

Python

Bootcamp 2021

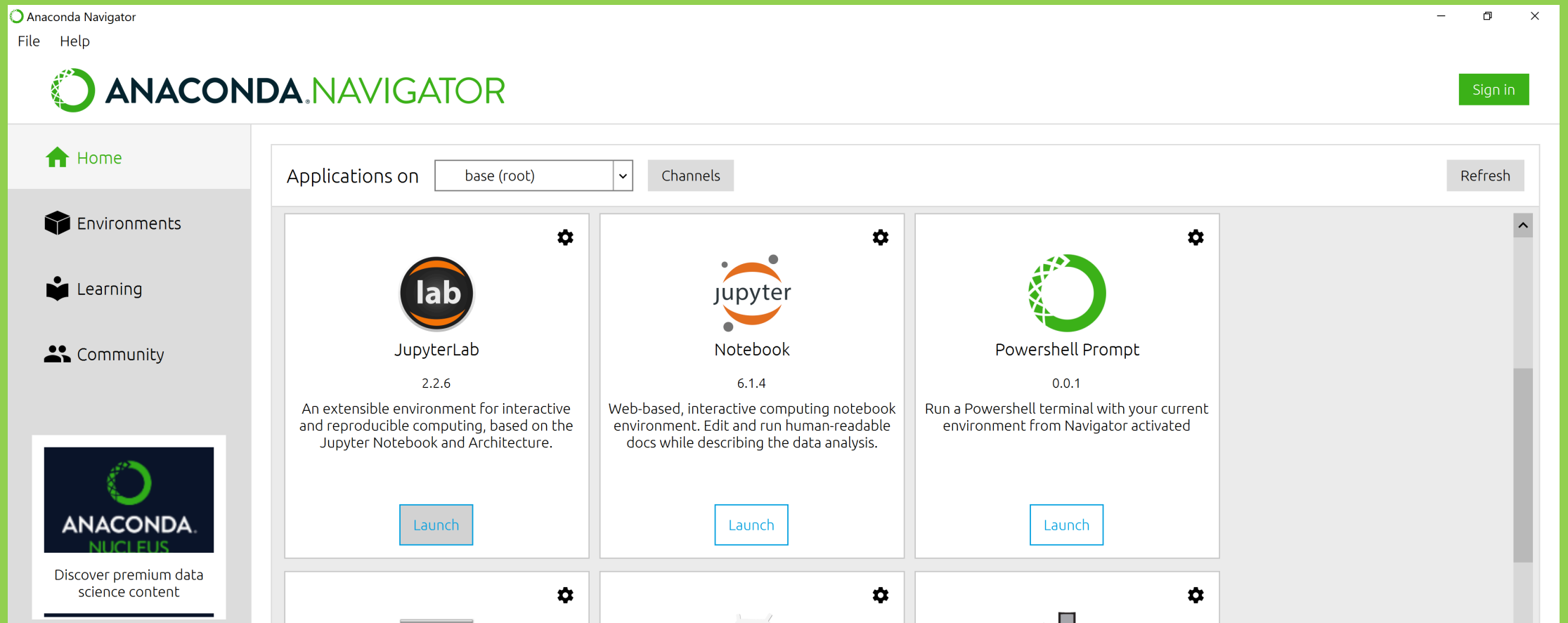
Outline for today

- Setup Anaconda
- Running Python
- Variables and Assignment
- Data Type
- Built-in functions
- Conditionals
- Loops

Outline for today

- Setup Anaconda ←
- Running Python
- Variables and Assignment
- Data Type
- Built-in functions
- Conditionals
- Loops

Setup Anaconda



Setup Anaconda

The screenshot displays the Anaconda Navigator application window. The top menu bar includes 'Anaconda Navigator', 'File', and 'Help'. The status bar at the top right shows system icons, a 91% battery level, the time 'Sun 3:06 PM', and the user 'Crisel Suarez'. The main interface features a sidebar on the left with navigation options: 'Home', 'Environments', 'Learning', and 'Community'. The 'Environments' section is active, showing a list of environments: 'base (root)', 'anaconda3', 'sun2', and 'sun3'. The 'sun3' environment is selected, and its details are shown in the main panel. The 'Installed' tab is active, displaying a table of installed packages.

