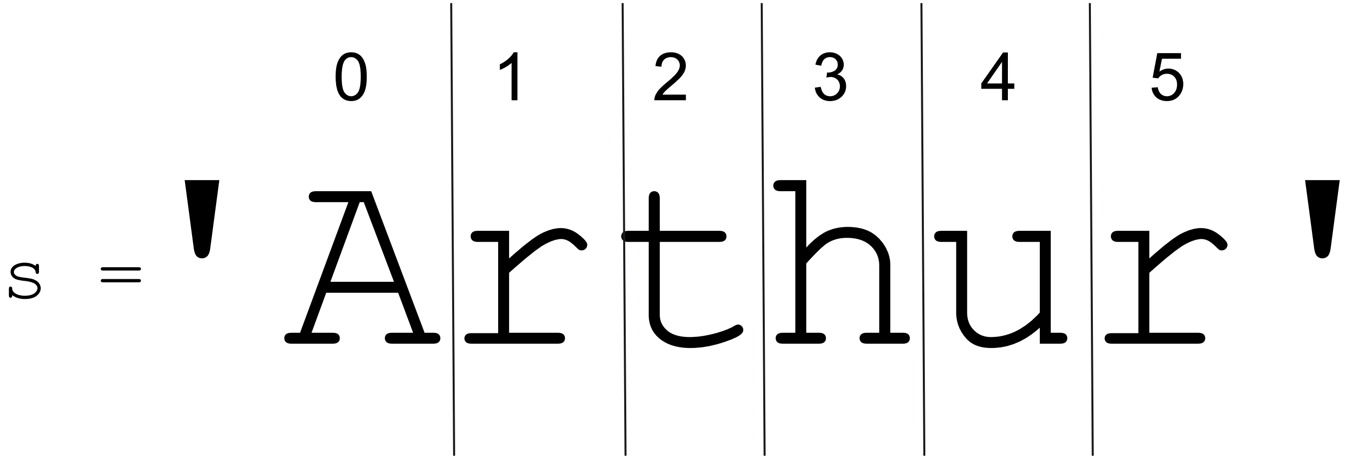
String Indexing

Finding the Length of a String

```
len(string_name)
```

```
string1 = 'hi'
string2 = ''
string3 = 'strings are cool!'

len(string1) = 2
len(string2) = 0
len(string3) = 17
```

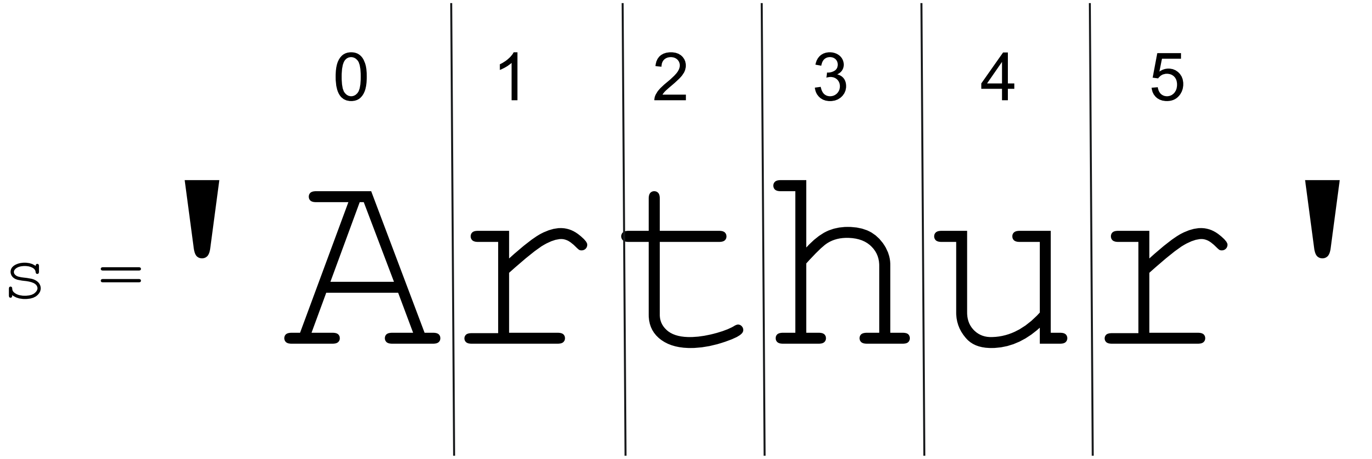
String Indexing



```
    0   1   2   3   4   5
s = 'Arthur'
```

```
len(s) = 6
s[len(s)] = ?
s[len(s)-1] = ?
```

