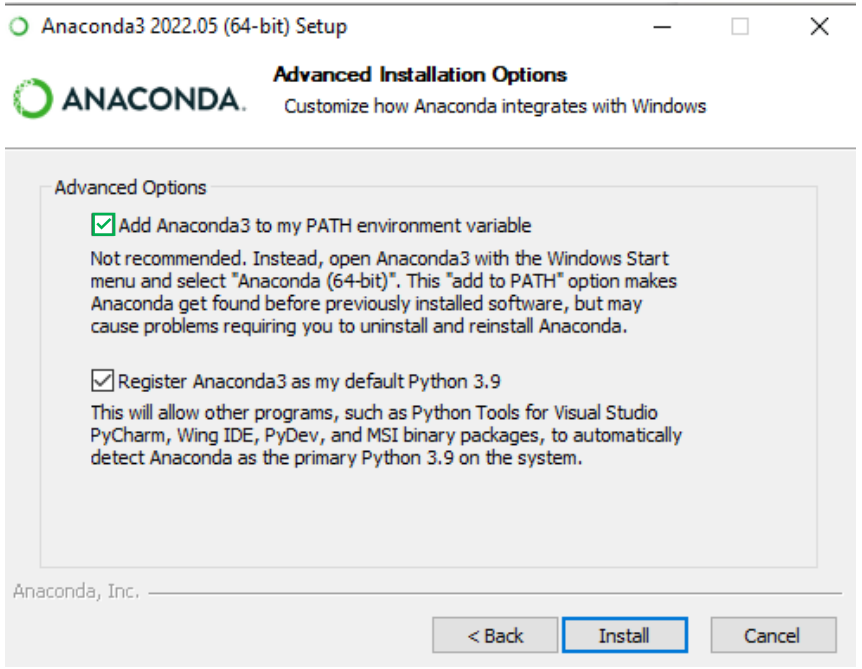
Essentials!

Install Anaconda

- It is important to do it by yourselves to become familiar with all the issues you may face!

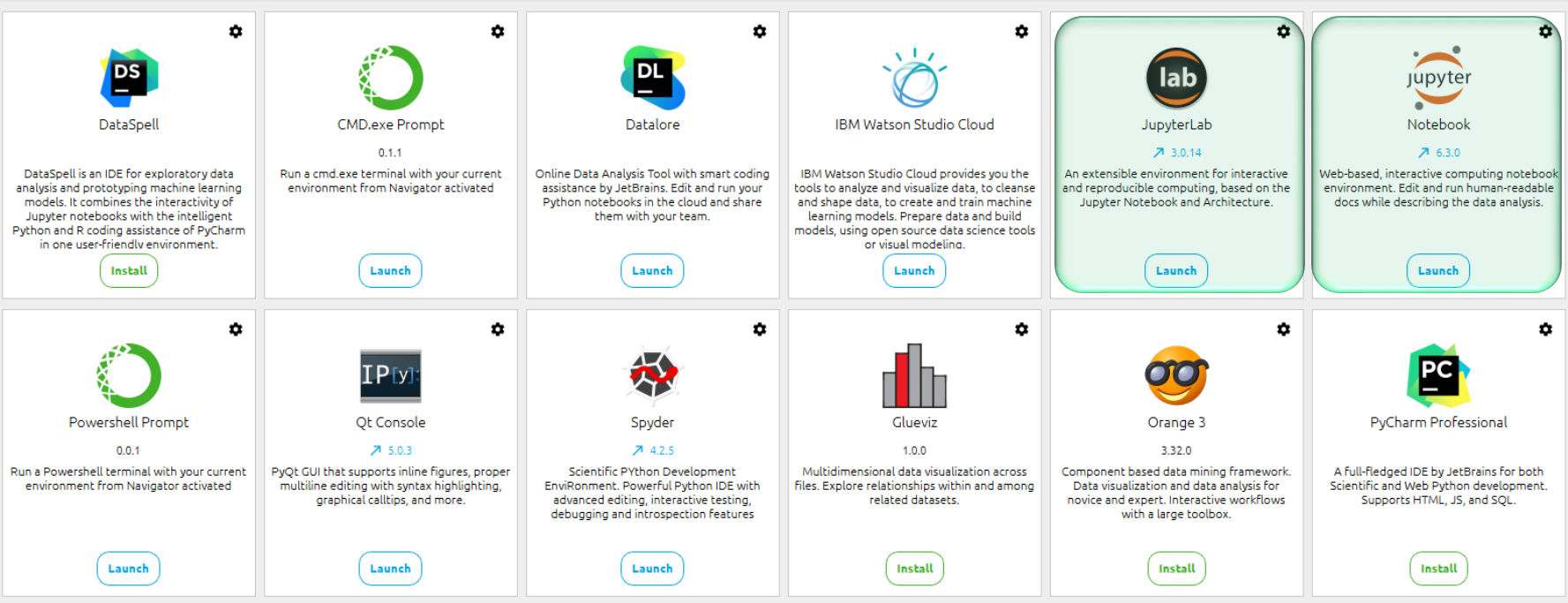
Install Anaconda – All the Steps

- Go to https://www.anaconda.com/
- You may want to choose from additional installer
- Run the file
- I would recommend to add to my path!

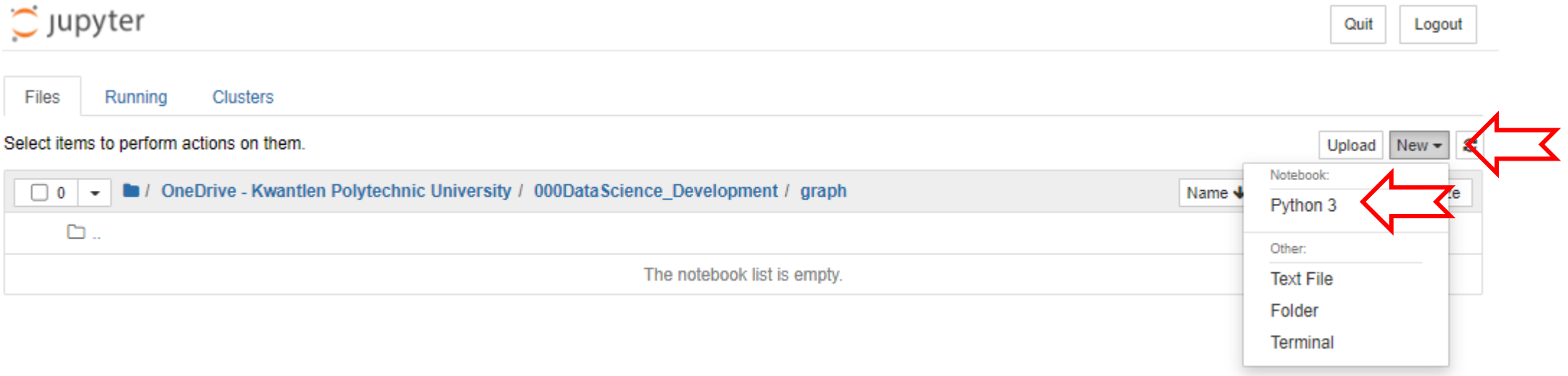


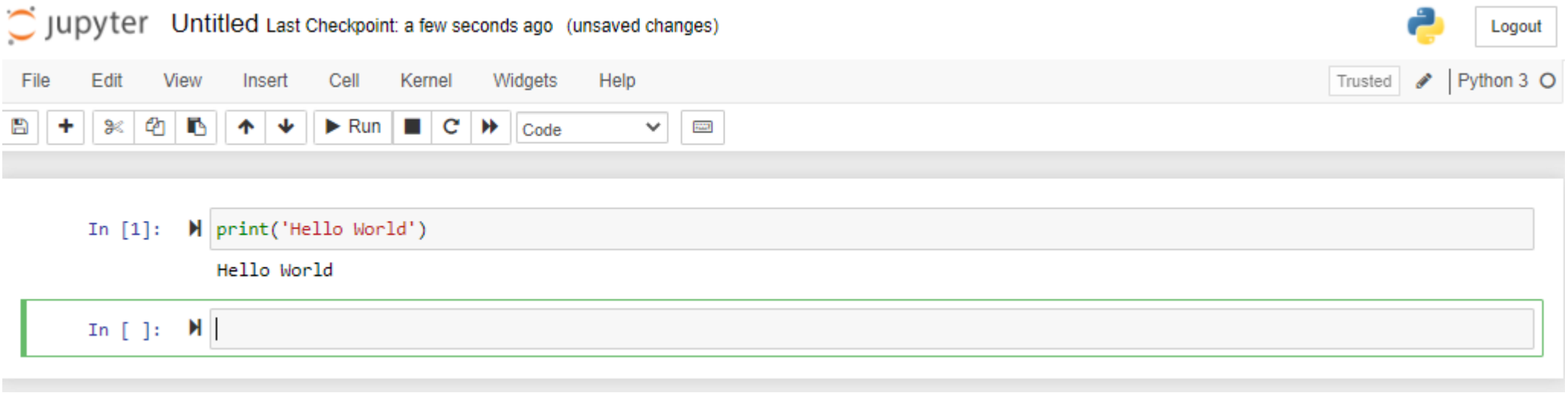
Jupyter – We believe in!