ANACONDA NAVIGATOR Upgrade Now Sign in to Anaconda.org

Search Environments

base (root) anaconda3 sun2 **sun3**

Installed Channels Update index... Search Packages

Name	T	Description	Version
✓ _anaconda_depends	○		2018.12
✓ _ipyw_jlab_nb_ex...	○	A configuration metapackage for enabling anaconda-bundled jupyter extensions	0.1.0
✓ alabaster	○	Configurable, python 2+3 compatible sphinx theme.	0.7.12
✓ anaconda	○	Simplifies package management and deployment of anaconda	custom
✓ anaconda-client	○	Anaconda.org command line client library	1.7.2
✓ anaconda-project	○	Tool for encapsulating, running, and reproducing data science projects	0.8.2
✓ appnope	○	Disable app nap on os x 10.9	0.1.0
✓ appscript	○	Control applescriptable applications from python	1.0.1

Setup Anaconda

- >On your bash shell
- \$ conda create --name bootcamp2021
- proceed ([y]/n)?
- Y
- \$ conda info --envs
- \$ conda env list
- \$ conda activate bootcamp2021
- \$ conda list -n bootcamp2021
- \$ conda install package-name
- \$ conda install package-name=2.3.4
- <https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html>
- <https://docs.anaconda.com/anaconda/user-guide/tasks/install-packages/>
- \$conda create --name bootcamp2021 --clone base

Scripts /Spyder/Jupyter Notebook/JupyterLab

- All have pros/cons
- Choose what works best for you
- It is okay to switch between platforms

Python Scripts

- Run scripts on your bash shell

- \$python

```
>>>
```

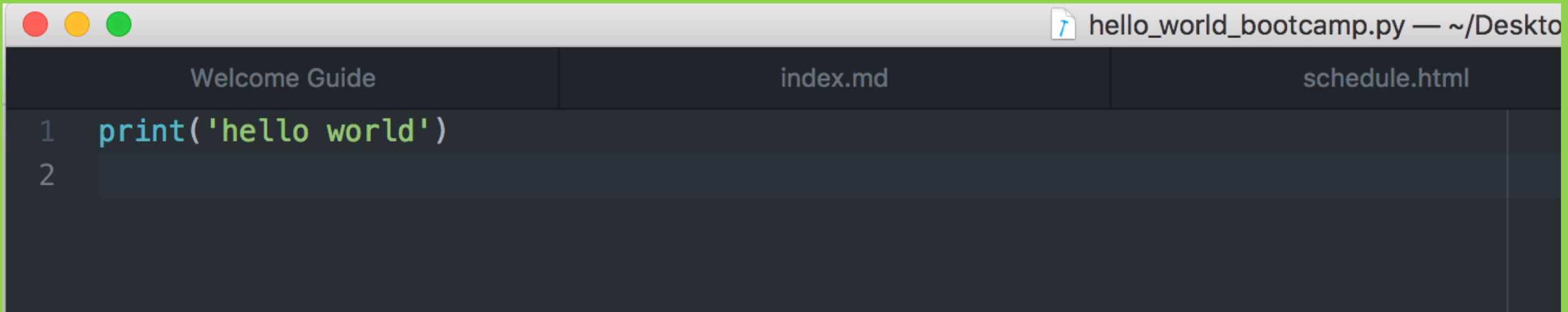
```
>>>print('hello world')
```

```
>>>exit() #Go back to your bash shell ($)
```

- \$ vim hello_world.py
- print('hello world')
- \$python hello_world.py

- vim
- Insert mode (i)
- Type your script/notes
- esc
- :wq

Python Scripts-Atom/Text Editor

A screenshot of the Atom text editor interface. The window title bar at the top shows three colored window control buttons (red, yellow, green) on the left and the file name 'hello_world_bootcamp.py — ~/Desktop' on the right. Below the title bar, there are three tabs: 'Welcome Guide', 'index.md', and 'schedule.html'. The main editing area has a dark background and shows a Python script with two lines. Line 1 is 'print('hello world')' and line 2 is empty. The line numbers '1' and '2' are visible on the left side of the editor.

```
1 print('hello world')
2
```

On your bash shell

```
$python hello_world_bootcamp.py
```

```
hello world
```

Spyder

Script
Code
goes here

The screenshot displays the Spyder Python IDE interface. The top menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. The toolbar contains icons for file operations and execution. The left pane shows the file explorer with the current file being edited: `8.8_dictionary_questions_UNRESOLVED.py`. The main editor displays Python code for two questions. The right pane shows the variable explorer with a table of declared variables. The bottom pane shows the IPython console with the output of the executed code.