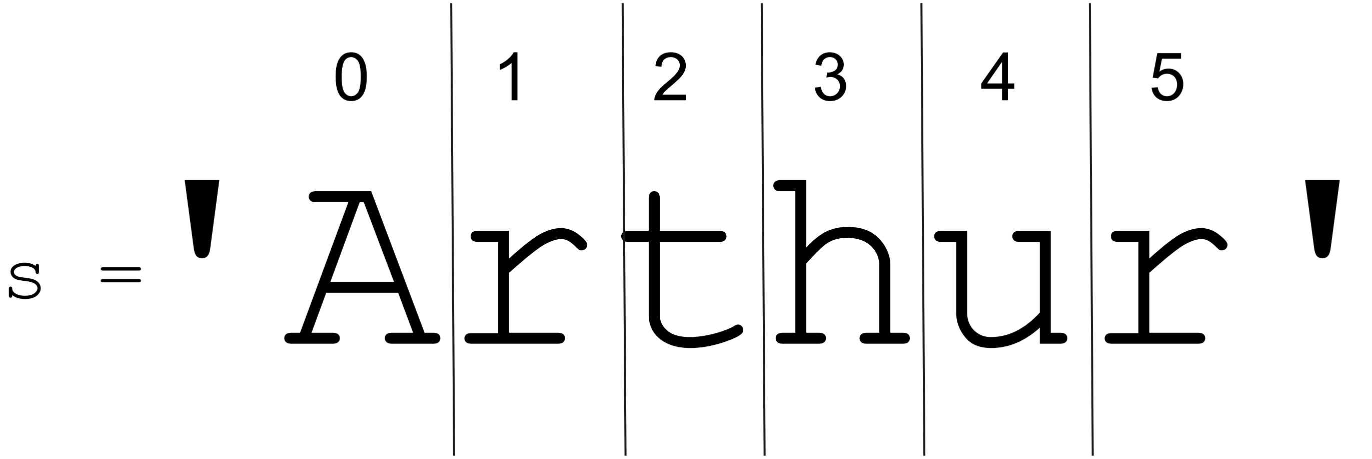
String Indexing



```
   0   1   2   3   4   5
s = 'Arthur'
```

```
len(s) = 6
s[len(s)] = IndexError
s[len(s)-1] = ?
```

String Indexing



```
 0  1  2  3  4  5
s = 'Arthur'
```

```
len(s) = 6
s[len(s)] = IndexError
s[len(s)-1] = 'r'
```

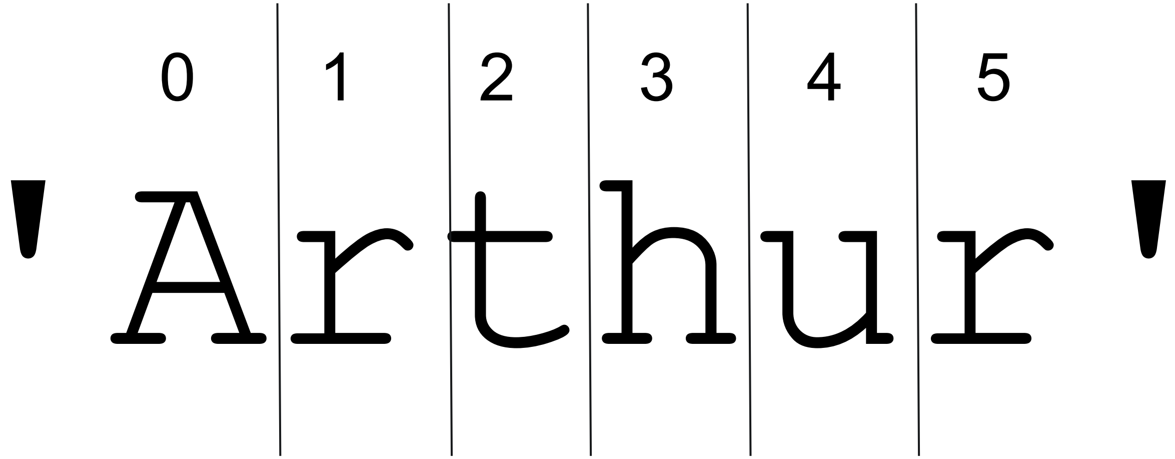
String Slicing

String indexing uses bracket notation with 1 index given, eg: **`s[2]`**
String slicing uses bracket notation with 2 indexes given, eg: **`s[2:4]`**

String Slicing

The resulting substring contains all characters starting from the first index given, up to but ***NOT*** including, the last index given

