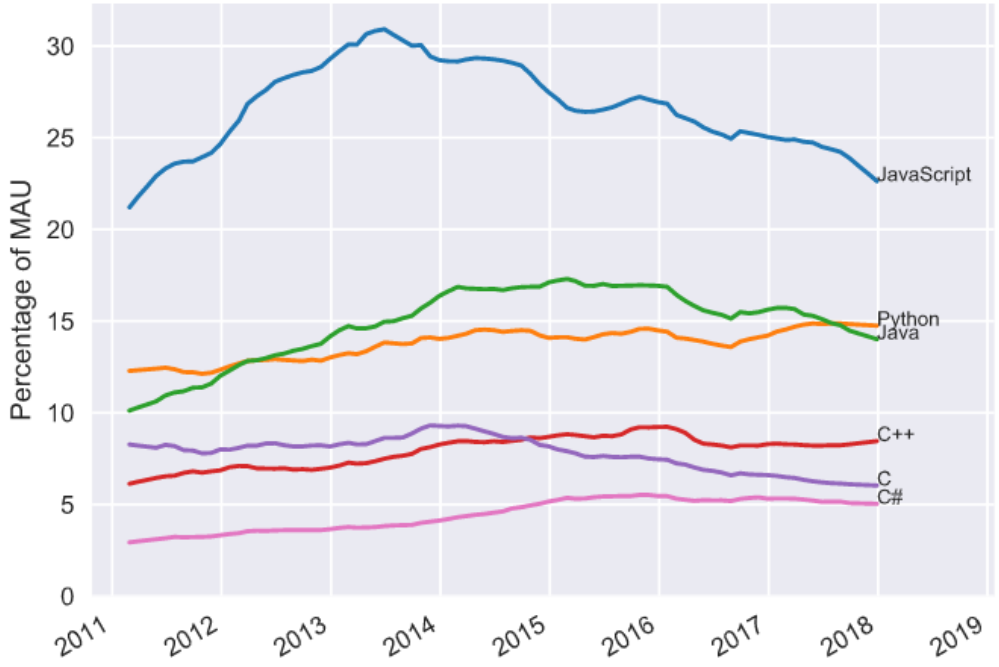
Python Workshop

By: Heather S. Deter

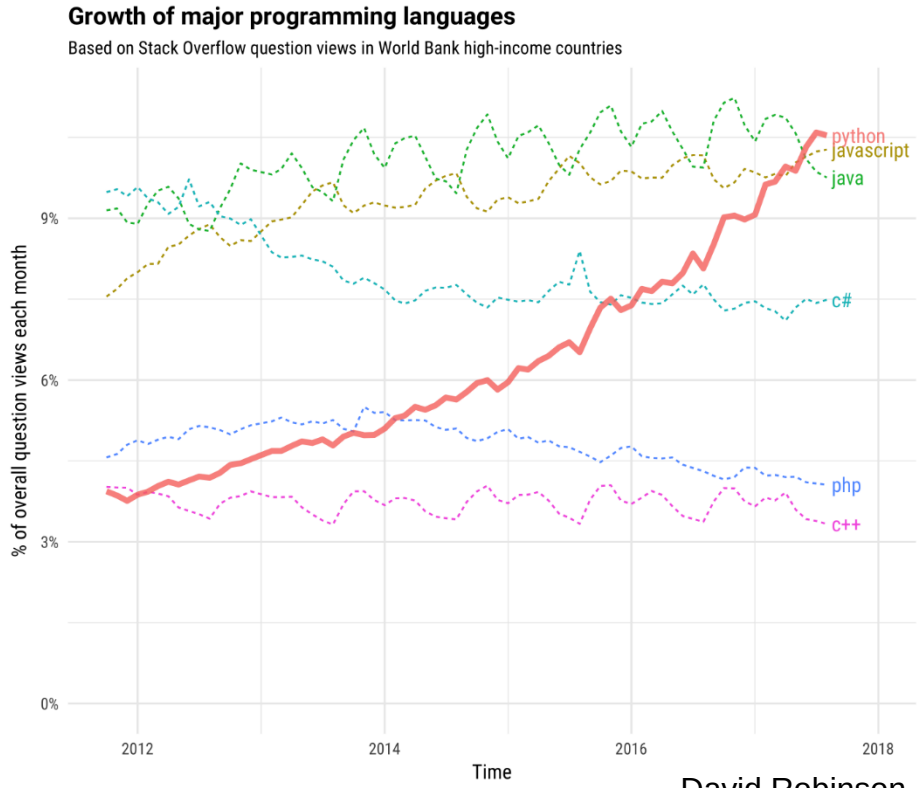
Why Python?