Declared Variables

Name	Type	Size	Value
d_kv	dict	12	{1:0, 2:1, 3:1, 4:2, 5:3, 6:5, 7:8, 8:13, 9:21, ...}
i	int	1	12
n	int	1	12
x	int	1	144
y	int	1	233

Output

```
Python 3.9.0 (tags/v3.9.0:9cf6752, Oct 5 2020, 15:34:40)
[MSC v.1927 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more
information.

IPython 7.19.0 -- An enhanced Interactive Python.

In [1]:
...: n = 12 #number of iteration in the fibonacci
...: sequence
...: x = 0 #first position of fibonacci sequence
...: y = 1 #second position and impact of next sequence
...: d_kv = {} #same as creating d_kv = dict ()
...: for i in range (1, n+1): #iterating the sequence
```

Spyder

Run your code

The screenshot displays the Spyder Python IDE interface. The top toolbar contains various icons for file operations and code execution. A white arrow points to the 'Run' icon (a green play button). The main editor window shows a Python script with the following code:

```
43 Question 2
44 Write python code that will create a dictionary containir
45 that represent the first 12 values of the Fibonacci sequ
46 i.e {1:0,2:1,3:1,4:2,5:3,6:5,7:8 etc}
47 '''
48
49 n = 12 #number of iteration in the fibonacci sequence
50 x = 0 #first position of fibonacci sequence
51 y = 1 #second position and impact of next sequence
52 d_kv = {} #same as creating d_kv = dict ()
53
54 for i in range (1,n+1): #iterating the sequence starting
55     d_kv [i] = x # 1st sequence d_kv [1] = 0, d_kv [2] =
56     x,y = y, x+y
57
58 print (d_kv)
59
60
61
62
63 Question 3
64 Create a dictionary to represent the open, high, low, clc
65 for 4 imaginary companies. 'Python DS', 'PythonSoft', 'Py
66 the 4 sets of data are [12.87, 13.23, 11.42, 13.10],[23.5
67 [98.99,102.34,97.21,100.065],[203.63,207.54,202.43,205.2
68 '''
69 companies = ['Python DS', 'PythonSoft', 'Pythazon', 'Pybc
70 status = ['Open', 'High', 'Low', 'Close']
71 prices = [[12.87, 13.23, 11.42, 13.10],[23.54,25.76,21.8
72 [98.99,102.34,97.21,100.065],[203.63,207.54,202.43,205.2
73
74 d_sp = {}
75
```

The right-hand pane shows the 'Variable explorer' with a table of variables:

Name	Type	Size	Value
d_kv	dict	12	{1:0, 2:1, 3:1, 4:2, 5:3, 6:5, 7:8, 8:13, 9:21, ...}
i	int	1	12
n	int	1	12
x	int	1	144
y	int	1	233

The bottom pane shows the IPython console output:

```
Python 3.9.0 (tags/v3.9.0:9cf6752, Oct 5 2020, 15:34:40)
[MSC v.1927 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more
information.

IPython 7.19.0 -- An enhanced Interactive Python.

In [1]:
...: n = 12 #number of iteration in the fibonacci
sequence
...: x = 0 #first position of fibonacci sequence
...: y = 1 #second position and impact of next sequence
...: d_kv = {} #same as creating d_kv = dict ()
...:
...: for i in range (1,n+1): #iterating the sequence
```

The status bar at the bottom indicates 'LSP Python: ready', 'Kite: initializing', and 'custom (Python 3.9.0)'.

Spyder

Debug your
code

The screenshot displays the Spyder Python IDE interface. The top toolbar contains various icons for file operations and execution. A white box highlights the 'Run' button (a green play icon) in the toolbar. The main editor window shows a Python script with the following code:

```
43 Question 2
44 Write python code that will create a dictionary containing
45 the first 12 values of the Fibonacci sequence
46 d_kv = {1:0, 2:1, 3:1, 4:2, 5:3, 6:5, 7:8 etc}
47
48
49 n = 12 #number of iteration in the fibonacci sequence
50 x = 0 #first position of fibonacci sequence
51 y = 1 #second position and impact of next sequence
52 d_kv = {} #same as creating d_kv = dict ()
53
54 for i in range (1,n+1): #iterating the sequence starting
55     d_kv [i] = x # 1st sequence d_kv [1] = 0, d_kv [2] =
56     x,y = y, x+y
57
58 print (d_kv)
59
60
61
62
63 Question 3
64 Create a dictionary to represent the open, high, low, close
65 for 4 imaginary companies. 'Python DS', 'PythonSoft', 'Py
66 the 4 sets of data are [12.87, 13.23, 11.42, 13.10],[23.54,25.76,21.8
67 [98.99,102.34,97.21,100.065],[203.63,207.54,202.43,205.24]
68
69 companies = ['Python DS', 'PythonSoft', 'Pythazon', 'Pybc
70 status = ['Open', 'High', 'Low', 'Close']
71 prices = [[12.87, 13.23, 11.42, 13.10],[23.54,25.76,21.8
72 [98.99,102.34,97.21,100.065],[203.63,207.54,202.43,205.24]
73
74 d_sp = {}
75
```

