3803ICT
Data Analytics

Lab 02 – Python and best practices

Course Convenor:
Dr. Henry Nguyen

Trimester 1 - 2019

Table of Contents

I. Environment set-up: Anaconda

So this course uses the Python programming language throughout it. So the first thing we need to do is make sure you have a Python development environment set up on your personal computer. So we will walk through installing a package called Anaconda which has both the development environment and all the Python packages you need pre-installed. It makes life really easy.

1. Anaconda Distributition by Continuum Analytics

Go to the website: https://www.anaconda.com/download/
Download the installer (Python 3.6 version) for your OS
Run the installer

2. Using Anaconda

If you want to install a new package into your Anaconda, type:
conda install <package>

3. Run Jupyter

The Jupyter Notebook App can be launched by clicking on the Jupyter Notebook icon installed by Anaconda in the start menu (Windows) or by typing in a terminal (cmd on Windows):
jupyter notebook
This will launch a new browser window (or a new tab) showing the Notebook Dashboard, a sort of control panel that allows (among other things) to select which notebook to open.
When started, the Jupyter Notebook App can access only files within its start-up folder (including any sub-folder). If you store the notebook documents in a subfolder of your user folder no configuration is necessary. So make sure you go to the folder you want (*using cd <folder>* in terminal) before starting Jupyter (*jupyter notebook*).
For more configurations, see: http://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/execute.html

4. Create a Jupyter notebook

In the Notebook Dashboard, select New → Notebooks → Python 3.
Click on the new notebook file. It will open the Jupyter Notebook App.
In the notebook cell, type:
print("Hello World")
Run cell, press: CTRL+ENTER
Run cell and insert next, press: ALT+ENTER
Run cell and select next, press: SHIFT+ENTER
Remember to save your notebook when done (although it will automatically do the save for your every 3 mins): CTRL+S

5. Export Jupyter notebook and run it as a Python script

There are two ways to run the Python code in Jupyter notebook as a program:
Select File → Download as → Python(.py)
On the command line, type:
jupyter nbconvert --to script <NOTEBOOK_NAME>.ipynb
Run the script:
python <NOTEBOOK_NAME>.py

II. Create a Github account

1. Install Github Desktop and create a GitHub account

Download and follow the instructions here (1) to install Github Desktop (if it's not already installed). Once you've done that, create a GitHub account here (2). (Accounts are free for public repositories, but there's a charge for private repositories.)

Note:

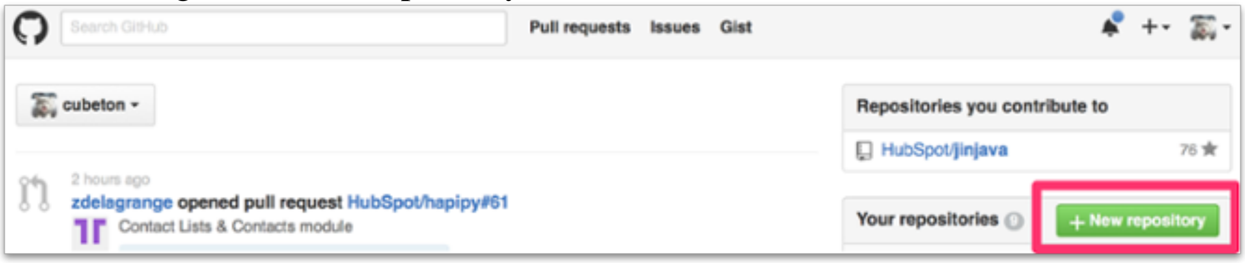
(1) https://desktop.github.com/

(2) https://github.com/join

2. Create a new repository on GitHub

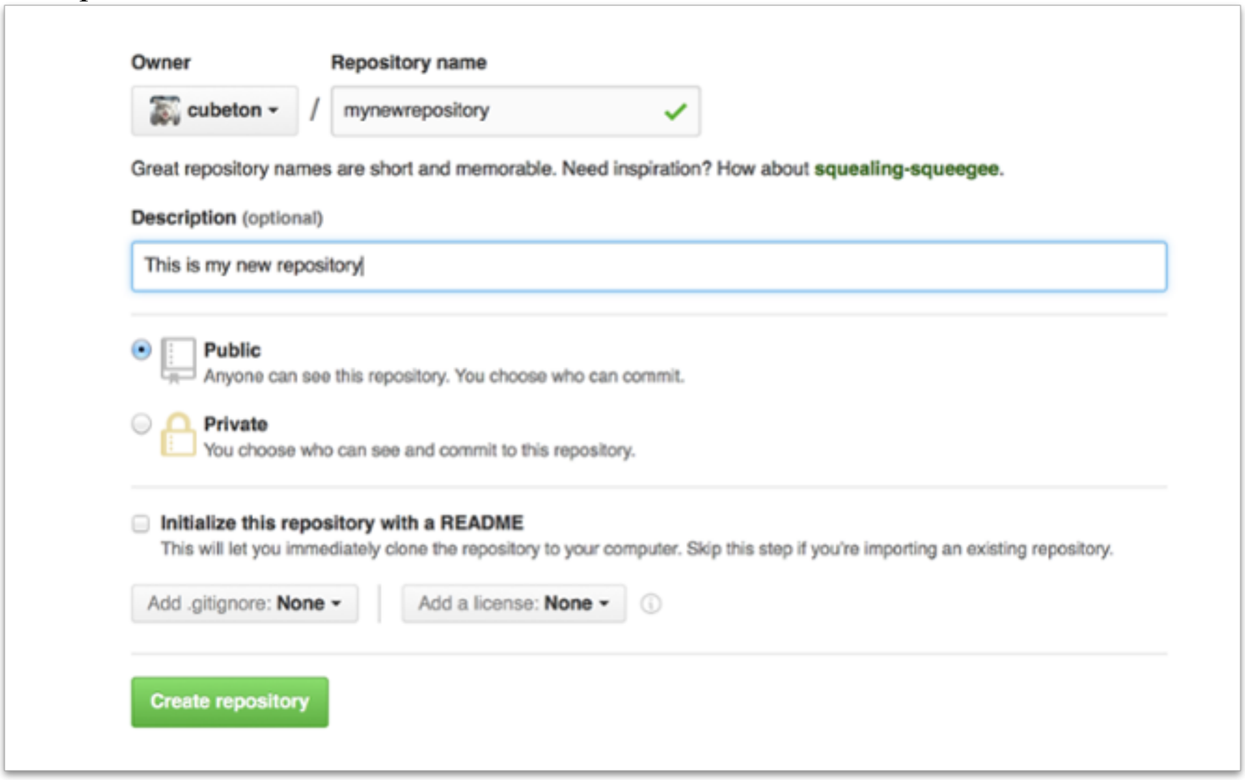
A repository is where you project will be host on Github, so you can keep a history of changes (versions) of your project, and can share your code it with everyone, and allow team work on the same project. Think of a repository as if it was a folder on your Dropbox with your project, but with more functionalities for version control and team work.