- Python is highly adaptable
- Python is widely used in the scientific community
- There are specific biology applications (e.g. BioPython)

GitHub Monthly Active Users (MAU)



Ben Frederickson



David Robinson

Installing Python through Anaconda

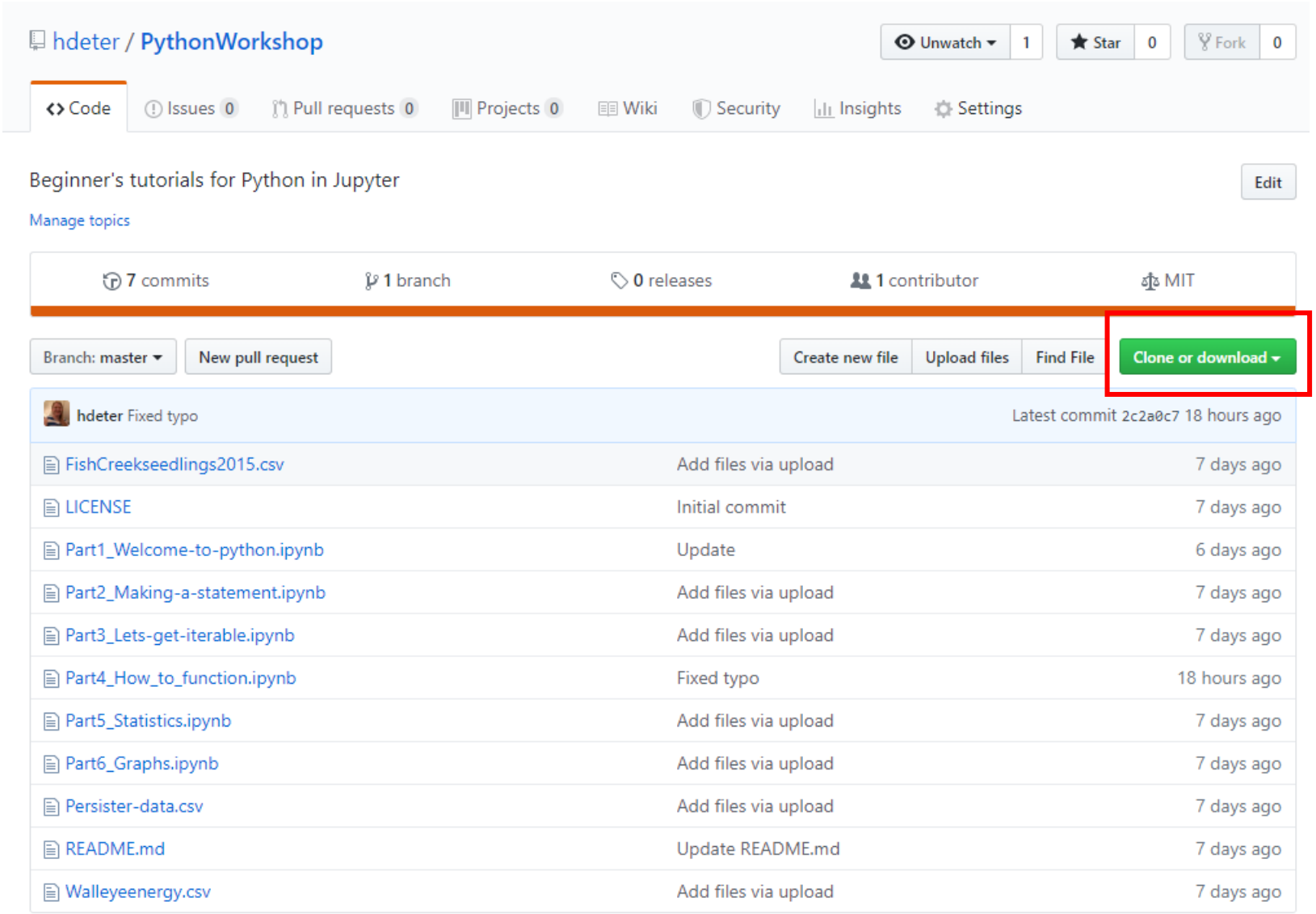
- Anaconda is a platform for Python and R that makes installation easy
- Go to the website below and download Anaconda
- https://www.anaconda.com/distribution/
- Install Anaconda