- Jupyter Lab or Jupyter Notebook! Whatever you feel more comfortable. I generally use Jupter Notebook



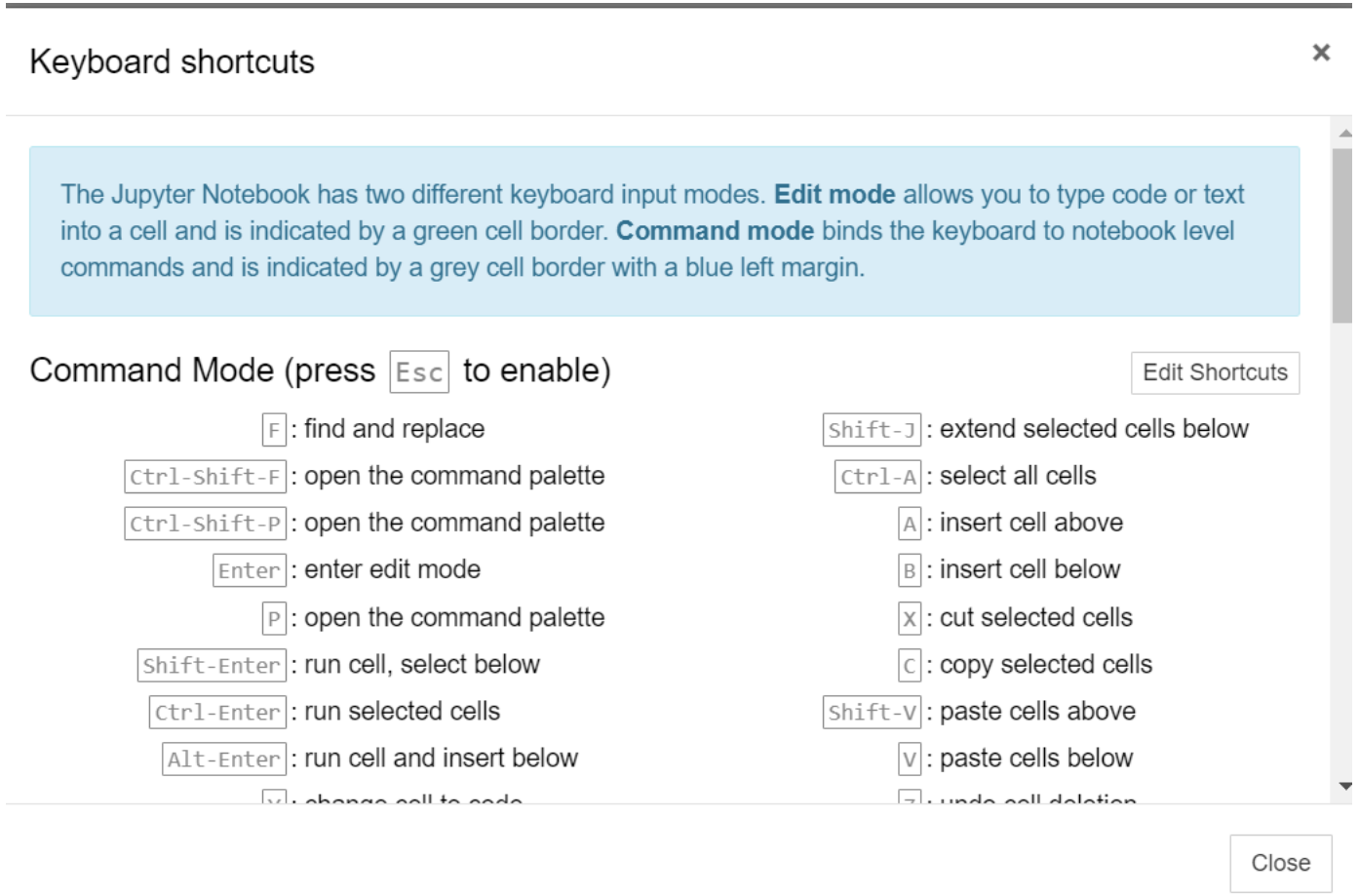
Jupyter


jupyter
Quit
Logout
Files
Running
Clusters
Select items to perform actions on them.
Upload
New
0
/ OneDrive - Kwantlen Polytechnic University / 000DataScience_Development / graph
Name
..
The notebook list is empty.
Notebook:
Python 3
Other:
Text File
Folder
Terminal

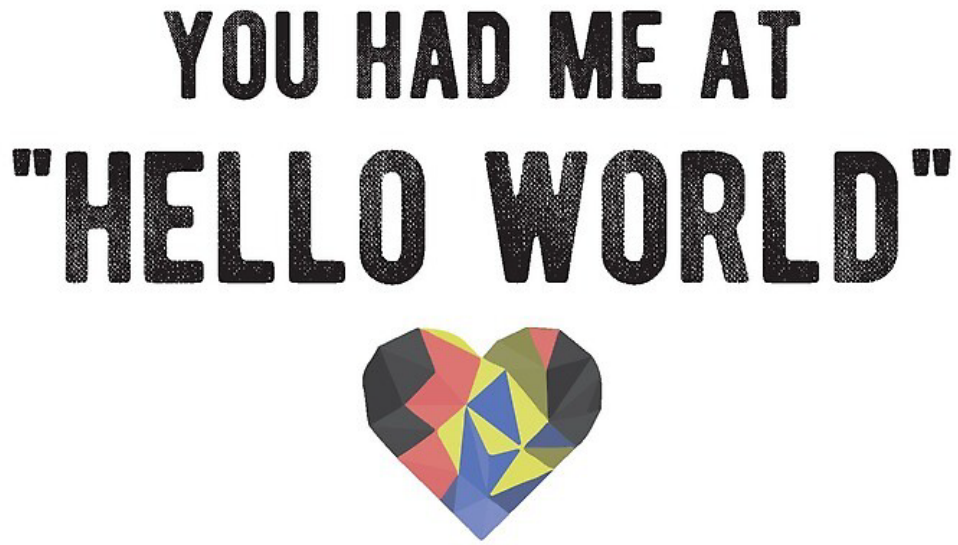

jupyter Untitled Last Checkpoint: a few seconds ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [1]:
print('Hello World')
Hello World
In [ ]:

Keyboard shortcuts

- Esc + H

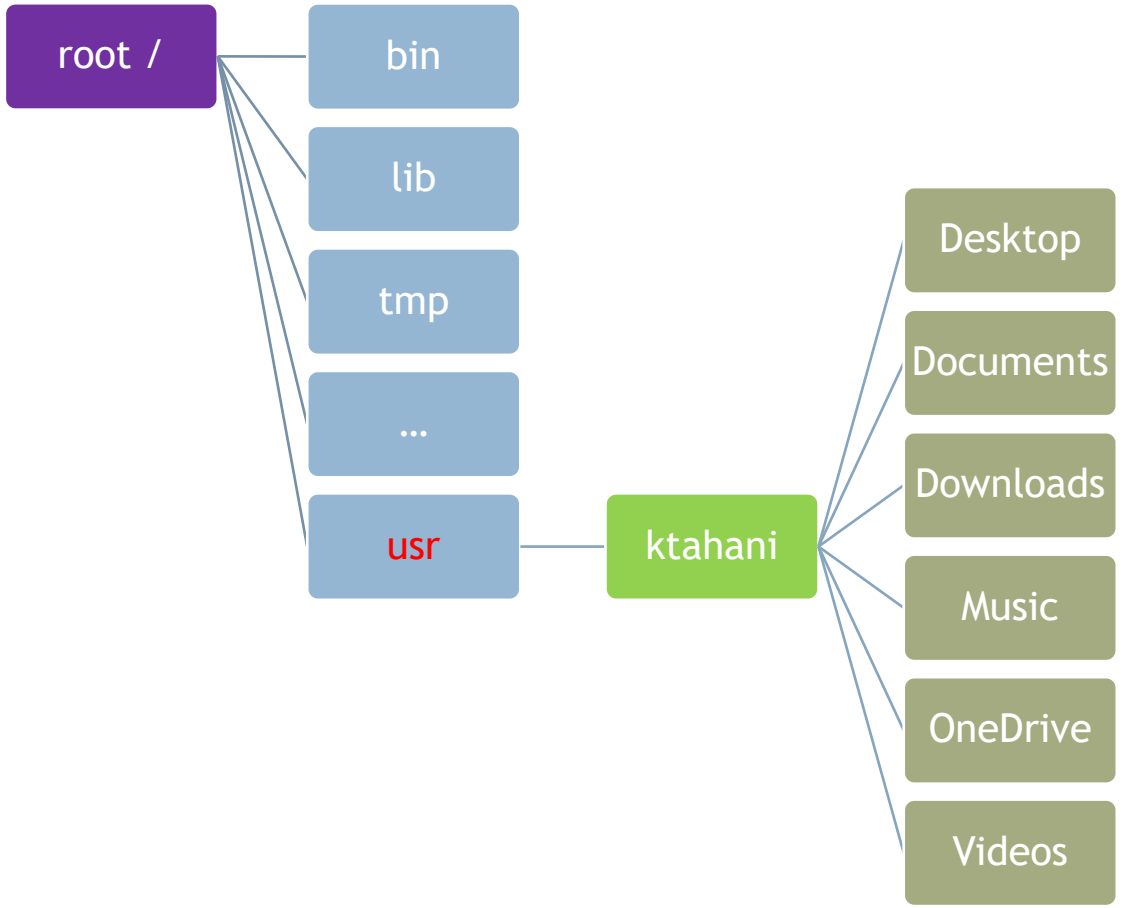


Let's Begin!

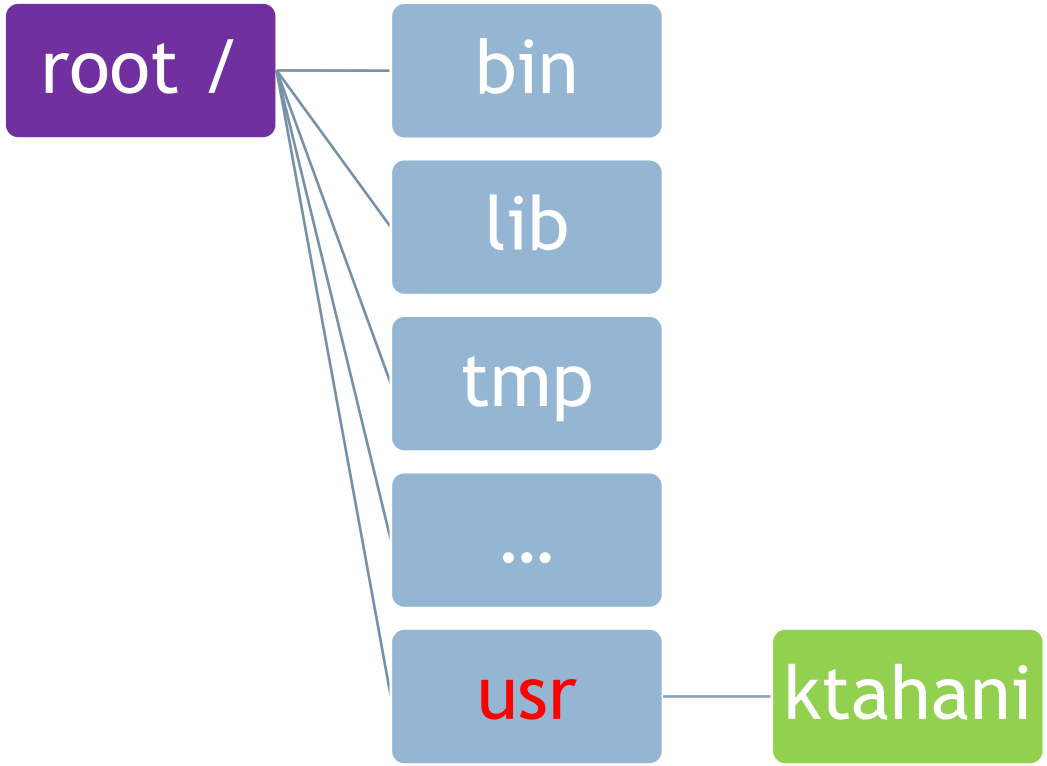


OS File Structures

- Operating Systems organize their folders in a hierarchy (a tree) with parents and children, all relative to a base **root** directory.
- Files and directories have absolute paths based on the root, where each additional level down adds a " /".
- The absolute path for Downloads is:
  - /usr/ktahani/Downloads