The Variable explorer on the right shows the following variables:

Name	Type	Size	Value
d_kv	dict	12	{1:0, 2:1, 3:1, 4:2, 5:3, 6:5, 7:8, 8:13, 9:21, ...}
i	int	1	12
n	int	1	12
x	int	1	144
y	int	1	233

The IPython console at the bottom shows the execution of the code:

```
Python 3.9.0 (tags/v3.9.0:9cf6752, Oct 5 2020, 15:34:40)
[MSC v.1927 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more
information.

IPython 7.19.0 -- An enhanced Interactive Python.

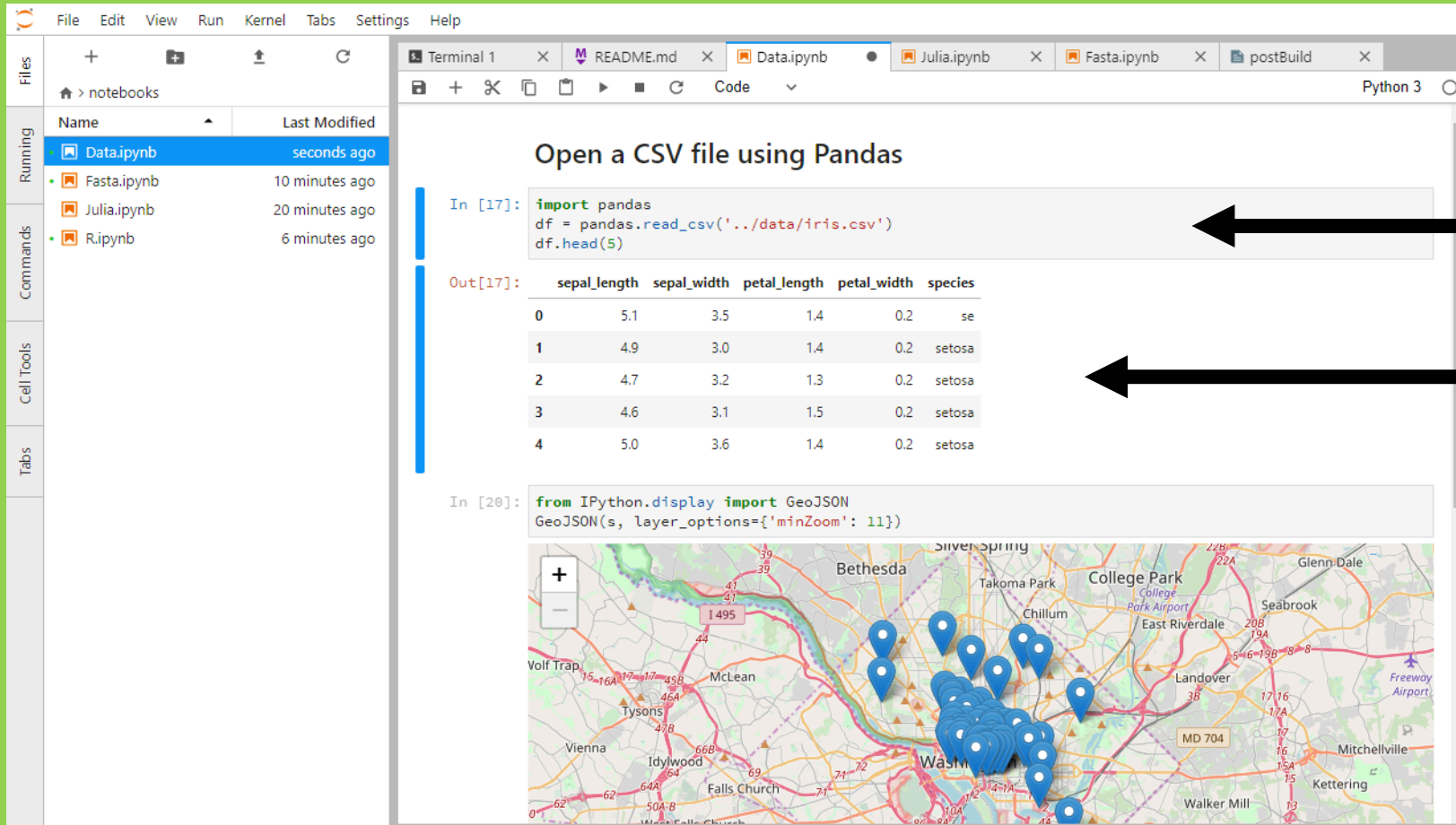
In [1]:
...: n = 12 #number of iteration in the fibonacci
sequence
...: x = 0 #first position of fibonacci sequence
...: y = 1 #second position and impact of next sequence
...: d_kv = {} #same as creating d_kv = dict ()
...:
...: for i in range (1,n+1): #iterating the sequence
```