Jupyter notebooks

- There are several platforms to edit and run Python
- Jupyter notebooks are a good place to start
- You can open Jupyter through Anaconda or in the terminal if you have anaconda installed in your path

We will be using Jupyter notebooks for this tutorial

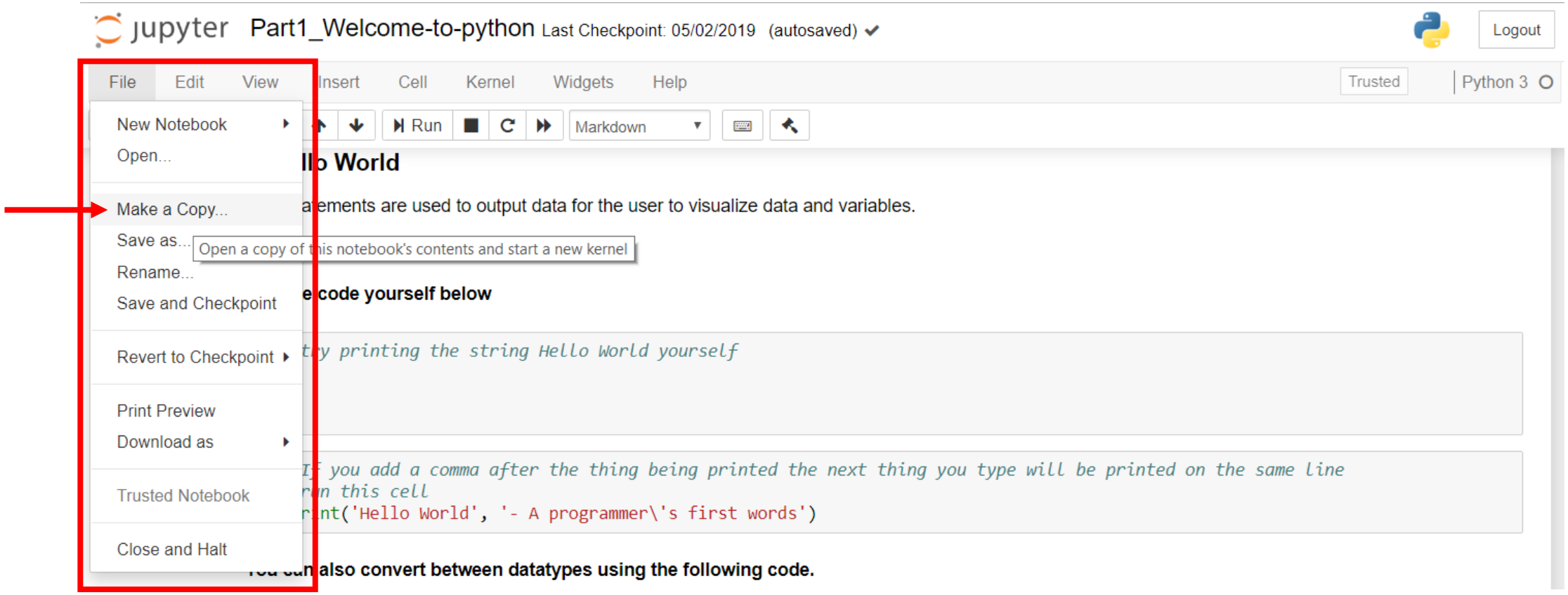
- Go to: https://github.com/hdeter/PythonWorkshop



Use this button to download

Once you have the files you can open them in jupyter.