Root vs Home directory



- The **green** directory is a special directory called "home", which is also known as "**~**".
- This is the default directory upon opening your terminal.

pwd and cd

your current location via PWD

```
In [2]: pwd
Out[2]: 'C:\\Users\\ktahani-la\\OneDrive - Kwantlen Polytechnic University\\000DataScience_Development\\graph'
```

Use 'cd' to navigate

```
In [3]: cd ..
C:\Users\ktahani-la\OneDrive - Kwantlen Polytechnic University\000DataScience_Development
```

Goes one folder backward

```
In [6]: cd
C:\Users\ktahani-la
```

Goes to the home folder

```
In [7]: cd/
C:\
```

Goes to the root folder – NOTE: slash vs backslash in windows!

```
In [9]: cd /Users/ktahani-la/OneDrive - Kwantlen Polytechnic University/000DataScience_Development
C:\Users\ktahani-la\OneDrive - Kwantlen Polytechnic University\000DataScience_Development
```

Goes to the given folder

```
In [10]: cd ~/OneDrive - Kwantlen Polytechnic University
C:\Users\ktahani-la\OneDrive - Kwantlen Polytechnic University
```

Means 'Home' directory

ls and mkdir

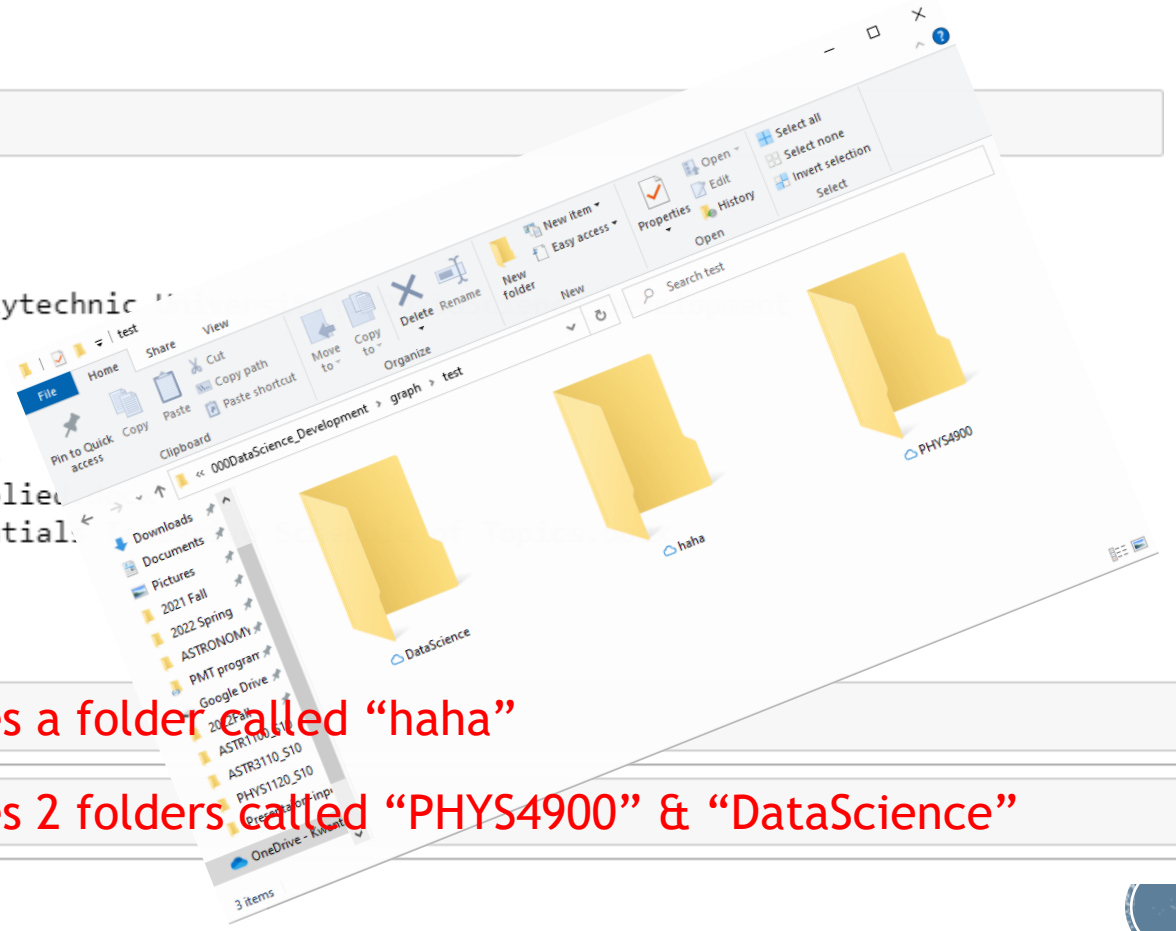
Check what is inside a folder

```
ls
```

```
 Volume in drive C is Windows
 Volume Serial Number is 540C-472D

 Directory of C:\Users\ktahani-la\OneDrive - Kwantlen Polytechnic

08/23/2022  12:58 PM    <DIR>          .
08/23/2022  12:58 PM    <DIR>          ..
08/16/2022  11:49 AM    <DIR>          .ipynb_checkpoints
02/25/2021  12:38 PM            (71,429) Comments To Be Applie
04/01/2021  08:34 PM            (24,829) Data Science Essential
```



Make Directory

```
mkdir haha
```

→ Creates a folder called “haha”

```
mkdir PHYS4900 DataScience
```

→ Creates 2 folders called “PHYS4900” & “DataScience”

Start Python – Code Styling

- Note when you write a long a code it should follow pep8 formatting
- https://peps.python.org/pep-0008/
- You may want to use autopep8 to take care of this for you. Then you would be having a nicely shaped code!
- https://pypi.org/project/autopep8/
- In this course, we don't care. Frankly, I don't know it either!

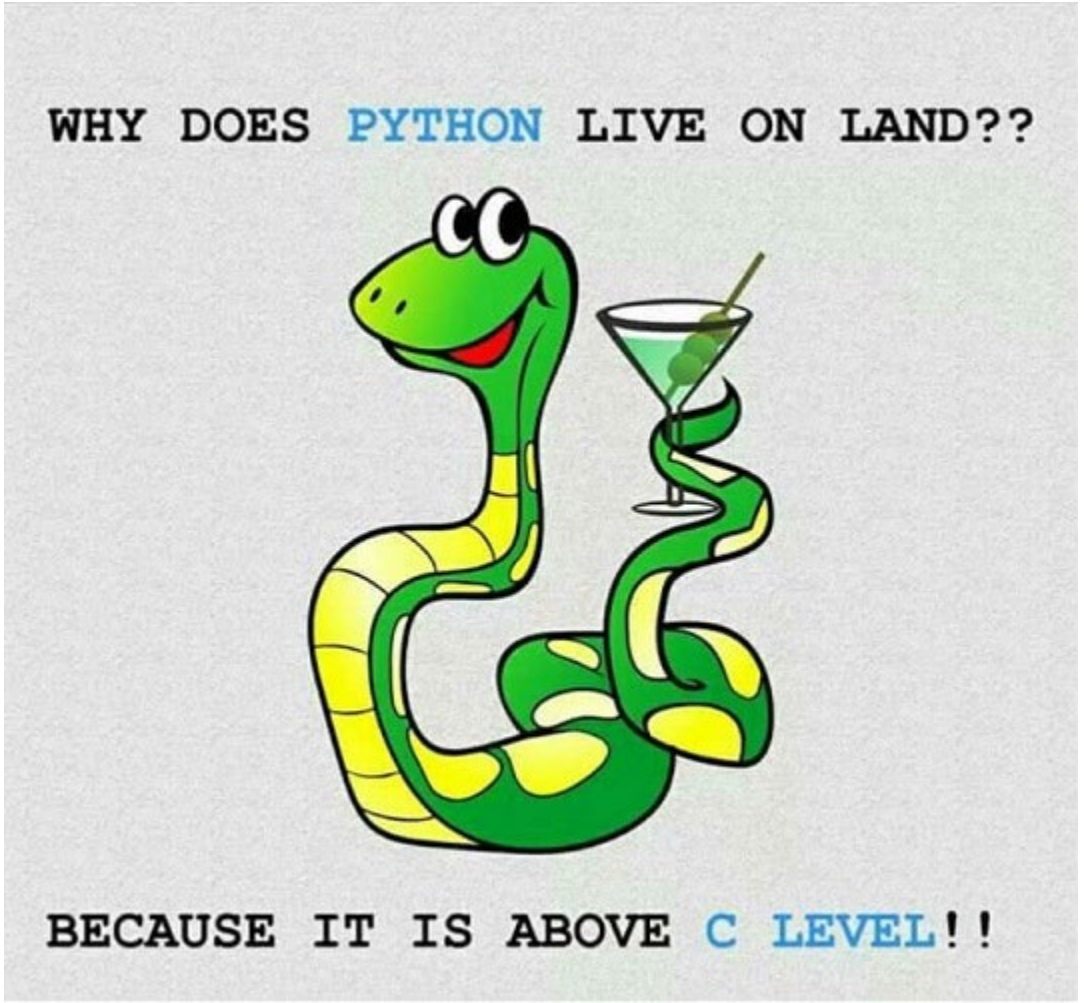
While we are at it ….