To create a new repo on GitHub, log in and go to the GitHub home page. You should see a green '+ **New repository**' button:



2 hours ago
zdelagrange opened pull request HubSpot/hapipy#61
Contact Lists & Contacts module

After clicking the button, GitHub will ask you to name your repo and provide a brief description:



Owner: cubeton
Repository name: mynewrepository

Great repository names are short and memorable. Need inspiration? How about squealing-squeegee.

Description (optional): This is my new repository

● Public
Anyone can see this repository. You choose who can commit.

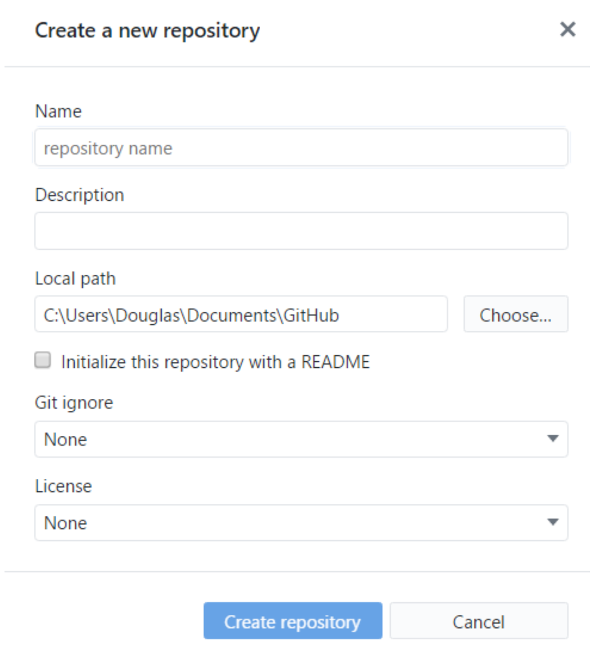
○ Private
You choose who can see and commit to this repository.

☐ Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None
Add a license: None

When you're done filling out the information, press the 'Create repository' button to make your new repo.

Note: you can also create a Repository using **Github Desktop** directly, once you login on to Github desktop using your account information:


Create a new repository

Name: repository name

Description:

Local path: C:\Users\Douglas\Documents\GitHub

☐ Initialize this repository with a README

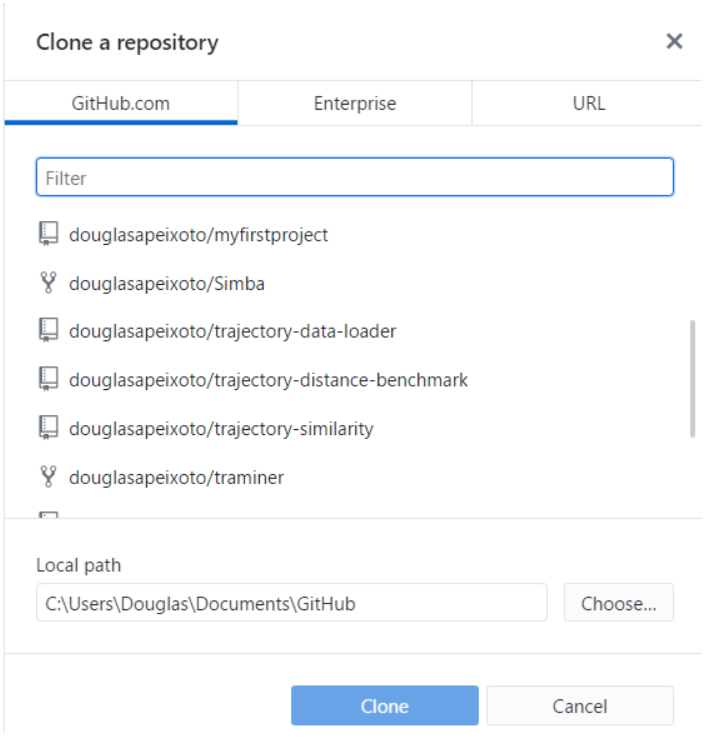
Git ignore: None

License: None

3. Cloning Your Repository

Now you can **Clone** your Github repository to your local machine, this will link your Github repository with your local repository, so that every change you make you project can be pushed to your online repo.