Before you start, always make a copy of the file



When writing your own code it’s important to save different versions

- The following slides are meant to compliment the jupyter notebooks.
- They include answers to the problems.
- These answers are often just one of many possible correct answers. You may even come up with better, more robust answers than the ones presented here.
- There’s no cheating.
    1. Try it on your own.
    2. Use whatever resources you have to figure it out.
    3. Try to save looking at the answer here until last so that you have a greater opportunity to learn.

Welcome to Python - Python Tutorial Part 1

By: Heather S. Deter

The purpose of this series is to get you familiar with concepts in Python 3.7. There are several online tutorials available for further study and clarification.

Tutorials:
https://developers.google.com/edu/python/
https://www.learnpython.org/

Helpful tips for Jupyter:

- A list of shortcuts is available under the Help menu; alternatively hit Esc then H
- To run a cell use Ctrl + Enter
- To save a new version of your notebook use File -> Make a Copy...

1. How to use comments

- Comments are used to annotate the script for humans (they are ignored by the computer)
- **Good comments are critical to maintaining and editing code**
- Comments are useful for both others who use your code and yourself
- Poorly commented code can be incredibly difficult to update or edit

```
##hashtags indicate a comment in python
#comments can also be used to ignore a line of code without removing it
# In jupyter multiple lines can be commented out by highlighting the lines then using "Ctrl + /"
```

2. Basic Datatypes

There are several datatypes used in python, we will briefly review them in this chapter.

Integers

Whole numbers are integers in python.

```
#integers
1
2
3
```

Floats

Numbers with decimals are floats.

```
#floats
1.0
3.14
0.00000023
```

Strings

Any set of characters can be represented as a string.
Strings are indicated by either ' or "
Some characters including ',", and \ must be prefaced with a forward slash (\) to be included in the string.
To have a line break use '\n' to indicate a new line.

```
#strings
'Hello World'
'3 Apples, 5 Bananas'
'20,000 Leagues Under the Sea'
'All the King\'s Horses'
```

Boolean

True/False values are referred to as booleans.

```
#booleans
True
False
```

3. Hello World

Print statements are used to output data for the user to visualize data and variables.

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

Run the code yourself below

In [1]:
```
#try printing the string Hello World yourself

print('Hello World')

```

```
Hello World
```

In [2]:
```
#If you add a comma after the thing being printed the next thing you type will be printed on the same line
#run this cell
print('Hello World', '- A programmer\'s first words')
```

```
Hello World - A programmer's first words
```

You can also convert between datatypes using the following code.

```
#convert to a string
str()

#convert to an integer
int() #this will round a float to the nearest whole number

#convert to a float
float()
```

4. Math ¶

Basic mathmatical functions are available in python.

In [3]:

```
#run the code below
print('Addition:', 2+2)
print('Subtraction:', 4-2)
print('Division:', 24/6)
print('Multiplication:', 3*12)
print('Exponents', 2**4)
```

```
Addition: 4
Subtraction: 2
Division: 4.0
Multiplication: 36
Exponents 16
```

5. Objects

Python is an object oriented programming language.
Objects are essentially a *thing* that you can define or output using other code.

An example of defining an object is below.

In [4]:
```
#run this cell
mystring = 'Hello World'
print(mystring)
```

```
Hello World
```

Objects stand in for what you assign them, therefore anything you can do with the value you can do with the object.

Example
1. Nicholas has 6 apples and Prajakta gives him 12 apples. How many apples are does Nicholas have?
2. Nicholas gives half of his apples to Heather, who eats 3 apples. How many apples does Heather have?
3. Tahmina offers to give Nicholas three times more oranges than he has apples. How many oranges should Tahmina give Nicholas?

Example

1. Nicholas has 6 apples and Prajakta gives him 12 apples. How many apples are does Nicholas have?
2. Nicholas gives half of his apples to Heather, who eats 3 apples. How many apples does Heather have?
3. Tahmina offers to give Nicholas three times more oranges than he has apples. How many oranges should Tahmina give Nicholas?