- I am a terrible computer coder! So, if you stuck and you needed to fix your code … Google it
- Seriously, this is the most important skill you would develop. Perform the research to solve the issues you are facing!

- We won't be hired as Software Engineers! But, in fact, as Data Scientist!
  - What is the difference?

Python documentation

- Python.org



Basics – Numbers

- ……, -3, -2, -1, 0, 1, 2, 3, …… → These are integers

```
type(-3)
```

```
int
```

- 1. or 1.0, 1.5, 2.2345 → floating point numbers (a positive or negative whole number with a decimal point) → We call them float

```
type(-3.23)
```

```
float
```

```
type(1.)
```

```
float
```

- What is type(1+1.)

a) int

b) float

Basics – Math Operators

	Operator	Name	Example	Try it
`1+1` → `2`	+	Addition	x + y	Try it »
`1-1` → `0`	-	Subtraction	x - y	Try it »
`2*3` → `6`	*	Multiplication	x * y	Try it »
`3/2` → `1.5`	/	Division	x / y	Try it »
`21%2` → `1`	%	Modulus	x % y	Try it »
`2**3` → `8`	**	Exponentiation	x ** y	Try it »
`3//2` → `1`	//	Floor division	x // y	Try it »

Variables – and +=, -=, *=

- x = 10
- num_of_stars = 1025
- num_of_stars = num_of_stars+1
  - Or num_of_stars += 1

```
num_of_stars = 1025
num_of_stars +=25
print(num_of_stars)
```

```
1050
```

Adding

```
num_of_stars = 1025
num_of_stars -=25
print(num_of_stars)
```

```
1000
```

Subtracting

```
num_of_stars = 1025
num_of_stars *=10
print(num_of_stars)
```

```
10250
```

Multiplying

Variables

this_is_snake_case

- Programmers prefer a standard style for naming variables. Thus, most variables that we use should be snake_case (underline in between)
- Naming restrictions:
  - Variables must start with letter or underscore:
    - stars, or _stars → Yup! ✅
    - 2stars → NO!! ✗ ✗
  - The rest of the name may contain, letters, underscore, numbers:
    - cats2 → Yup! ✅
    - cats@home → NO!! ✗ ✗
  - Names are case sensitive: cats != CATS, cats!=Cats (!= means NOT EQUAL)
- CAPITAL_SNAKE_CASE usually used for constants (e.g. PI = 3.1415)
- Variables starting & ending with two underscores, “dunder” (i.e. **d**ouble **under**score) are private and supposed to be left alone!
  - __do_not_change_it__

Data Types

```
type(True)
```

```
bool
```

- Any assigned values must always be a valid data type!

Data Type	Examples
Binary Types	memoryview, bytearray, bytes
Boolean Type	bool
Numeric Types	int, float, complex
Sequence Types	range, tuple, list
Set Types	frozenset, set
Mapping Type	dict
Text Type	str

Boolean – True or False

- Note the capital letter!

```
'a' in 'Kianoosh'
```

```
True
```

```
'c' in 'Kianoosh'
```

```
False
```

- Python is highly flexible on reassessing variables into a different type. (aka dynamic typing)

```
awesomeness = True    # This is boolean
print(awesomness)
```

```
True
```

```
awesomeness = 20      # This is integer
print(awesomeness)
```

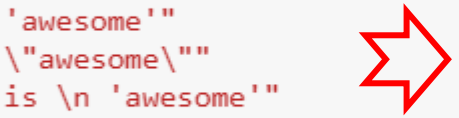
```
20
```

- This is not the case for many other languages! not_awesomeness!

String & String Escape Sequences

```
name1 = 'Astrophysics'
name2 = "Astrophysics"
name3 = 'Astrophysics is "awesome"'
name4 = "Astrophysics is 'awesome'"
name5 = "Astrophysics is \"awesome\""
name6 = "Astrophysics \n is \n 'awesome'"


back_slash = 'this is backslash: \\'
```



```
print(name1)
print(name2)
print(name3)
print(name4)
print(name5)
print(name6)


print(back_slash)
```

```
Astrophysics
Astrophysics
Astrophysics is "awesome"
Astrophysics is 'awesome'
Astrophysics is "awesome"
Astrophysics
 is
 'awesome'
this is backslash: \
```

- https://www.w3schools.com/python/gloss_python_escape_characters.asp

String – Concatenation & Access a letter!

```
str1 = 'astro'
str2 ='physics'
str3 = str1+str2
str4 = str1+' '+str2

print(str3)
print(str4)
```

```
astrophysics
astro physics
```

```
str1 = 'astro'
str1 += 'physics'
print(str1)
```

```
astrophysics
```

```
str1 = 'astrophyic'
print(str1[0])
print(str1[1])
print(str1[-1])
```

index	0	1	2	3	4	5	6	7	8	9
letter	'a'	's'	't'	'r'	'o'	'p'	'h'	'y'	'i'	'c'

Index →

```
a
s
c
```

String – Formatting

- I use F-string formatting and please use this method.
- .format method is kinda old!

```
mag_field = 109
print(f"Magnitic filed determined to be {mag_field} uT")
```

```
Magnitic filed determined to be 109 uT
```

Convert data type

```
float(8)
```

8 is an integer by itself. Here, it is converted to a float (i.e. 8.0)

```
8.0
```

```
int(12.7)
```

12.7 is a floating number. Converting will make it 12 (an integer)

```
12
```

```
str(8)
```

This converts a number (can be integer or float) to a string

```
'8'
```

Question

- What is the outcome of str(8)+ ‘ books’?

A. ‘8books’

B. ‘8 books’

C. Error!!

D. ‘booksbooksbooksbooksbooksbooksbooksbooks’

```
str(8)+' books'
```

```
'8 books'
```

- What is the outcome of str(8+6)?

A. ‘8+6’

B. ‘14’

```
str(8+6)
```

```
'14'
```

A few more things on strings!

```
'KIANOOSH'.lower()
```

Make a string lower case!

```
'kianoosh'
```

```
'kianoosh'.upper()
```

Make a string upper case!

```
'KIANOOSH'
```

```
'kianoosh'.split('a')
```

Split a string using an indicator

```
['ki', 'noosh']
```

```
lst = 'kianoosh'.split('a')
lst
```

I am setting up a list! We will be coming back to it in a few slides!

```
['ki', 'noosh']
```

```
''.join(lst)
```

Join the two strings

```
'kinoosh'
```

```
'a'.join(lst)
```

Join the two strings, using ‘a’ in this case!

```
'kianoosh'
```

Comparison and Logical Operators

- Comparison

Operator	What it does	Example a=2 b=2
==	Equal to	a == b → True
!=	Not equal to	a != b → False
> <	Greater than Smaller than	a > b → False a < b → False
>= <=	Greater and equal than Smaller and equal than	a >= b → True a <= b → True

- Logical

Operator	What it does	Example a=2 b=2
and	Truthy if both condition a AND b are True	a > 1 and b < 3 → True
or	Truthy if either or condition a OR b is True	a > 3 or b < 4 → True a > 3 or b < 1 → False
not	Truthy if the opposite of the condition is True	not a > 3 → True

Take an input

```
print('How many minutes did you code today>')
min=input()
hr = float(min)/60
hr = round(hr, 2)
print(f"OK! You were coding for {hr} hours")
```

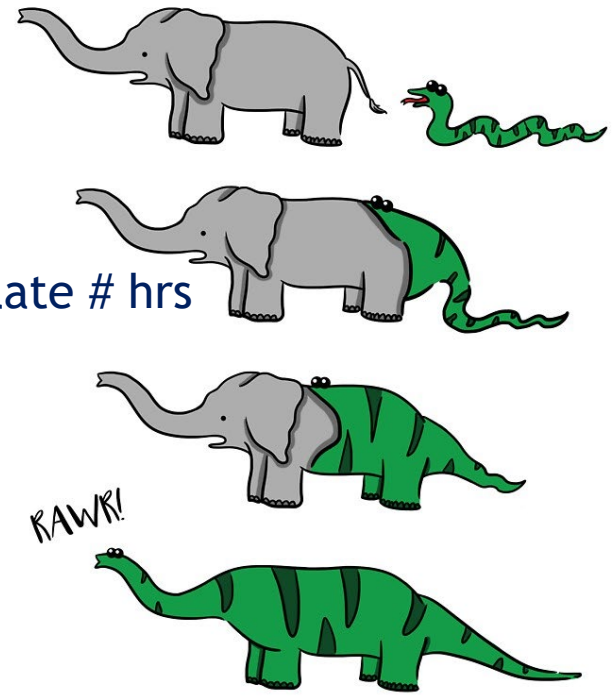
A message for the user!
Taking the input
Convert to float and calculate # hrs
Round to two numbers!
Print it out

```
How many minutes did you code today>
66
OK! You were coding for 1.1 hours
```

```
type(min)
```

```
str
```

- Note: Type of input is string!



Stretch Your Coding Muscles!

Part_I_Essentials_a

Conditional Logic & Booleans

```
if num > 0:
    print("Positive number")
elif num == 0:
    print("Zero")
else:
    print("Negative number")
```

```
if some condition is True:
    do something
elif some other condition is True:
    do something
elif* some other condition is True:
    do something
else:
    do something
```

* Note: Multiple elif is fine

Loops – For & While

- We use **For** and **While** to iterate over a range
- Example: print all the numbers from 0-10!

```
print(0)
print(1)
print(2)
print(3)
print(4)
print(5)
print(6)
print(7)
print(8)
print(9)
print(10)
```

VS

- Or:

```
for i in range(11):
    print(i)
```

Ranges

- As the name suggests it provides a range! It is very useful when using **for** loops
- range(5) → 0, 1, 2, 3, 4
- range(2, 10) → 2, 3, 4, 5, 6, 7, 8, 9
- range(2, 10, 2) → 2, 4, 6, 8
- range(start, end, step)

Question: What is range(6,1,-1)?