Just choose a directory in your local machine where you want to put your project files, and use the option **Clone a repository** on Github Desktop to link and synchronize your github repo with your local repo.


Clone a repository

douglasapeixoto/myfirstproject
douglasapeixoto/Simba
douglasapeixoto/trajectory-data-loader
douglasapeixoto/trajectory-distance-benchmark
douglasapeixoto/trajectory-similarity
douglasapeixoto/traminer

Local path: C:\Users\Douglas\Documents\GitHub

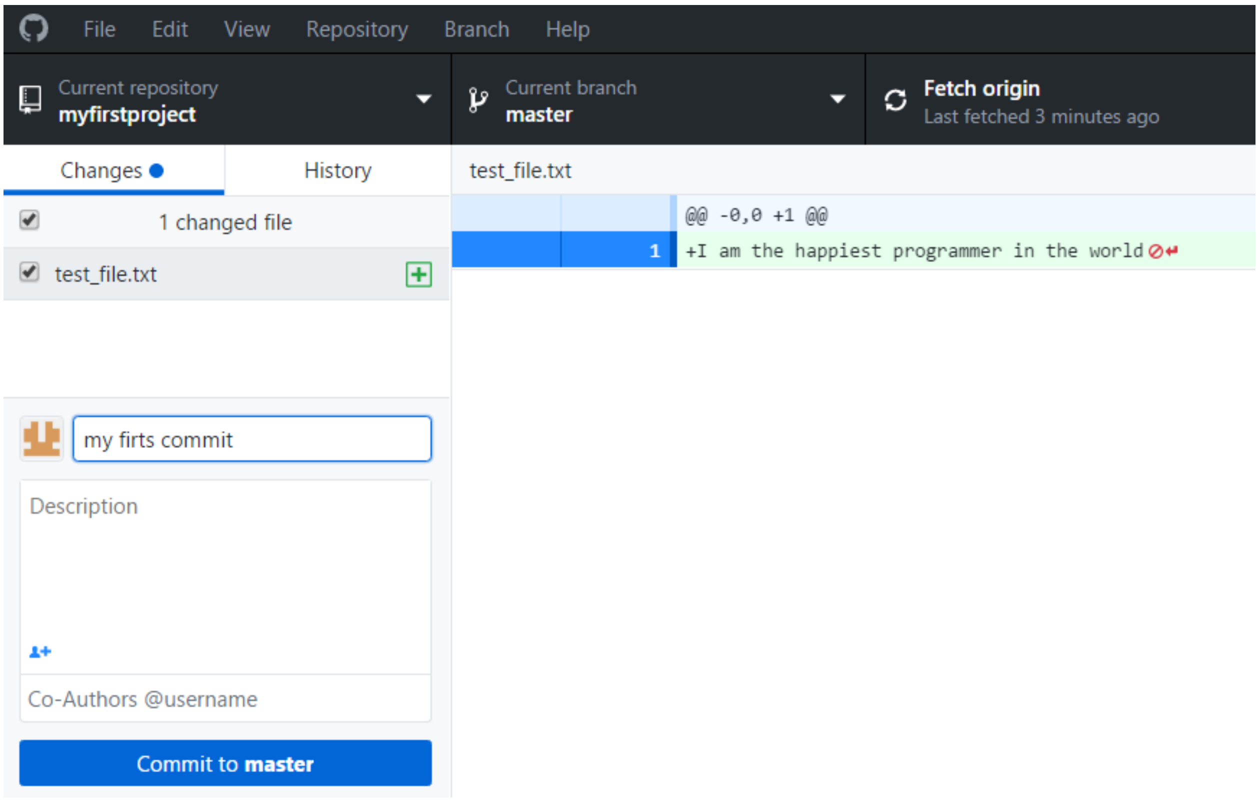
4. Add a new file to your repo

Go ahead and add a new file to the project, using any text editor you like or running a touch command.

Once you've added or modified files in a folder containing a git repo, git will notice that changes have been made inside the repo. But, git won't officially keep track of the file (that is, put it in a commit - we'll talk more about commits next) unless you explicitly tell it to.

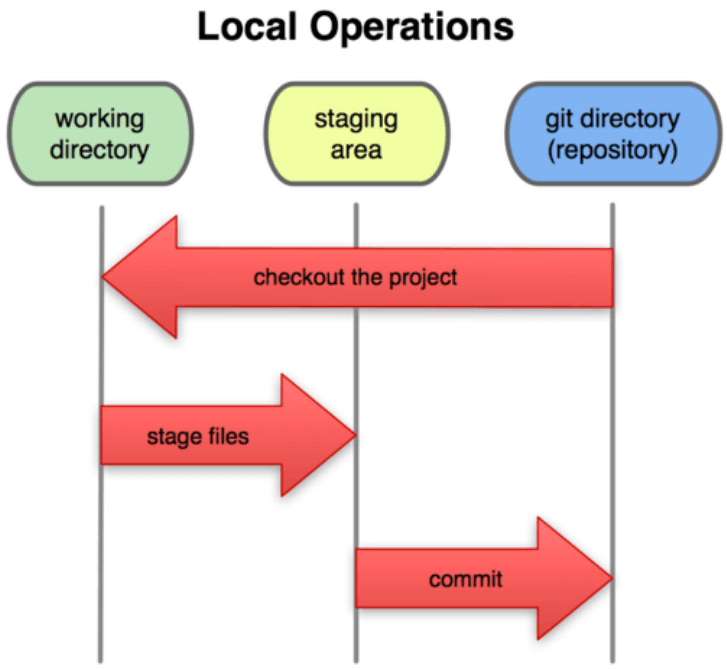
touch test_file.txt

After creating the new file, you can check on GitHub Destop that your new file has been identified.



```
@@ -0,0 +1 @@
1 +I am the happiest programmer in the world
```