In [5]:

```python
#You can use do math with objects
##run this cell to see the answers

##1. Nicholas has 6 apples and Prajakta gives him 12 apples. How many apples are does Nicholas have?
#Starting number of apples
Nicholas = 6
Prajakta = 12

#Prajakta gives Nicholas her apples
Nicholas = Nicholas + Prajakta

##2. Nicholas gives half of his apples to Heather, who eats 3 apples. How many apples does Heather have?
#Nicholas gives Heather apples
Nicholas = Nicholas/2
Heather = Nicholas - 3

##3. Tahmina offers to give Nicholas three times more oranges than he has apples.
###  How many oranges should Tahmina give Nicholas?

#Tahmina give Nicholas oranges
Tahmina = 3*Nicholas

print('1. Nicholas has', Nicholas, 'apples')
print('2. Heather has', Heather, 'apples')
print('3. Tahmina gives Nicholas', Tahmina, 'oranges')
```

```
1. Nicholas has 9.0 apples
2. Heather has 6.0 apples
3. Tahmina gives Nicholas 27.0 oranges
```

An important note on naming objects

Understanding what an object represents is critical to understanding your code.
Always name an object something that makes sense (be brief and descriptive).

```
#bad names
x = 'Hello World' #single letters are commonly used in other sections of code and are not descriptive
ThatonestringIwrotebecauseyoutoldmeto = 'Hello World' #I can't even read that

#good names
greeting = 'Hello World'
mystring = 'Hello World'
intro = 'Hello World'
```

Also avoid names that are commonly used commands, classes, etc. in python

```
#bad name
str = 'Hello World' #str is used to denote strings in python and should not be an object
```

If it turns any color other than black when using jupyter rule of thumb is do not use it.

"Variable strings"

Python has a printf()-like facility, which basically means you can put in a variable to be defined later.
This is done using the % symbol.

Examples

```
#integers are indicated by %d
'Volker has %d sheep in his office'

#strings are indicated by %s
'%s met with Volker today'

#the variable is also filled in using %
#for example
print('Volker has %d sheep in his office' %3)
```

Run the cell below to see the code in action

In [6]:

```
#run this cell
statement1 = '%s met with Volker today'
student = 'Stephanie'

print(statement1 %student)

statement2 = 'Volker has %d sheep in his office'
sheep = 17

print(statement2 %sheep)

##you can have multiple variables
statement3 = '%s put %d sheep in Volker\'s office'

print(statement3 %(student,sheep))
```

```
Stephanie met with Volker today
Volker has 17 sheep in his office
Stephanie put 17 sheep in Volker's office
```

Problem 1

Elise gets into the lab early in the morning to count the number of colonies on her plates (Table 1).
Courtney comes in that afternoon and counts the same plates (Table 2). Calculate and print your answers for the following problems in the cell below.

a. Calculate the average for both Elise's and Courtney's results.
b. Calculate the percent difference.

Table 1

Plate	Colonies
1	200
2	232
3	201
4	195
5	225
6	217

Table 2

Plate	Colonies
1	193
2	238
3	195
4	199
5	229
6	213

Problem 1

Elise gets into the lab early in the morning to count the number of colonies on her plates (Table 1).
Courtney comes in that afternoon and counts the same plates (Table 2). Calculate and print your answers for the following problems in the cell below.

a. Calculate the average for both Elise's and Courtney's results.
b. Calculate the percent difference.

Table 1

Plate	Colonies
1	200
2	232
3	201
4	195
5	225
6	217

Table 2

Plate	Colonies
1	193
2	238
3	195
4	199
5	229
6	213

In [9]:

```
##Write a code below to answer the above question
#Be sure to comment out your code (explain what each section is doing in comments)

#part a
##calculate averages
EliseAvg = (200 + 232 + 201 + 195 + 225 + 217) / 6
CourtneyAvg = (193 + 238 + 195 + 199 + 229 + 213) / 6

#print answers
print('Elise Average:', EliseAvg)
print('Courtney Average:',CourtneyAvg)

#part b
##calculate percent error
PercDiff = (EliseAvg - CourtneyAvg)/EliseAvg * 100

print('Percent Difference:',PercDiff)

```

```
Elise Average: 211.66666666666666
Courtney Average: 211.16666666666666
Percent Difference: 0.23622047244094488
```

Problem 2

On the way to school Adrian and Charlie decide to sing *99 Bottles of Pop on the Wall*. Print the first ten verses without typing out 'bottles of pop on the wall' more than once in your code. **Hint:** use a "variable string" in your code.