A. 1, 2, 3, 4, 5
B. 6, 5, 4, 3, 2
C. 6, 5, 4, 3, 2, 1
D. 5, 4, 3, 2, 1

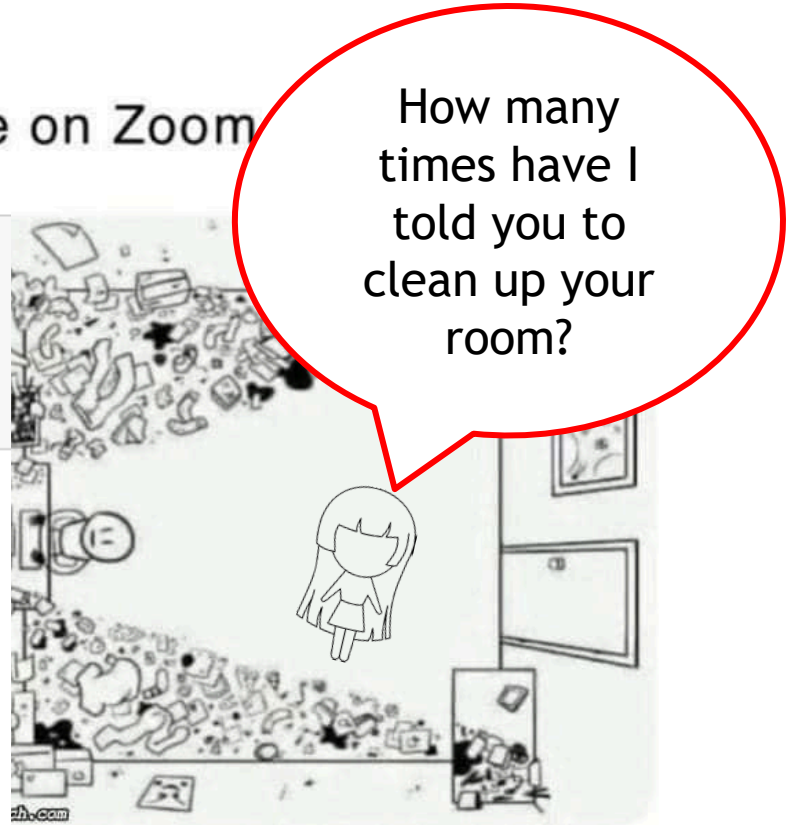
For Loops

```
for items in iteratable_object:
    do something
```

```
num = int(input('How many times I have told you to clean up your room? '))

for i in range(num):
    print('CLEAN UP YOU ROOM, KIANOOSH!')
```

```
How many times I have told you to clean up your room? 6
CLEAN UP YOU ROOM, KIANOOSH!
CLEAN UP YOU ROOM, KIANOOSH!
CLEAN UP YOU ROOM, KIANOOSH!
CLEAN UP YOU ROOM, KIANOOSH!
CLEAN UP YOU ROOM, KIANOOSH!
CLEAN UP YOU ROOM, KIANOOSH!
```



Exercise!

- Write a code to ask for an integer input. Then print out the factorial of that using a for loop!

While Loop

- While loop continues to loop while the condition is Truthy. It will stop if it becomes Falsy!
- Be careful! Since it may always run if the condition never becomes false. Ctrl+c to stop it.

List (array, matrix, etc.)

- A collection or grouping of items!
- Lists are used to store multiple items in a single variable.

```
fruits = ['apple', 'banana', 'tomato!']
three_mult = [3, 6, 9, 12]
```

First index

```
three_mult[0]
```

```
3
```

```
three_mult[-1]
```

```
12
```

```
len(three_mult)
```

```
4
```

List

- A collection or grouping of items!
- Lists are used to store multiple items in a single variable.

```
fruits = ['apple', 'banana', 'tomato!']
three_mult = [3, 6, 9, 12]
```

```
three_mult[0]
```

```
3
```

```
three_mult[-1]
```

```
12
```

```
len(three_mult)
```

```
4
```

List

- A collection or grouping of items!
- Lists are used to store multiple items in a single variable.

```
fruits = ['apple', 'banana', 'tomato!']
three_mult = [3, 6, 9, 12]
```

Length

```
three_mult[0]
```

```
3
```

```
three_mult[-1]
```

```
12
```

```
len(three_mult)
```

```
4
```

List

- append()
- pop()
- extend()

```
three_mult = [3, 6, 9, 12]
```

```
three_mult.append(15)
three_mult
```

```
[3, 6, 9, 12, 15]
```

- Adds 15 to the list

```
three_mult.append([18,21,24])
three_mult
```

```
[3, 6, 9, 12, 15, [18, 21, 24]]
```

- Adds the entire list [18,21,24] as one item

```
three_mult.pop()
```

```
[18, 21, 24]
```

- So, let's use pop() to delete this last item!
  - [18,21,24] is deleted

```
three_mult
```

```
[3, 6, 9, 12, 15]
```

- We can check it

```
three_mult.extend([18,21,24])
three_mult
```

```
[3, 6, 9, 12, 15, 18, 21, 24]
```

- We use extend to add the list

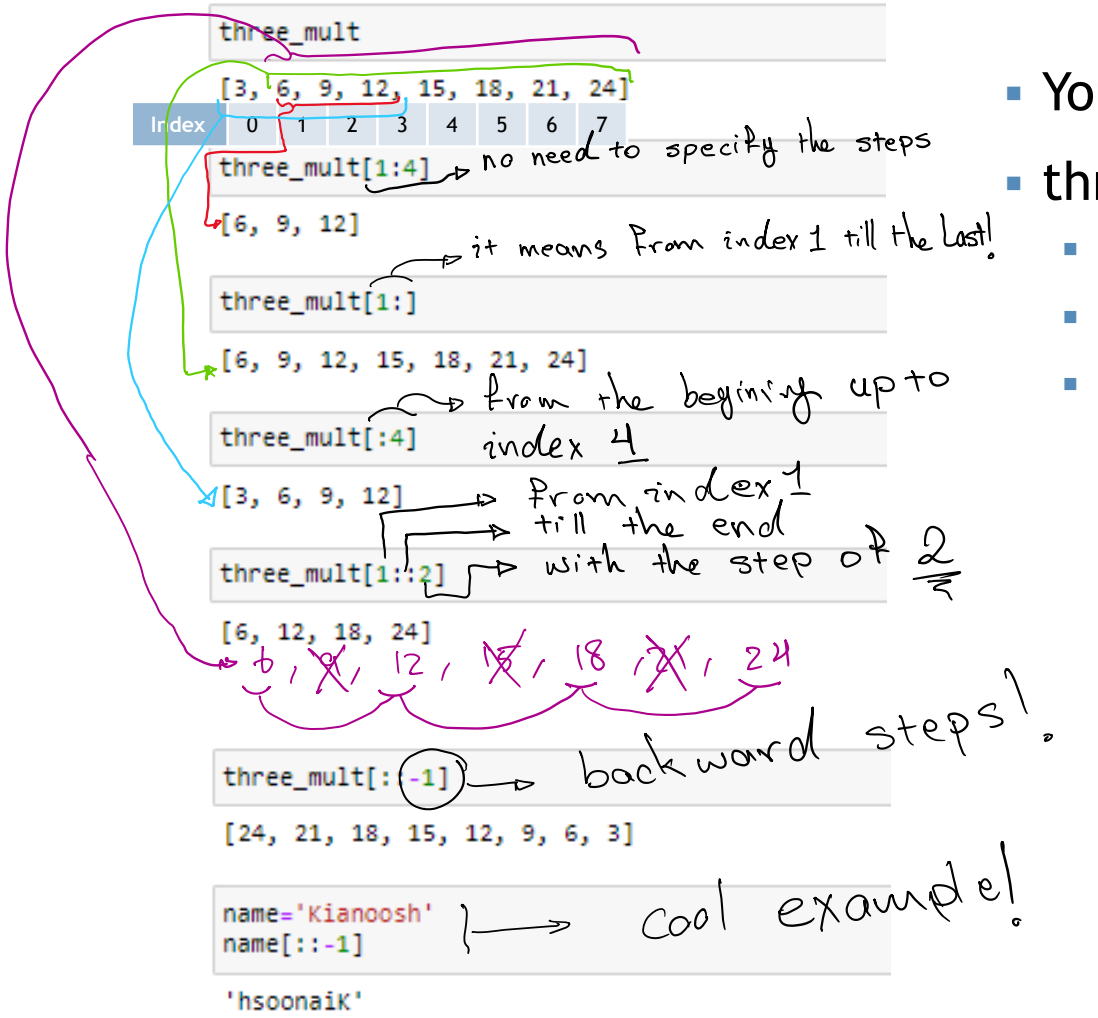
Or

```
three_mult = three_mult + [18, 21, 24]
three_mult
```

```
[3, 6, 9, 12, 15, 18, 21, 24]
```

- Or you may extend it with '+'

List – Slicing[start:end:step]



- You have a list called ‘three_mult’
- three_mult[start:end:step]
  - If start is not specified means the first (i.e. [0::])
  - If end is not specified means the last (i.e. [:-1:])
  - If step is not specified means step of one (i.e[::1])

List – A few more tricks

Reverses

```
name='Kianoosh'
name[::-1]
```

```
'hsoonaiK'
```

Modify portion of the list

```
three_mult[1:3]=['a', 'b', 'c']
three_mult
```

```
[3, 'a', 'b', 'c', 12, 15, 18, 21, 24]
```

swap values

```
three_mult[1], three_mult[4] = three_mult[4], three_mult[1]
three_mult
```

```
[3, 12, 'b', 'c', 'a', 15, 18, 21, 24]
```

pop the last value

```
three_mult.pop()
```

```
24
```

pop based on the index

```
three_mult.pop(0)
```

```
3
```

```
three_mult
```

```
[12, 'b', 'c', 'a', 15, 18, 21]
```

remove based on the value

```
three_mult.remove(15)
three_mult
```

```
[12, 'b', 'c', 'a', 18, 21]
```

clear the entire list

```
three_mult.clear()
three_mult
```

```
[]
```

List – Check index

```
three_mult
```

```
[3, 6, 9, 12, 15, 18, 21, 24]
```

Index	0	1	2	3	4	5	6	7

```
three_mult.index(12)
```

```
3
```

```
three_mult.insert(3,12)
three_mult.insert(6,12)
three_mult
```

```
[3, 6, 9, 12, 12, 15, 12, 18, 21, 24]
```

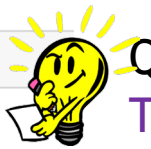
Now on index 3 let's insert number 12

also, on index 6 let's insert another 12

items	3	6	9	12	12	15	12	18	21	24
Index	0	1	2	3	4	5	6	7	8	9

```
three_mult.index(12)
```

```
3
```



Question: What will be the outcome of this line?
This only shows the index of the first 12!

```
three_mult.index(12, 5)
```

```
6
```

The first 12 starting at index 5

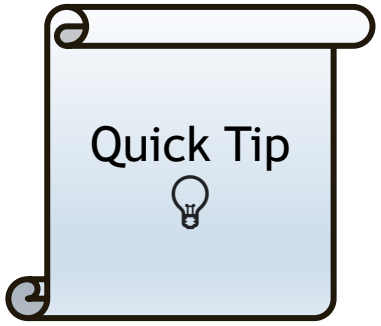
```
three_mult.index(12,7,9)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-118-e3960d216e02> in <module>
----> 1 three_mult.index(12,7,9)

ValueError: 12 is not in list
```

check between indices of 7-9

NOTE: press shift+tab in the parenthesis and get to see what parameters are, examples, and more!



tab & shift+tab

- Tab autocompletes
- Shift + tab → shows documentation.