5. The staging environment, the commit, and you



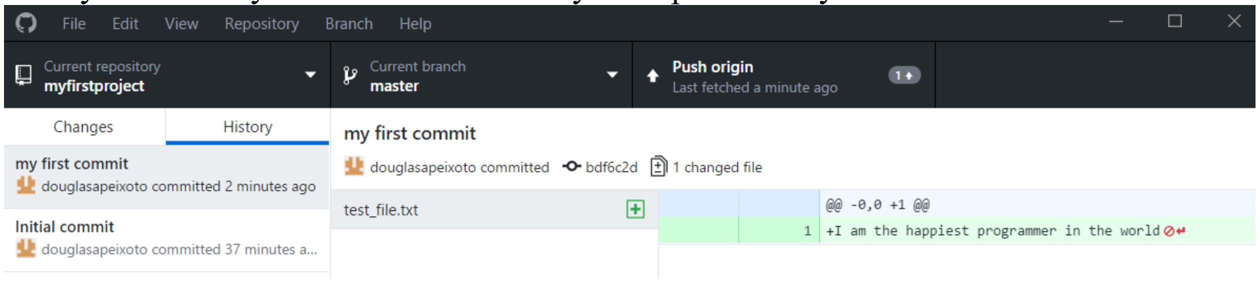
One of the most confusing parts when you're first learning git is the concept of the staging environment and how it relates to a commit.

A commit is a record of what files you have changed since the last time you made a commit. Essentially, you make changes to your repo (for example, adding a file or modifying one) and then tell git to put those files into a commit.

Commits make up the essence of your project and allow you to go back to the state of a project at any point.

So, how do you tell git which files to put into a commit? This is where the **staging environment** or **index** come in. As seen in Step 3, when you make changes to your repo, like adding a file, git notices that a file has changed but won't do anything with it (like adding it in a commit). To add a file to a commit using GitHub Desktop, just add a Summary message to your commit and click in **Commit to Master.**

Now you can see you committed files in your repo's history



my first commit
douglasapeixoto committed 2 minutes ago

Initial commit
douglasapeixoto committed 37 minutes a...

my first commit
douglasapeixoto committed bdf6c2d 1 changed file

test_file.txt

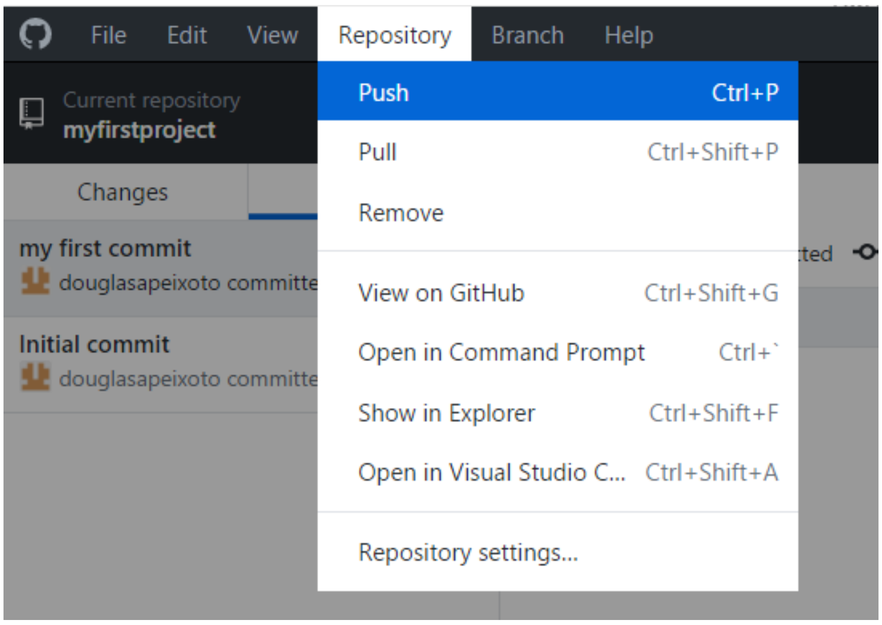
```
@@ -0,0 +1 @@
+I am the happiest programmer in the world
```

Note: The staging environment, also called 'staging', is the new preferred term for this, but you can also see it referred to as the 'index'.