The status bar at the bottom indicates the current file is 'custom (Python 3.9.0)' and the cursor is at Line 58, Col 13.

More on this later

Jupyter Lab (.ipynb)

\$ jupyter lab



The screenshot displays the Jupyter Lab web interface. On the left, a sidebar shows a file explorer with a list of notebooks: Data.ipynb (selected), Fasta.ipynb, Julia.ipynb, and R.ipynb. The main area shows the 'Data.ipynb' notebook with a code cell titled 'Open a CSV file using Pandas'. The code cell contains the following Python code:

```
In [17]: import pandas
df = pandas.read_csv('../data/iris.csv')
df.head(5)
```

The output of this cell is a table showing the first five rows of the 'iris.csv' file:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	se
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Below the table, there is another code cell titled 'GeoJSON' that uses the 'GeoJSON' library to display the data on a map. The map shows a geographical area with several blue location pins, indicating the spatial distribution of the data points.

Cell – Code

Output

To run a cell:
shift + enter

<https://jupyterlab.readthedocs.io/en/stable/user/notebook.html>

Jupyter notebook (.ipynb)

- When in Command mode (esc/gray),
 - The **b** key will make a new cell below the currently selected cell.
 - The **a** key will make one above.
 - The **x** key will delete the current cell.
 - The **z** key will undo your last cell operation (which could be a deletion, creation, etc).

Jupyter notebook (.ipynb)

- Markdown great for commenting/adding notes to your code!
- A simple plain-text format for writing lists, links, and other things that might go into a web page.

Turn the current cell into a **Markdown cell** by entering the Command mode (**Esc**) and press the **M** key.

In []: will disappear to show it is no longer a code cell and you will be able to write in Markdown.

Turn the current cell into a **Code** cell by entering the Command mode (**Esc**) and press the **y** key

Markdown – html

* Use asterisks
* to create
* bullet lists.



Lists

A Level-1 Heading



Headings

A Level-2 Heading (etc.)


[Create links](http://software-carpentry.org) with `...`.



urls + links

Or use [named links][data_carpentry]. [data_carpentry]:
http://datacarpentry.org

Outline for today

- Setup Anaconda
- Running Python
- Variables and Assignment ← 
- Data Type
- Built-in functions
- Packages
- Conditionals
- Loops

Variables and Assignments

- In Python the **=** symbol assigns the value on the right to the name on the left
- `age = 42`
- `my_name = 'Crisel Suarez'`
- `Grade1 = 'A'`
- Variable names
 - can **only** contain letters, digits, and underscore `_`
 - cannot start with a digit
 - are **case sensitive** (`age`, `Age` and `AGE` are three different variables)

Variables and Assignments

- `first_name = 'Kathy '`
- `age = 10`
- `print(first_name, 'is', age, 'years old')`
- Variables can be used in calculations:
 - `new_age = age +10`
- Indexing
- `print(first_name[0])`

*** Python indexing starts at 0 ***


Indexing and Slices

- `[start:stop]`
- `atom_name = 'sodium'`
- `print(atom_name[0:3])`
 - `> sod`
- `len(atom_name)`
- `6`

Key Points

- Use variables to store values.
- Use `print()` to display values.
- Variables persist between cells.
- Variables must be created before they are used.
- Variables can be used in calculations.
- Use an index to get a single character from a string.
- Use a slice to get a substring.
- Use the built-in function `len()` to find the length of a string.
- Python is case-sensitive.
- Use meaningful variable names