To see more of the documentation

```
three_mult.index(12,7,9)
```

```
Signature: three_mult.index(value, start=0, stop=9223372036854775807, /)
Docstring:
Return first index of value.
```

jupyter

List – A few more things

```
lst = [2,3,8,9,7,2,2,4,6,7,2,8,2]

lst.count(2)
```

```
5
```

- Define the list
- Use .count() to count for a value in it.
  - In this example it counts how many times number 2 is repeated in the list

```
lst.sort()
lst
```

```
[2, 2, 2, 2, 2, 3, 4, 6, 7, 7, 8, 8, 9]
```

- .sort() function sorts and applies the change to the list.
  - Note: this is different that sorted(list) function.

Sort vs sorted

You tell python to **sort** the values!
Thus, list will be updated!

```
lst = [2,3,8,9,7,2,2,4,6,7,2,8,2]

lst.sort()
print(lst)
```

```
[2, 2, 2, 2, 2, 3, 4, 6, 7, 7, 8, 8, 9]
```

You ask python how it looks if **sorted**!
Thus, the list won't be updates

```
lst = [2,3,8,9,7,2,2,4,6,7,2,8,2]

print(sorted(lst))
print(lst)
```

```
[2, 2, 2, 2, 2, 3, 4, 6, 7, 7, 8, 8, 9]
[2, 3, 8, 9, 7, 2, 2, 4, 6, 7, 2, 8, 2]
```

List – A few more things

```
lst = [1,2,3,4,5,6,7,8,9]
```

Defining the list

```
max(lst)
```

```
9
```

Max → determines the maximum value in the list

```
min(lst)
```

```
1
```

Min → determines the minimum value in the list

```
len(lst)
```

```
9
```

Determine the length of the list (i.e. how many items in it)

```
sum(lst)
```

```
45
```

Sum → what is the outcome of adding all the numbers in the list!

List Comprehension

- […. for …. in ….]

```
lst = [num for num in range(1,7)]
lst
```

```
[1, 2, 3, 4, 5, 6]
```

How is this better than this?

```
lst = [1, 2, 3, 4, 5, 6]
lst
```

```
[1, 2, 3, 4, 5, 6]
```

Example: let's say we want to multiply every item in lst by two using a for loop

```
lst2=[]

for item in lst:
    doubled = item*2
    lst2.append(doubled)

lst2
```

```
[2, 4, 6, 8, 10, 12]
```

However

```
lst2= [item*2 for item in lst]
lst2
```

```
[2, 4, 6, 8, 10, 12]
```

Embrace the beauty of list comprehension

List Comprehension – if & else

```
lst
```

```
[1, 2, 3, 4, 5, 6]
```

```
even = [num for num in lst if num%2==0]
even
```

```
[2, 4, 6]
```

You may add conditions to the list comprehension with the if at the end

```
[num if num%2==0 else num/2 for num in lst]
```

```
[0.5, 2, 1.5, 4, 2.5, 6]
```

However, if and else used together is added in the middle!

```
[num if num%2==0 for num in lst]
```

```
  File "<ipython-input-146-6f15b94211d6>", line 1
    [num if num%2==0 for num in lst]
                     ^
SyntaxError: invalid syntax
```

NOTE: if just by itself in the middle with no else followed will result in an error!
if has to be always at the end, if used alone!

Exercise!

- Utilize list comprehension for all the followings:
  - Create a list of numbers from 2-20 (include 20)
  - Use the new list created above to:
    - If the numbers are odd multiply them by two
    - If the number is divisible by three multiply by 6 else divide it by 2

Nested List (Multi-dimensional List)

```
nested_list=[[1,2],[3,4],[5,6]]
nested_list
```

```
[[1, 2], [3, 4], [5, 6]]
```

```
i=0
j=1
nested_list[i][j]
```

```
2
```

```
nested_list[1][-1]
```

```
4
```

```
for item in nested_list:
    for num in item:
        print(num)
```

```
1
2
3
4
5
6
```

THINGS TO LEARN ON YOUR OWN

- DICTIONARIES

```
tel = {'jack': 4098, 'sape': 4139}
```

- TUPLES: Tuples are immutable. Less memory is used.

```
(12345, 54321, 'hello!')
```

- SETS: A set is an unordered collection with no duplicate elements.

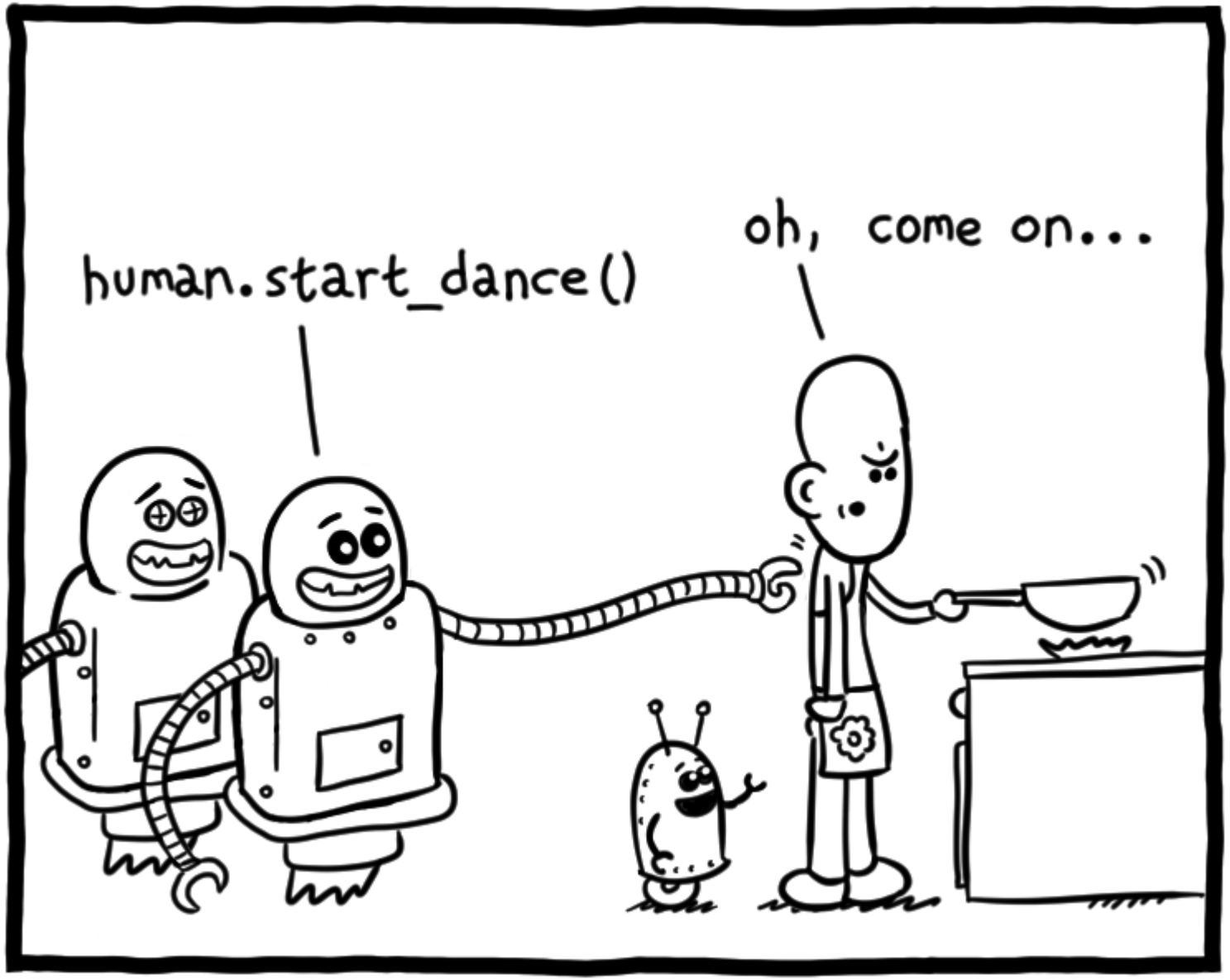
```
>>> basket = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'}
>>> print(basket)                      # show that duplicates have been removed
{'orange', 'banana', 'pear', 'apple'}
```

```
>>> a = {x for x in 'abracadabra' if x not in 'abc'}
>>> a
{'r', 'd'}
```

Stretch Your Coding Muscles!

Part_I_Essentials_b

Functions



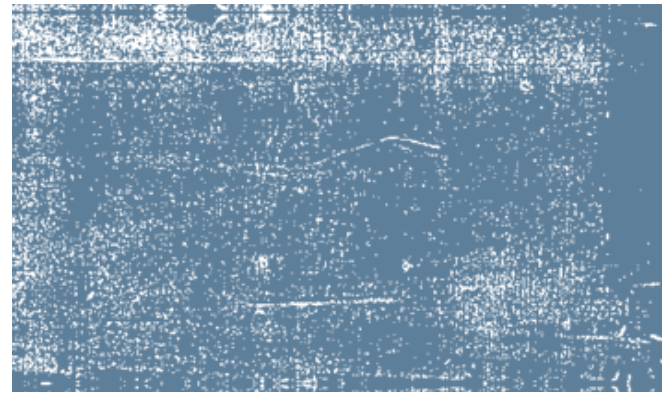
Why a Function?