Problem 2

On the way to school Adrian and Charlie decide to sing *99 Bottles of Pop on the Wall*. Print the first ten verses without typing out 'bottles of pop on the wall' more than once in your code. **Hint:** use a "variable string" in your code.

In [13]:

```
##Write a code below to answer the above question
#Be sure to comment out your code (explain what each section is doing in comments)

#set the verse to a string
verse = '%d bottles of pop on the wall'

#print out the verses
print(verse %99)
print(verse %98)
print(verse %97)
print(verse %96)
print(verse %95)
print(verse %94)
print(verse %93)
print(verse %92)
print(verse %91)
print(verse %90)

```

```
99 bottles of pop on the wall
98 bottles of pop on the wall
97 bottles of pop on the wall
96 bottles of pop on the wall
95 bottles of pop on the wall
94 bottles of pop on the wall
93 bottles of pop on the wall
92 bottles of pop on the wall
91 bottles of pop on the wall
90 bottles of pop on the wall
```

Making a Statement - Python Tutorial Part 2

By: Heather S. Deter

There are many statements used in programming to control the flow of the script. Here we introduce several key statements in Python.

First a note on whitespace

Python uses whitespace to denote which sections of code should be run under specific conditions and pieces of code.
Here we go through several statements that are indicated by a colon (:) followed by and indented block of text (**use tab to indent text for consistency**).
When checking for errors always check the colons and indentations.

Statement:
    A block of text with code and stuff

If statements

To check whether or not a condition is true before running the code we can use an if statement. Adding not before the statement checks for the opposite of the condition.

Examples

```
#if statements can be used with booleans
if True:
    print 'It is true'
if False:
    print 'It is false'
if not True:
    print 'It is false'

#if statements can also be used to test for a specific condition
mynumber = 5
if mynumber == 5:
    print 'My number is 5'
if not mynumber == 5:
    print 'My number is not 5'
```

When testing for conditions there are several comparison operators you can use.

Operator	Definition
==	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
!=	Not equal to

Example problem
On Tuesdays and Thursdays Bikram has to go to class at 9 am. Write a statement below to tell Bikram to go to class if it is a Tuesday or Thursday.

In [ ]:

```
#Statement: It is a Tuesday or Thursday.
##Write True/False to define the statement below.
statement =

#write an if statement to tell Bikram to go to class
```

Example problem
On Tuesdays and Thursdays Bikram has to go to class at 9 am. Write a statement below to tell Bikram to go to class if it is a Tuesday or Thursday.

In [1]:
```
#Statement: It is a Tuesday or Thursday.
##Write True/False to define the statement below.
statement = True

#write an if statement to tell Bikram to go to class
if statement:
    print('Bikram, go to class')
```

```
Bikram, go to class
```

If, else statements

If, else statements build on the if statement. If the condition is not fulfilled, the code executes the else statement.

```
#improving on the earlier example
mynumber = 5
if mynumber == 5:
    print 'My number is 5'
else:
    print 'My number is not 5'

#there is also the elif (else, if) statement
if mynumber == 5:
    print 'My number is 5'
elif mynumber == 4:
    print 'My number is 4
else:
    print 'My number is not 4 or 5'
```

Example problem: Part 2

On Mondays, Wednesdays and Fridays, Bikram goes to the gym. Write and if, else statement to tell Bikram to go to class if it is Tuesday or Thursday, or to go to the gym if it is Monday, Wednesday or Friday.