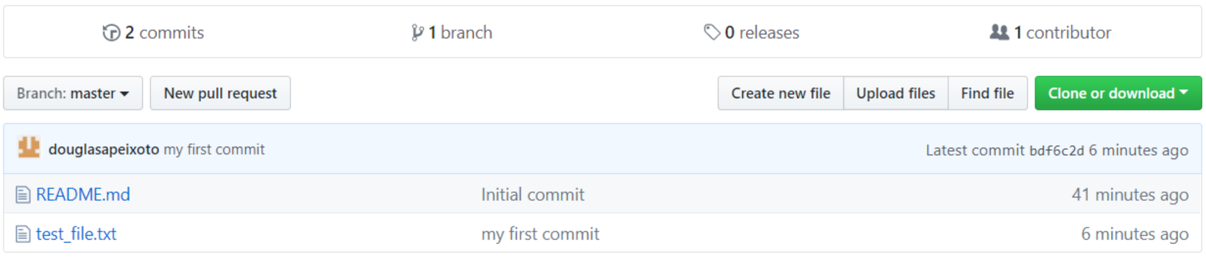
The message at the end of the commit should be something related to what the commit contains - maybe it's a new feature, maybe it's a bug fix, maybe it's just fixing a typo. Don't put a message like "asdfadsf" or "foobar". That makes the other people who see your commit sad. Very, very, sad.

6. Push a branch to GitHub

Now we'll **push** the commit in your branch to your new GitHub repo. This allows other people to see the changes you've made, this will update all changes you made into your Github repository. To push changes onto a new branch on GitHub Desktop, just go to **Repository -> Push**.



And that's all folks, now you can see your changes on your Github repository.



douglasapeixoto my first commit — Latest commit bdf6c2d 6 minutes ago

README.md	Initial commit	41 minutes ago
test_file.txt	my first commit	6 minutes ago

III. Python basics

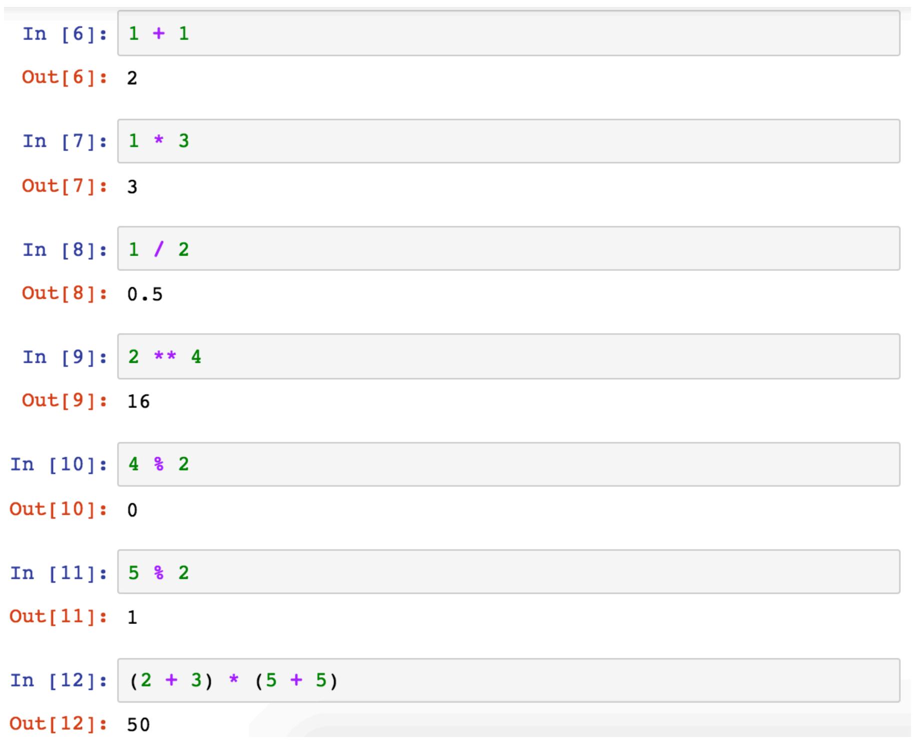
Please note, this is not meant to be a comprehensive overview of Python or programming in general, if you have no programming experience, you should probably take other courses.

This notebook will just go through the basic topics in order:

- Data types
  - Numbers
  - Strings
  - Printing
  - Lists
  - Dictionaries
  - Booleans
  - Tuples
  - Sets
- Comparison Operators
- if,elif, else Statements
- for Loops
- while Loops
- range()
- list comprehension
- functions
- lambda expressions
- map and filter
- methods

1. Data types

1.1. Numbers



```
In [6]: 1 + 1
Out[6]: 2

In [7]: 1 * 3
Out[7]: 3

In [8]: 1 / 2
Out[8]: 0.5

In [9]: 2 ** 4
Out[9]: 16

In [10]: 4 % 2
Out[10]: 0

In [11]: 5 % 2
Out[11]: 1

In [12]: (2 + 3) * (5 + 5)
Out[12]: 50
```

1.2. Variable assignment

```
In [13]: # Can not start with number or special characters
         name_of_var = 2
```

```
In [14]: x = 2
         y = 3
```

```
In [15]: z = x + y
```

```
In [16]: z
Out[16]: 5
```

1.3. Strings

```
In [17]: 'single quotes'
Out[17]: 'single quotes'
```

```
In [18]: "double quotes"
Out[18]: 'double quotes'
```

```
In [19]: " wrap lot's of other quotes"
Out[19]: " wrap lot's of other quotes"
```