- To Stay **DRY: D**on't **R**epeat **Y**ourself
- Clean us and prevent code duplication
- Think, how annoying it would have been if you had define integral every single time using it in a code! Or a simpler example is our famous print('Hello, world')! How annoying it would have been to define what print does every single time
- So, avoid **WET** (**W**rite **E**verything **T**wice) and stay **DRY** (**D**on't **R**epeat **Y**ourself)
  - After all you live in Vancouver! You should be a master of it by now. ;-)

```
def hi():
    print('Hello, World!')
```

```
hi()
```

```
Hello, World!
```



KEEP
CALM
AND
STAY
DRY

Functions – Return

- Return → Exits the function

```
def cube(n):
    print('Before Return')
    return(n**3)
    print('AFTER Return')

cube(2)
```

```
Before Return

8
```

When cube(2) is called, since this line in after return

It won't be printed in the outcome

Function – Parameter

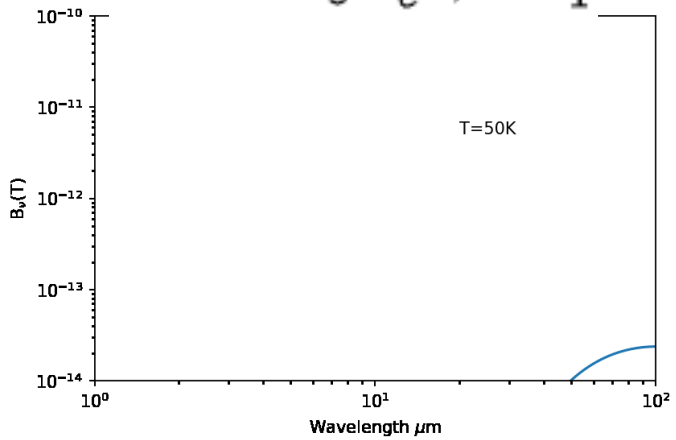
- Variable that are passed into a function are the parameters!
  - Parameter is the variable
  - Argument is the value that is put into the parameter

```
def SED(wavelength, temp):
    ''' Wavelength is in micon and temperature is K'''
    h=6.62607004e-34    # Planck Constant Joule per Sec
    c=299792458         # Speed of Light
    k=1.38064852e-23    #Boltzmann constant
    nu = c/(wavelength * 1.0e-6)# ; hz

    planck_function= ( 2.0* h * nu**3 ) / ( c**2 * ( np.exp(h*nu/(k*temp)) - 1.) )

    return planck_function
```

$$B_\nu(T) = \frac{2\nu^2}{c^2} \frac{h\nu}{e^{h\nu/kT} - 1}$$



- You may call parameters whatever you like! I suggest name them appropriately!

Exercise!

- Define a function to return the force of gravity. The force of gravity is a function of three parameters/variables (i.e mass1, mass2, and the distance between the masses): $F_G = G\frac{m_1 m_2}{r^2}$

- Add a line into your function to ensure that arguments provided by the user are positive!

What if you want to set the default value of one of the masses to the mass of the Earth and the radius to that of the Earth?

Function – A few more notes

```
def force_gravity(m1, m2=5.97e24, r=6.371e6):
    '''m1 & m2 are masses in kg
    r is the distance between the center of the masses in meter
    Default values are mass and radius of the Earth, respectively.'''
    G = 6.67e-11 # this is the gravitational constant
    if m1>0 and m2>0 and r>0:
        force_gravity = G*m1*m2/r**2
        return force_gravity
    return 'Check if all your input values are positive'
```

Default parameter can be simply defined here

Add documentation, use three quotation “‘ ”’!

Note that I am not adding else! Why?

```
force_gravity(100)
```

```
981.0360234523878
```

Uses m1=100 and default parameters

```
force_gravity(1,1,1)
```

```
6.67e-11
```

You may still change the default parameters to your own values!.

```
force_gravity(100, r=6.371+10000)
```

```
397692099.3052699
```

You may also choose to update only radius!

```
force_gravity()
```

```
Signature: force_gravity(m1, m2=5.97e+24, r=6371000.0)
Docstring:
m1 & m2 are masses in kg
r is the distance between the center of the masses in meter
```

Shift + tab to see your documentation!

Function - *args

- What if you don’t know the number of variables?
- For instance, you need to calculate the average of all the provided numbers!
  - You may add as many numbers as you wish

```
def average_finder(*args):
    addition = 0
    for num in args:
        addition += num
    return addition/len(args)

average_finder(9,10,11)
```

```
10.0
```

```
average_finder(9,10,11,9.5,10.5)
```

```
10.0
```

Unpacking!

```
def average_finder(*args):
    addition = 0
    for num in args:
        addition += num
    return addition/len(args)

average_finder(9,10,11)
```

- **Question:** Create the following list: lst=[50, 62, 74, 80, 85, 100]
- Now feed this list into your average_finder function from the previous slide!
- You would get an error:

```
lst=[50, 62, 74, 80, 85, 100]
average_finder(lst)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-97-3bc81a2c11e2> in <module>
```

- How to fix this? **Feeding a list as an *args.**
  - **The * at the beginning tells python to unpack the list!**

```
average_finder(*lst)
```

```
75.16666666666667
```

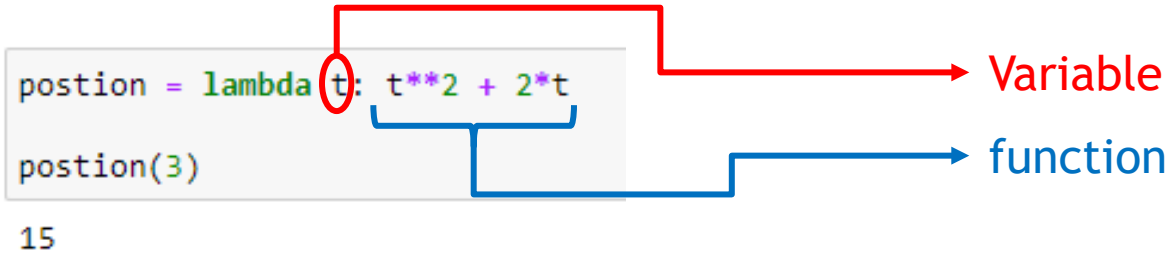
Function – Parameters Ordering

- The ordering of the parameters matters, and it goes:
  - def function(parameters, *args, default_parameters, **kwargs)
    - **kwargs woks with dictionaries and beyond the scope of this course

```
def function(parameters, *args, default_parameters, **kwargs)
```

Lambda – A Small Function

- A lambda function is a small anonymous function



Example:

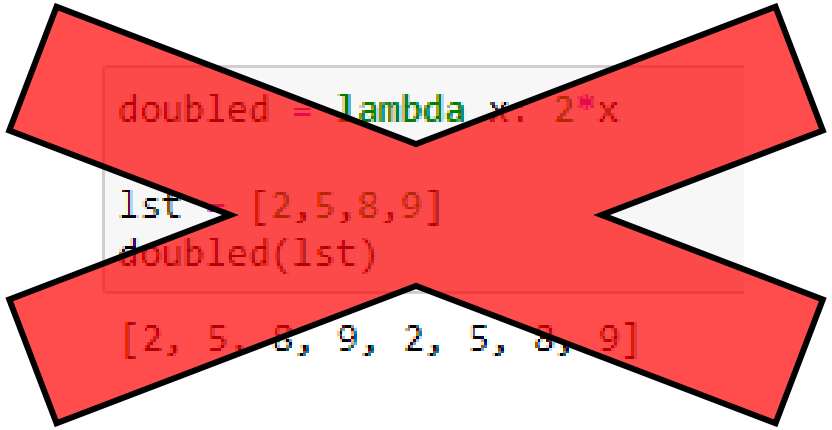
Summarize argument a, b, and c and return the result:

```
x = lambda a, b, c : a + b + c
print(x(5, 6, 2))
```

Try it Yourself – Click Here»

Question

- Create the following list:
- lst = [2,5,8,9]
- Feed the list into a lambda function to double each item!



So, what is the solution to correctly map the function on the list?

Map

- Maps a function/lambda to something

```
lst = [2,5,8,9]

doubled = lambda x: 2*x
```

Defining the function. In this case, I used lambda

```
map(doubled, lst)
```

```
<map at 0x18677577790>
```

mapping(function, something)

```
list(map(doubled, lst))
```

```
[4, 10, 16, 18]
```

Note: I need to convert it to list to see it

```
list(map(lambda x:2*x, lst))
```

```
[4, 10, 16, 18]
```

I personally, like to combine all the steps, since I am lazy!

Learn on your own!

- abs, round
- any & all
- filter

```
abs(-5)
```

```
5
```

```
round(2.56)
```

```
3
```

```
lst=[True, False, True]
```

```
any(lst)
```

```
True
```

```
all(lst)
```

```
False
```

```
lst = [1,2,3,4]
filter(lambda x:x%2==0, lst)
```

```
<filter at 0x186046b42e0>
```