String Slicing

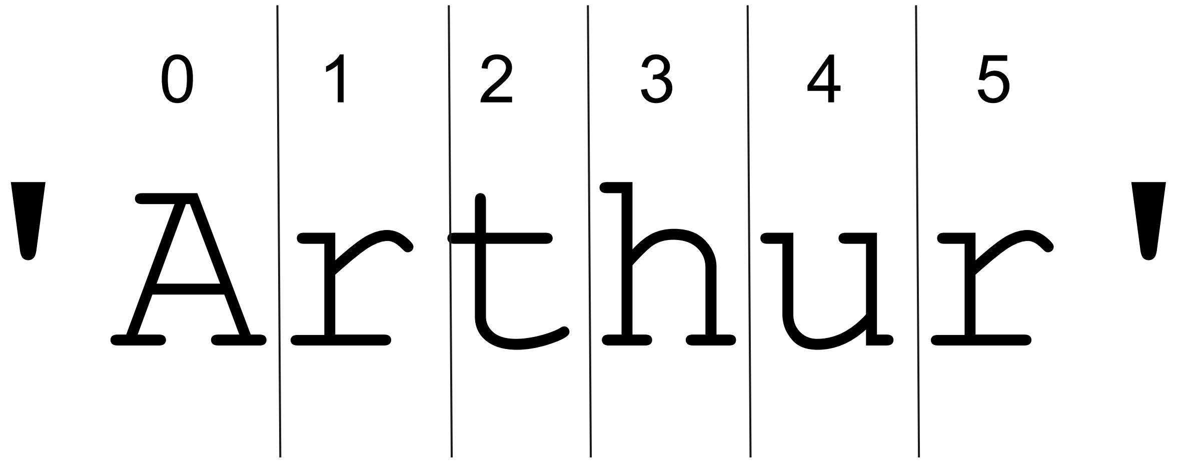


0	1	2	3	4	5
A	r	t	h	u	r

'Arthur'

```
s[2:4] = ?
```

String Slicing



0 1 2 3 4 5

'Arthur'

```
s[2:4] = 'th'
```

String Slicing

```
drink = 'Gin and tonic'
new_drink = drink[0:3]
print(new_drink)
```

```
Gin
```

String Slicing

```
drink = 'Gin and tonic'
print(drink[:3])
print(drink[4:])
print(drink[:])
```

```
Gin
```

```
and tonic
```

```
Gin and tonic
```

If the first index is not given, Python starts at the first character
If the last index is not given, Python ends at the last character

Try it!

UNITED STATES POSTAL SERVICE

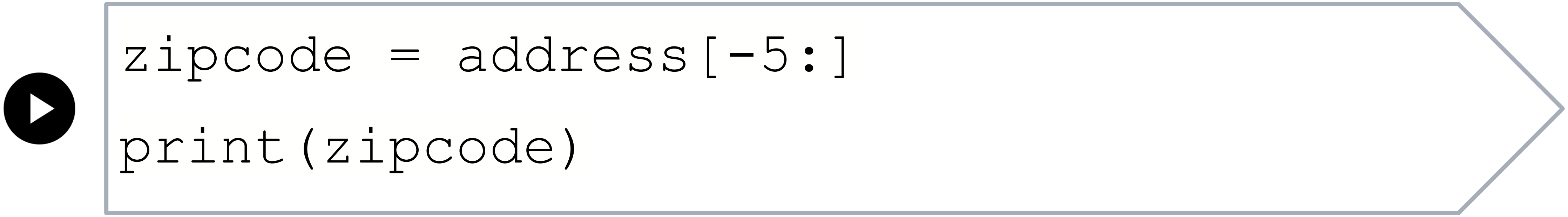
The USPS is responsible for the timely provision of postal services throughout the United States. A key step to ensure timely delivery is to sort all deliveries by Zip code so the sorting centers can handle deliveries in the same geographical area together.

Put your newly acquired Python knowledge to the test by extracting the Zip code from delivery addresses so that customers do not need to wait any longer for their parcels.

Example input: `address = '3022 Broadway, New York, NY 10027'`

Example output (to be printed): `'10027'`

Solutions



```
zipcode = address[-5:]
print(zipcode)
```

Note: The code works here because the data is well-structured, showing the importance of having good data.

String Splitting

Splitting a String by a Character

```
string_name.split(character)
```

```
quote = 'My bounty is as boundless as the sea; my love as deep.'
quote_split_by_whitespace = quote.split(' ')
print(quote_split_by_whitespace)
```

String Splitting

```
print(quote_split_by_whitespace)
```

```
['My', 'bounty', 'is', 'as', 'boundless', 'as', 'the', 'sea;', 'my',
'love', 'as', 'deep.']
```

What is this? Try:

```
type(quote_split_by_whitespace)
```

```
list
```

Splitting a string returns an variable of list datatype
→ Useful for list manipulations we'll learn later on!