Outline for today

- Setup Anaconda
- Running Python
- Variables and Assignment
- Data Type ← 
- Lists
- Built-in functions
- Packages
- Conditionals
- Loops

Data Types

- `str()` – String
- `int()`- integer
- `Float()` - decimals
- `Type()` > What kind of data type

Outline for today

- Setup Anaconda
- Running Python
- Variables and Assignment
- Data Type ← 
- Lists
- Built-in functions
- Packages
- Conditionals
- Loops

Lists

- Storing multiple variables
- `pressures = [0.273, 0.275, 0.277, 0.275, 0.276]`
- `print('pressures:', pressures)`
- `print('length:', len(pressures))`
- `print('zeroth item of pressures:', pressures[0])`
- `pressures[0] = 0.265`
-

Lists – Appending

- `list_name.append()`
- `primes = [2, 3, 5]`
- `print('primes is initially:', primes)`
- `primes.append(7)`
- `print('primes has become:', primes)`

Lists – Deleting

- `del list_name[index]` to remove an element from a list
- `primes = [2, 3, 5, 7, 9]`
- `print('primes before removing last item:', primes)`
- `del primes[4]`
- `print('primes after removing last item:', primes)`

List- Empty []

- `Empty_list = []`
- Helpful as a starting point for collecting values

Practice:

- **print**('string to list:', list('tin'))
- **print**('list to string:', ''.join(['g', 'o', 'l', 'd']))

What does list do?

What does .join do?

*We will come back to list with Numpy's version ...arrays

Outline for today

- Setup Anaconda
- Running Python
- Variables and Assignment
- Data Type
- Lists
- Built-in functions ←
- Conditionals
- Loops

Built-in functions

- Think math function
 - $f(x) = x + 5$
 - $x \rightarrow$ input
 - $f(x) \rightarrow$ output
-
- Functions can take 0 or many arguments
 - `print()`
 - $f(x_1, x_2, x_3, \dots) = x_1 + x_2 + x_3 + \dots$

Built-in functions

- `max(1,2,3)`
- `min(5,6,7)`
- `round(3.712, 1)` #rounds to 1 decimal place
- `help(round)`

Functions attached to objects are called methods

- Methods have parentheses like functions, but come after the variable.

```
my_string = 'Hello world!' # creation of a string object
```

```
print(my_string.swapcase())
```

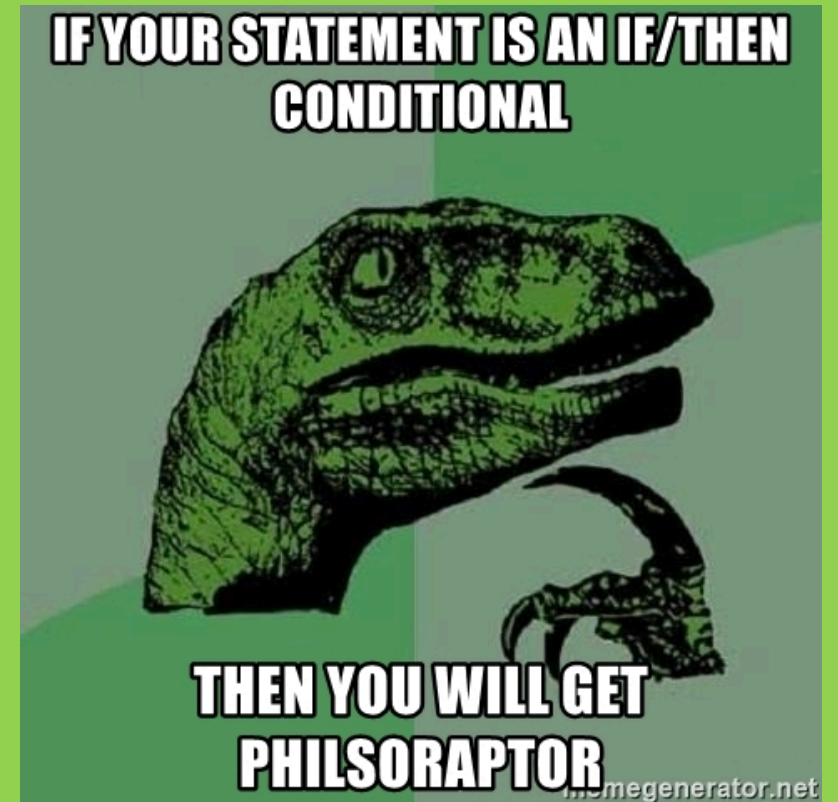
```
# calling the swapcase method on the my_string object
```