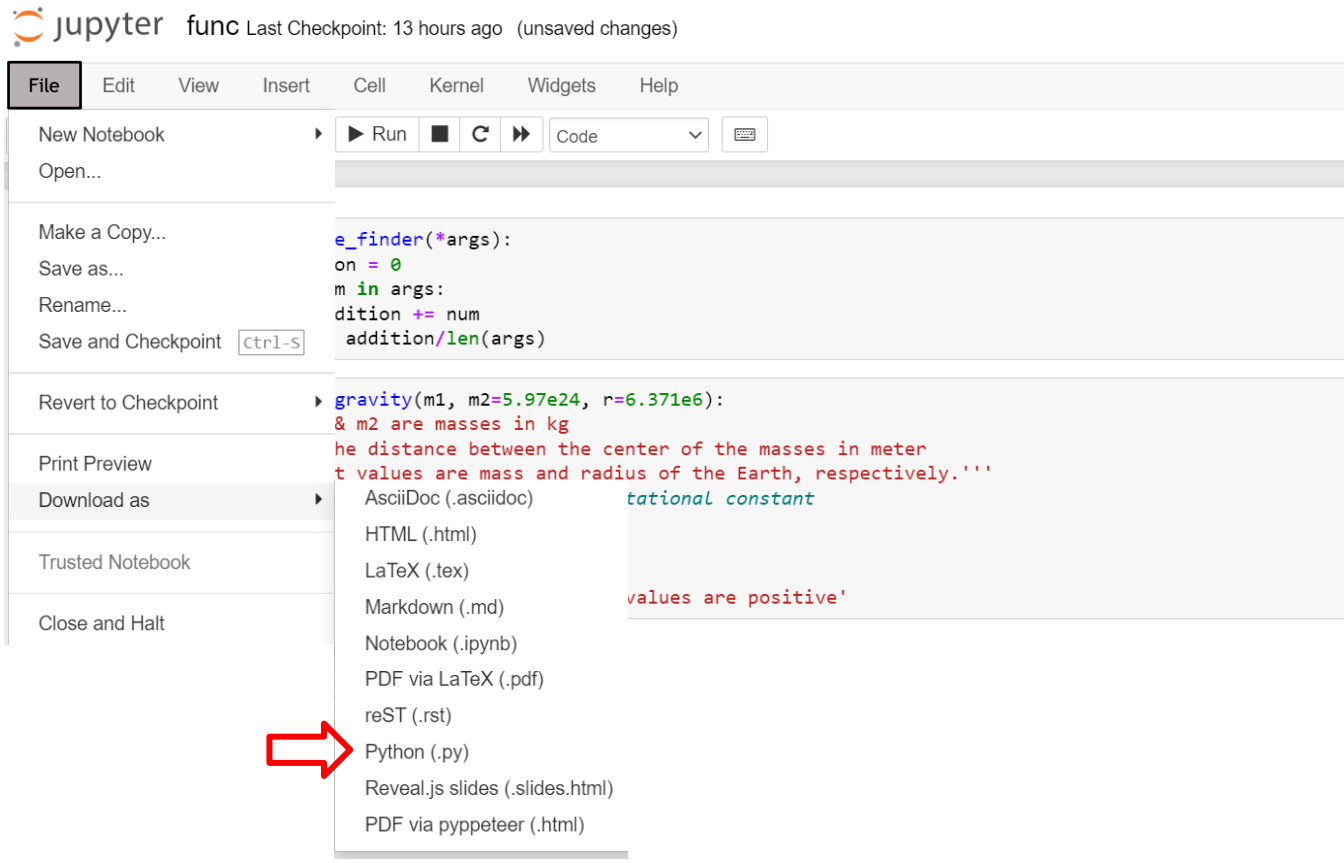
```
list(filter(lambda x:x%2==0, lst))
```

```
[2, 4]
```

- Well! Now that you know functions in python, the only thing left is to ….

Save functions and call them!

- Let's say we write the two functions:
- You may save it by going to:
  - File > Download as > Python (.py)
- Note: that you may need to move the downloaded file to where you are at.



Save functions and call them!

- Let's say we write the two functions:
- You may save it by going to:
  - File > Download as > Python (.py)
- Note: that you may need to move the downloaded file to where you are at.
- Now on a new Jupiter Note file you get to import this file
- AKA **Modulus**

```
import func
```

```
func.average_finder(1,2,3)
```

```
2.0
```

Note that there are two functions in this file. So, you need to specify which one you are calling!

```
average_finder(1,2,3)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-5-04f4c3819ffb> in <module>
----> 1 average_finder(1,2,3)

NameError: name 'average_finder' is not defined
```

Note: you can no longer call with the name of the functions, unless ……

```
from func import average_finder
```

… you directly import it from the file!

```
average_finder(1,2,3)
```

```
2.0
```

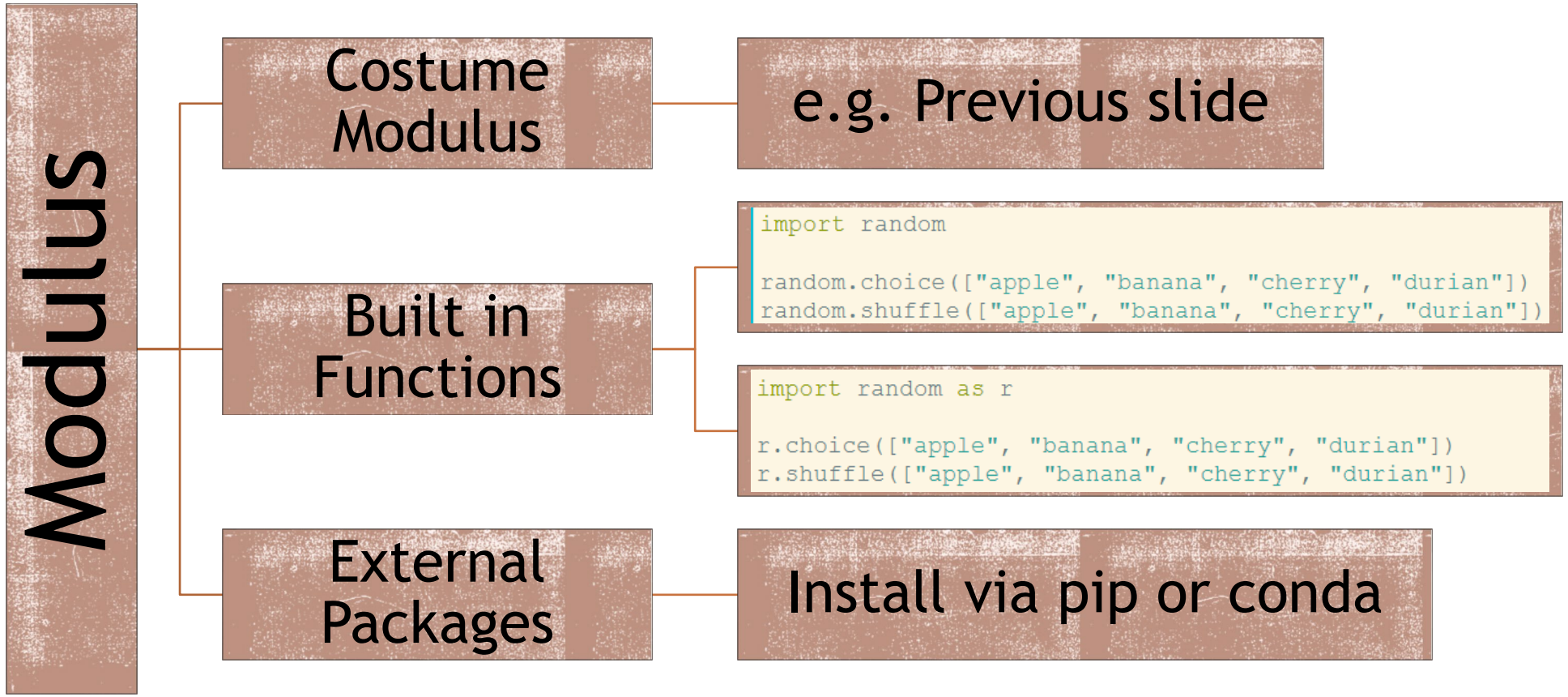
```
from func import average_finder as ave
```

```
ave(1,2,3)
```

```
2.0
```

Note: you may also change the name of the function to reduce your typing!

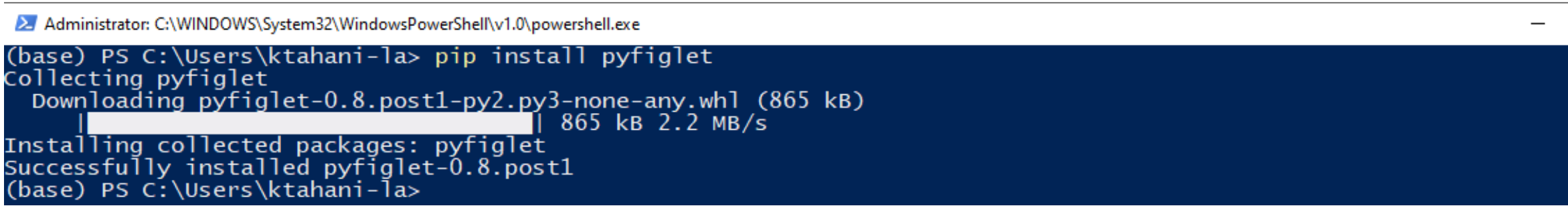
Modulus


Modulus
Costume Modulus
e.g. Previous slide
Built in Functions
import random
random.choice(["apple", "banana", "cherry", "durian"])
random.shuffle(["apple", "banana", "cherry", "durian"])
import random as r
r.choice(["apple", "banana", "cherry", "durian"])
r.shuffle(["apple", "banana", "cherry", "durian"])
External Packages
Install via pip or conda

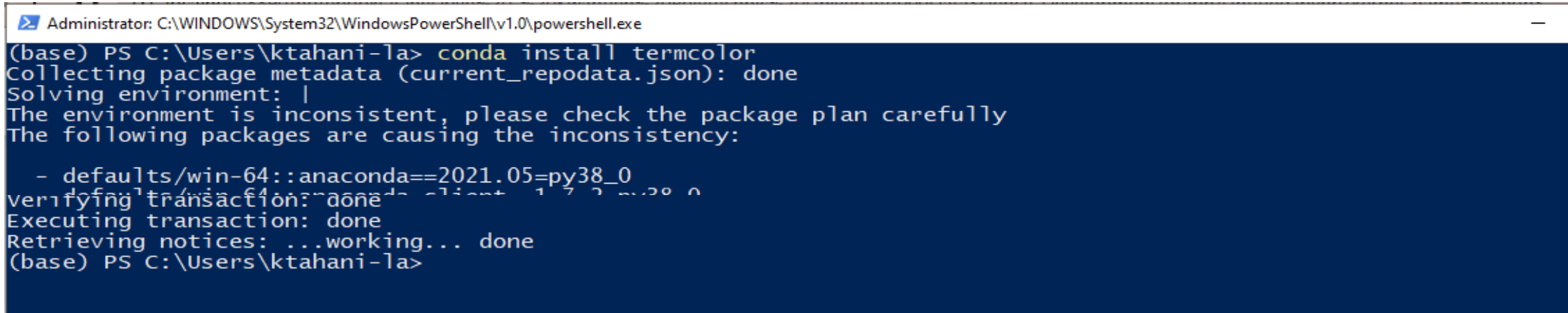
External Packages

How to install them?

1. Pip install name_of_package → Should always work



2. Conda install name_of_package → Also, works with Anaconda packages



More info at https://pypi.org/

Homework

- Install the following packages
  - numpy
  - pandas
  - matplotlib
  - seaborn

Stretch Your Coding Muscles!

Part_I_Essentials_c