1.4. Printing

```
In [20]: x = 'hello'
```

```
In [21]: x
Out[21]: 'hello'
```

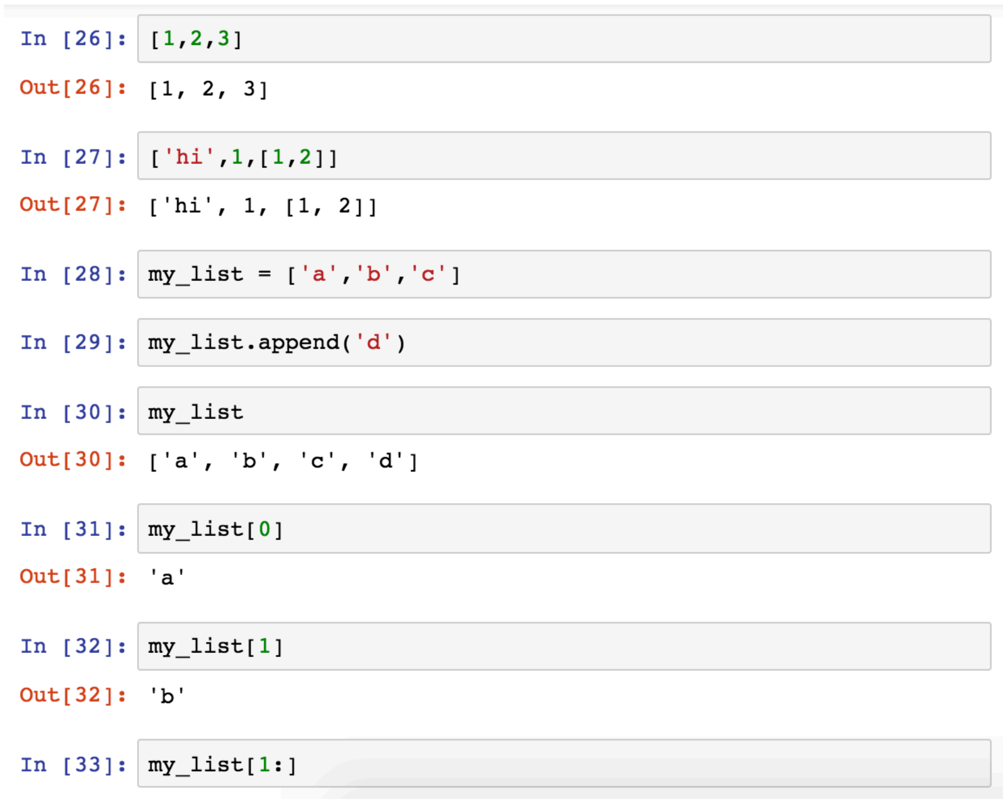
```
In [22]: print(x)
         hello
```

```
In [23]: num = 12
         name = 'Sam'
```

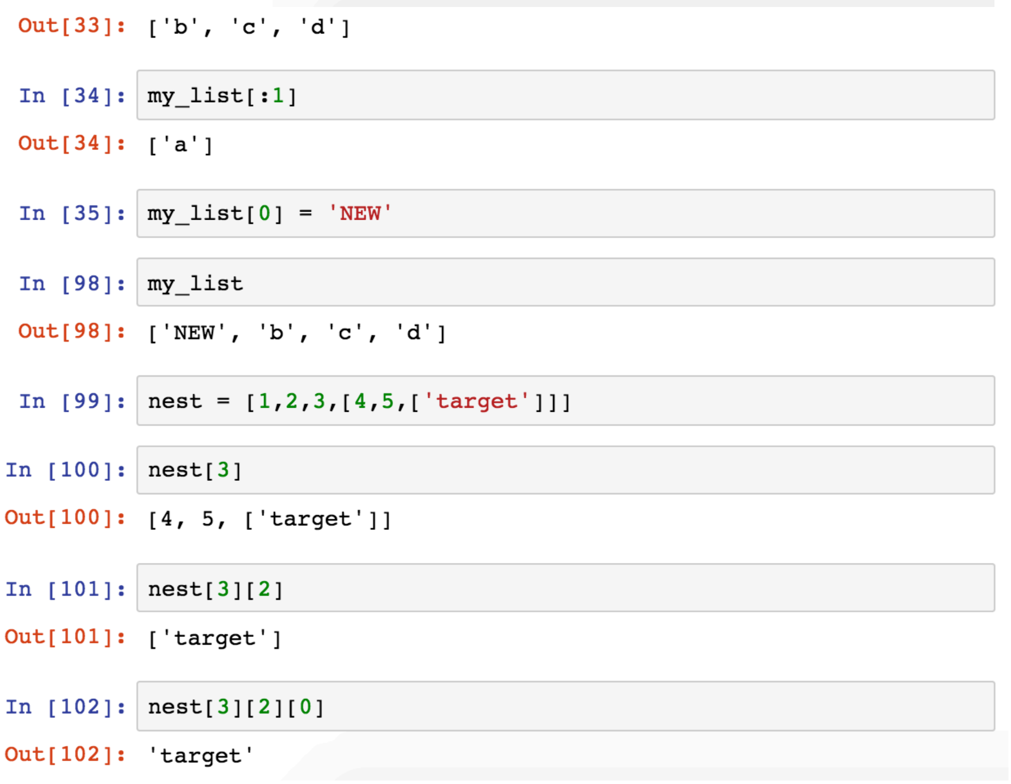
```
In [24]: print('My number is: {one}, and my name is: {two}'.format(one=
         My number is: 12, and my name is: Sam
```

```
In [25]: print('My number is: {}, and my name is: {}'.format(num,name))
         My number is: 12, and my name is: Sam
```