Outline for today

- Setup Anaconda
- Running Python
- Variables and Assignment
- Data Type
- Built-in functions
- Conditionals ←
- Loops

Conditionals

- if (condition is True):
 then do something
- if (condition is True):
 then do something
- else:
 - Do something else
- if (condition is True):
 then do something
- elif (this condition is true):
 - then do this
- else:
 - Do this



Conditionals – Try it out

- `mass = 3.4`
- `If mass > 3.0:`
 - `print('Mass is ', mass)`
- `if mass > 3:`
 - `print('Mass is less than 3')`
- `else:`
 - `print('Mass is more than 3')`
- `if mass < 3.7:`
 - `print('mass less than 3.7')`
- `elif (if mass > 3.2):`
 - `print('mass greater than 3.2')`
- `else:`
 - `print(mass greater than 3.7 or less than 3.2)`

Conditionals – Try it out

- mass = 3.4
- If ((mass < 3.7) and (mass >3.2)):
 - print(mass less than 3.7 or greater than 3.2)

- mass = 3.4
- If ((mass < 3.7) or (mass >3.2)):
 - print(mass less than 3.7 or greater than 3.2)

- mass = 3.8
- If ((mass < 3.7) and (mass >3.2)):
 - print(mass less than 3.7 or greater than 3.2)


- mass = 3.8
- If ((mass < 3.7) or (mass >3.2)):
 - print(mass less than 3.7 or greater than 3.2)

Conditionals

p	q	p and q
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
FALSE	TRUE	FALSE
FALSE	FALSE	FALSE

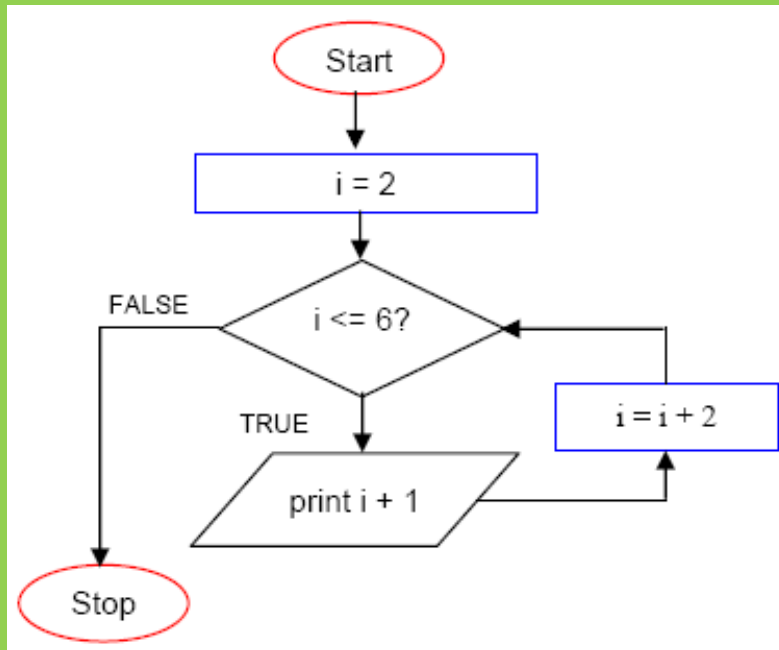
p	q	p or q
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE

Loops

- Setup Anaconda
 - Running Python
 - Variables and Assignment
 - Data Type
 - Built-in functions
 - Conditionals
 - Loops
- 