String Splitting

```
quote = 'My bounty is as boundless as the sea; my love as deep.'
quote_split_by_colon = quote.split(';')
print(quote_split_by_colon)
```

```
['My bounty is as boundless as the sea', ' my love as deep.']
```

String Splitting

```
['My bounty is as boundless as the sea', ' my love as deep.']
```

```
len(quote_split_by_colon)
```

```
2
```

`len()` can be used on more than just strings!

- Use it to find the number of elements inside any iterable in Python (more on this later)

String Splitting

```
quote = 'My bounty is as boundless as the sea; my love as deep.'
quote_split_by_phrase = quote.split('as the sea')
print(quote_split_by_phrase)
```

```
['My bounty is as boundless ', '; my love as deep.']
```

Joining The String Back Together

```
print(quote_split_by_whitespace)
```

```
['My', 'bounty', 'is', 'as', 'boundless', 'as', 'the', 'sea;', 'my',
'love', 'as', 'deep.']
```

Joining a String by a Character

```
(character).join(list_name)
```

Joining The String Back Together

```
print(quote_split_by_whitespace)
```

```
['My', 'bounty', 'is', 'as', 'boundless', 'as', 'the', 'sea;', 'my',
'love', 'as', 'deep.']
```

```
original_quote = (" ").join(quote_split_by_whitespace)
print(original_quote)
```

```
'My bounty is as boundless as the sea, my love as deep.'
```

String Methods

Methods typically follow the syntax **`noun.verb()`**

(with the exception of the **`join()`** we just saw)

1) `string_name.upper(), string_name.lower()`

Returns the same string but with all letters converted to uppercase or lowercase respectively. The original string remains unchanged!

2) `string_name.isupper(), string_name.islower()`

Returns `True` if all characters in the string are uppercase or lowercase respectively, `False` otherwise

String Methods

```
3) string_name.isalpha(), string_name.isdigit()
```

Returns `True` if all characters in the string are letters (a-z, A-Z) or digits (0-9) respectively, `False` otherwise

```
4) string_name.startswith(substring),
   string_name.endswith(substring)
```

Returns `True` if the string starts with or ends with the given substring respectively, `False` otherwise

String Methods

3) `string_name.replace(old, new)`

`old:` character to be replaced
`new:` character to replace it with
Replaces all occurrences of the old character with the new character
The original string remains unchanged!

4) `string_name.index(character)`

Returns the index at which the given character **first** appears in the string

String Methods

`5) string_name.find(old, new)`

Returns the index at which the given character **first** appears in the string (similar to `.index()`). If not found, it returns -1.

String Methods- Let’s test them out!

```
s = 'hi'
new_s = s.upper()
print(new_s)
```

```
'HI'
```

```
print(s)
```

```
'hi'
```

Variable ‘s’ remains unchanged!

```
s = '1234'
s.isalpha()
```

```
False
```

```
s.isdigit()
```

```
True
```

```
s = '1234a'
s.isalpha()
```

```
False
```

```
s.isdigit()
```

```
False
```

Try it!

You've been working hard on a report due tonight. As you go over your report, you realise that your keyboard has malfunctioned, causing the character '#' to appear randomly throughout. Write a piece of code to fix this problem.

Example input:

```
report = 'This rep#ort highlig#hts #the importance# of fisc#al prudence###.'
```

Example output (to be printed):

```
'This report highlights the importance of fiscal prudence.'
```

Solution

```
new_report = report.replace('#', '')
print(new_report)
```

Try it!

Try it! You are about to go on a Twitter rant, but first, we need to check if what you want to say satisfies the 280-character limit. Your code should return True if it satisfies and False if it does not.

```
len(rant) < 280
```

Try it!

You are about to send out a monthly email to your clients to update them about your company's latest products. Your clients’ emails are:

```
list_of_emails = ["ej9212@columbia.edu",
"sj4837@harvard.edu", "jk6666nyu.edu"]
```

Write code that checks if the database of emails you have is valid, ie. check if all email addresses contain the symbol ‘@’.

Let’s try another one!

```
pip install translate
from translate import Translator
translator = Translator(to_lang='fr')
translation = translator.translate('Something you wanted to know in French.')
print(translation)
```

```
Quelque chose que vous vouliez savoir en Français.
```

Importing Packages

Import specific functions from a package

```
from translate import Translator
translator = Translator(to_lang='fr')
```

Compare this with:

```
import translate
translator = translate.Translator(to_lang='fr')
```

using Translator function from translate package