1.5. Lists

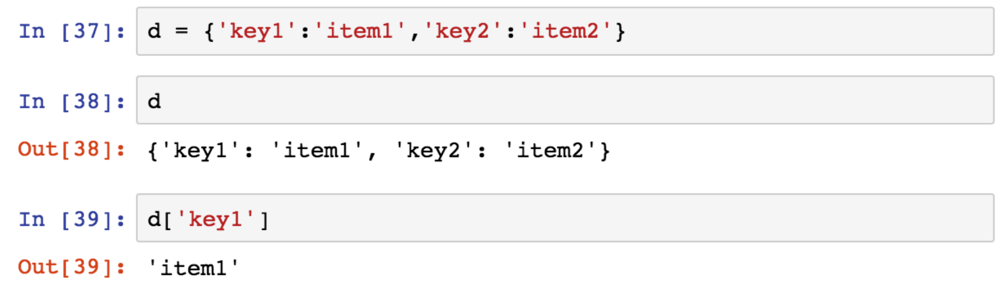


```
In [26]: [1,2,3]
Out[26]: [1, 2, 3]

In [27]: ['hi',1,[1,2]]
Out[27]: ['hi', 1, [1, 2]]

In [28]: my_list = ['a','b','c']

In [29]: my_list.append('d')

In [30]: my_list
Out[30]: ['a', 'b', 'c', 'd']

In [31]: my_list[0]
Out[31]: 'a'

In [32]: my_list[1]
Out[32]: 'b'

In [33]: my_list[1:]
```



```
Out[33]: ['b', 'c', 'd']

In [34]: my_list[:1]
Out[34]: ['a']

In [35]: my_list[0] = 'NEW'

In [98]: my_list
Out[98]: ['NEW', 'b', 'c', 'd']

In [99]: nest = [1,2,3,[4,5,['target']]]

In [100]: nest[3]
Out[100]: [4, 5, ['target']]

In [101]: nest[3][2]
Out[101]: ['target']

In [102]: nest[3][2][0]
Out[102]: 'target'
```

1.6. Dictionaries



```
In [37]: d = {'key1':'item1','key2':'item2'}

In [38]: d
Out[38]: {'key1': 'item1', 'key2': 'item2'}

In [39]: d['key1']
Out[39]: 'item1'
```

1.7. Booleans

```
In [40]: True
Out[40]: True

In [41]: False
Out[41]: False
```

1.8. Tuples

```
In [42]: t = (1,2,3)

In [43]: t[0]
Out[43]: 1

In [44]: t[0] = 'NEW'

---------------------------------------------------------------------------
TypeError                                 Traceback (most r
ecent call last)
<ipython-input-44-97e4e33b36c2> in <module>()
----> 1 t[0] = 'NEW'

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

1.9. Sets

```
In [45]: {1,2,3}
Out[45]: {1, 2, 3}

In [46]: {1,2,3,1,2,1,2,3,3,3,3,2,2,2,1,1,2}
Out[46]: {1, 2, 3}
```

2. Comparison Operators

```
In [47]: 1 > 2
Out[47]: False

In [48]: 1 < 2
Out[48]: True

In [49]: 1 >= 1
Out[49]: True

In [50]: 1 <= 4
Out[50]: True

In [51]: 1 == 1
Out[51]: True

In [52]: 'hi' == 'bye'
Out[52]: False
```

3. Logic Operators

```
In [53]: (1 > 2) and (2 < 3)

Out[53]: False
```

```
In [54]: (1 > 2) or (2 < 3)

Out[54]: True
```

```
In [55]: (1 == 2) or (2 == 3) or (4 == 4)

Out[55]: True
```

4. If, elseif, else Statement

```
In [56]: if 1 < 2:
             print('Yep!')

         Yep!
```

```
In [57]: if 1 < 2:
             print('yep!')

         yep!
```

```
In [58]: if 1 < 2:
             print('first')
         else:
             print('last')

         first
```

```
In [59]: if 1 > 2:
             print('first')
         else:
             print('last')

         last
```

```
In [60]: if 1 == 2:
             print('first')
         elif 3 == 3:
             print('middle')
         else:
             print('Last')

         middle
```

5. For Loops

```
In [61]: seq = [1,2,3,4,5]
```

```
In [62]: for item in seq:
             print(item)

         1
         2
         3
         4
         5
```

```
In [63]: for item in seq:
             print('Yep')

         Yep
         Yep
         Yep
         Yep
         Yep
```

```
In [64]: for jelly in seq:
             print(jelly+jelly)

         2
         4
         6
         8
         10
```

6. While Loops

```
In [65]: i = 1
         while i < 5:
             print('i is: {}'.format(i))
             i = i+1

         i is: 1
         i is: 2
         i is: 3
         i is: 4
```

7. Range

```
In [66]: range(5)

Out[66]: range(0, 5)
```

```
In [67]: for i in range(5):
             print(i)

         0
         1
         2
         3
         4
```

```
In [68]: list(range(5))

Out[68]: [0, 1, 2, 3, 4]
```

8. List Comprehension

```
In [69]: x = [1,2,3,4]
```

```
In [70]: out = []
         for item in x:
             out.append(item**2)
         print(out)

         [1, 4, 9, 16]
```

```
In [71]: [item**2 for item in x]