Loops

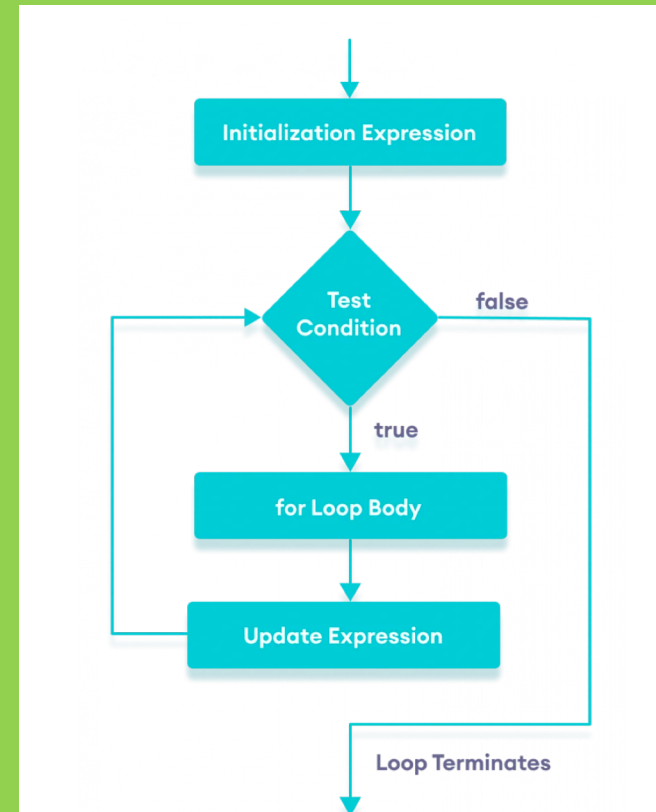
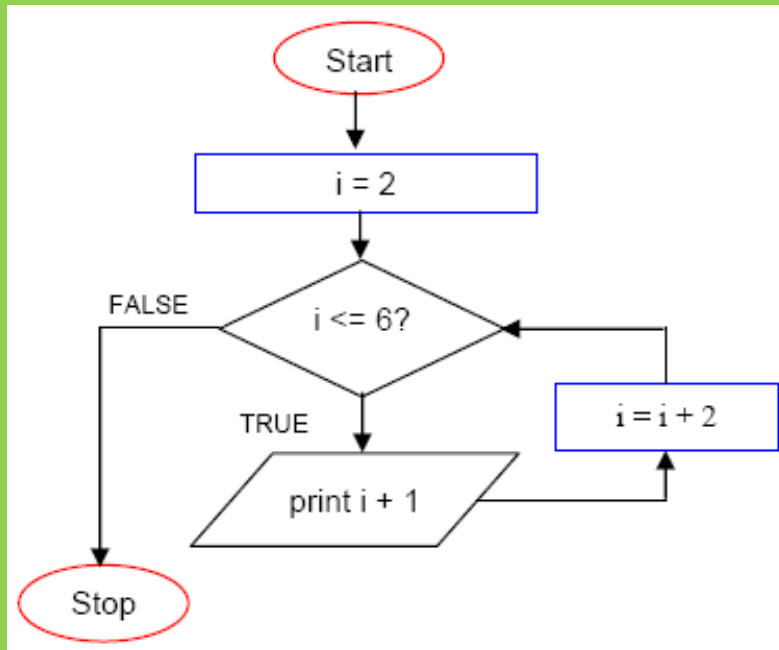
Loops are a programming construct which allow us to repeat a command or set of commands for each item in a list. As such they are key to productivity improvements through automation



i	i <= 6	Output
2	True	3
4	True	5
6	True	7
8	False	

Loops

Loops are a programming construct which allow us to repeat a command or set of commands for each item in a list. As such they are key to productivity improvements through automation



Loops

- **for** number **in** [2, 3, 5]:
 - **print**(number)
- primes = [2, 3, 5]
- **for** p **in** primes:
 - squared = p ** 2
 - cubed = p ** 3
 - **print**(p, squared, cubed)

Loops

- The built-in function [range](#) produces a sequence of numbers.
- Not a list: the numbers are produced on demand to make looping over large ranges more efficient.
- **print**('a range is not a list: range(0, 3)')
- **for** number **in** range(0, 3):
 - **print**(number)

Loops – Practice

- *# List of word lengths: ["red", "green", "blue"] => [3, 5, 4]*
- `lengths = _____`
- **for** word **in** ["red", "green", "blue"]:
 - `lengths.____(_____)`
- **print**(lengths)

Loops – Practice

- *# List of word lengths: ["red", "green", "blue"] => [3, 5, 4]*
- `lengths = []`
- **for** word **in** ["red", "green", "blue"]:
 - `lengths.append(len(word))`
- **print**(lengths)

Loops – Practice

- *# Concatenate all words: ["red", "green", "blue"] => "redgreenblue"*
- words = ["red", "green", "blue"]
- result = _____
- **for** _____ **in** _____:
 - _____
- **print**(result)

Loops – Practice

- *# Concatenate all words: ["red", "green", "blue"] => "redgreenblue"*
- `words = ["red", "green", "blue"]`
- `result = ""`
- **for** `word` **in** `words`:
 - `result = result+word`
- **print**(`result`)