Out[71]: [1, 4, 9, 16]
```

9. Functions

```
In [72]: def my_func(param1='default'):
             """
             Docstring goes here.
             """
             print(param1)
```

```
In [73]: my_func

Out[73]: <function __main__.my_func>
```

```
In [74]: my_func()

         default
```

```
In [75]: my_func('new param')

         new param
```

```
In [76]: my_func(param1='new param')

         new param
```

```
In [77]: def square(x):
             return x**2
```

```
In [78]: out = square(2)
```

```
In [79]: print(out)

         4
```

10.Lambda expression

```
In [80]: def times2(var):
             return var*2

In [81]: times2(2)

Out[81]: 4

In [82]: lambda var: var*2

Out[82]: <function __main__.<lambda>>
```

11.Map and filter

```
In [83]: seq = [1,2,3,4,5]

In [84]: map(times2,seq)

Out[84]: <map at 0x105316748>

In [85]: list(map(times2,seq))

Out[85]: [2, 4, 6, 8, 10]

In [86]: list(map(lambda var: var*2,seq))

Out[86]: [2, 4, 6, 8, 10]

In [87]: filter(lambda item: item%2 == 0,seq)

Out[87]: <filter at 0x105316ac8>

In [88]: list(filter(lambda item: item%2 == 0,seq))

Out[88]: [2, 4]
```

12.Methods

```
In [111]: st = 'hello my name is Sam'

In [112]: st.lower()

Out[112]: 'hello my name is sam'

In [113]: st.upper()

Out[113]: 'HELLO MY NAME IS SAM'

In [103]: st.split()

Out[103]: ['hello', 'my', 'name', 'is', 'Sam']

In [104]: tweet = 'Go Sports! #Sports'

In [106]: tweet.split('#')

Out[106]: ['Go Sports! ', 'Sports']

In [107]: tweet.split('#')[1]

Out[107]: 'Sports'

 In [92]: d

Out[92]: {'key1': 'item1', 'key2': 'item2'}

 In [93]: d.keys()

Out[93]: dict_keys(['key2', 'key1'])
```

```
In [94]: d.items()
Out[94]: dict_items([('key2', 'item2'), ('key1', 'item1')])

In [95]: lst = [1,2,3]

In [96]: lst.pop()
Out[96]: 3

In [108]: lst
Out[108]: [1, 2]

In [109]: 'x' in [1,2,3]
Out[109]: False

In [110]: 'x' in ['x','y','z']
Out[110]: True
```

IV. Python basics – Exercises

Answer the questions or complete the tasks outlined in bold below, use the specific method described if applicable.

What is 7 to the power of 4?

```
In [1]: 
Out[1]: 2401
```

Split this string:

```
s = "Hi there Sam!"
```

into a list.

```
In [4]: 

In [3]: 
Out[3]: ['Hi', 'there', 'dad!']
```

Given the variables:

```
planet = "Earth"
diameter = 12742
```

Use .format() to print the following string:

```
The diameter of Earth is 12742 kilometers.
```

```
In [5]: planet = "Earth"
        diameter = 12742

In [6]: 
        The diameter of Earth is 12742 kilometers.
```

Given this nested list, use indexing to grab the word "hello"

```
In [7]: lst = [1,2,[3,4],[5,[100,200,['hello']],23,11],1,7]

In [14]: 
Out[14]: 'hello'
```

Given this nested dictionary grab the word "hello". Be prepared, this will be annoying/tricky

```
In [16]: d = {'k1':[1,2,3,{'tricky':['oh','man','inception',{'target':[1,2,3,'hello'

In [22]: 
Out[22]: 'hello'
```

What is the main difference between a tuple and a list?

```
In [23]: # Tuple is immutable
```

Create a function that grabs the email website domain from a string in the form:

```
user@domain.com
```

So for example, passing "user@domain.com" would return: domain.com

```
In [24]:
```

```
In [26]: domainGet('user@domain.com')

Out[26]: 'domain.com'
```

Create a basic function that returns True if the word 'dog' is contained in the input string. Don't worry about edge cases like a punctuation being attached to the word dog, but do account for capitalization.

```
In [27]:
```

```
In [28]: findDog('Is there a dog here?')

Out[28]: True
```

Create a function that counts the number of times the word "dog" occurs in a string. Again ignore edge cases.

```
In [30]:
```

```
In [31]: countDog('This dog runs faster than the other dog dude!')

Out[31]: 2
```

Use lambda expressions and the filter() function to filter out words from a list that don't start with the letter 's'. For example:

```
seq = ['soup','dog','salad','cat','great']
```

should be filtered down to:

```
['soup','salad']
```

```
In [34]: seq = ['soup','dog','salad','cat','great']
```

```
In [35]:

Out[35]: ['soup', 'salad']
```

Final Problem

You are driving a little too fast, and a police officer stops you. Write a function to return one of 3 possible results: "No ticket", "Small ticket", or "Big Ticket". If your speed is 60 or less, the result is "No Ticket". If speed is between 61 and 80 inclusive, the result is "Small Ticket". If speed is 81 or more, the result is "Big Ticket". Unless it is your birthday (encoded as a boolean value in the parameters of the function) -- on your birthday, your speed can be 5 higher in all cases.

```
In [36]: def caught_speeding(speed, is_birthday):
             pass
```

```
In [42]: caught_speeding(81,True)

Out[42]: 'Small Ticket'
```

```
In [43]: caught_speeding(81,False)

Out[43]: 'Big Ticket'
```

Great Job!