In [ ]:

```
#Statement: It is a Tuesday or Thursday. Write True/False to define the statement below.
statement =

#write an if/else statement to tell Bikram to go to class if it is Tuesday or Thursday and if not (else) to the gym
```

Example problem: Part 2

On Mondays, Wednesdays and Fridays, Bikram goes to the gym. Write and if, else statement to tell Bikram to go to class if it is Tuesday or Thursday, or to go to the gym if it is Monday, Wednesday or Friday.

In [4]:

```
#Statement: It is a Tuesday or Thursday.
##Write True/False to define the statement below.
statement = False

#write an if statement to tell Bikram to go to class
if statement:
    print('Bikram, go to class')
```

And, or, not

In conditional statements you can use the terms 'and', 'or' and 'not' for more complex conditions.

Examples

```
#cond1: Nicholas is in the lab.
#cond2: Students are in the lab.

if cond1 and cond2:
    print 'Nicholas stays in the lab.'

if cond1 or cond2:
    print 'Someone is in the lab.'

if cond1 and not cond2:
    print 'Nicholas goes back to his office.'
```

While loops

Sometimes you may want to repeat a section of code until a certain condition is met. While loops test for a condition and then iterate (repeat) over the code until the condition is met.

In [5]:
```
#run this cell to see a while loop in action
#count up until a specific number
number = 1

while number < 10:
    number = number + 1

print(number)
```

```
10
```

Notice how we print the number 10. What happens if we use the <= symbol? Test your answer by editing the code above.

And, or, not

In conditional statements you can use the terms 'and', 'or' and 'not' for more complex conditions.

Examples

```
#cond1: Nicholas is in the lab.
#cond2: Students are in the lab.

if cond1 and cond2:
    print 'Nicholas stays in the lab.'

if cond1 or cond2:
    print 'Someone is in the lab.'

if cond1 and not cond2:
    print 'Nicholas goes back to his office.'
```

While loops

Sometimes you may want to repeat a section of code until a certain condition is met. While loops test for a condition and then iterate (repeat) over the code until the condition is met.

In [6]:
```
#run this cell to see a while loop in action
#count up until a specific number
number = 1

while number <= 10:
    number = number + 1

print(number)
```

```
11
```

Notice how we print the number 10. What happens if we use the <= symbol? Test your answer by editing the code above.

Example problem
During advent, Hannah and Mackayla get 2 pieces of chocolate every morning until Christmas. Use a while loop to add up all the candy they get until the 25th.

In [ ]:

```
#variable for the date
DecDate = 1
#the amount of candy Hannah and Mackayla have eaten
candy = 0

#fill in the while loop below to increase the date each day and add up the amount of candy the girls are getting
while :


    #don't forget to increase DecDate every iteration


print(candy)



```

Be careful with while loops, because they can run infinitely if the condition is never met.

To stop a code from running in Jupyter go to Kernel -> Interrupt

Example problem
During advent, Hannah and Mackayla get 2 pieces of chocolate every morning until Christmas. Use a while loop to add up all the candy they get until the 25th.

In [7]:
```
#variable for the date
DecDate = 1
#the amount of candy Hannah and Mackayla have eaten
candy = 0

#fill in the while loop below to increase the date each day and add up the amount of candy the girls are getting
while DecDate <=25:
    candy+=1
    DecDate+=1


    #don't forget to increase DecDate every iteration


print(candy)
```

```
25
```

Be careful with while loops, because they can run infinitely if the condition is never met.

To stop a code from running in Jupyter go to Kernel -> Interrupt

For loops

Another (often preferable) method of iterating over a block of code is using a for loop. The advantage of the for loop is that **the code repeats a finite number of times**, which lowers the risk of stalling the code or using too much memory.

There are several methods to assign the number of loops in a for loop.
1) The simplest method is to use the range function. range(y) creates a list of numbers starting at 0 and ending at *y-1*.
2) Alternatively range(x,y) creates a series of numbers starting at *x* and ending at *y-1* and range(x,y,z) creates a list of numbers starting at *x*, ending at *y-1*, and increasing by ***z***

In [ ]:
```
##run this cell to see the output
print('range(5)')

for i in range(5):
    print(i, end=' ')

print('\nNote how the last number is the first digit is zero and the last digit is the range minus one.\n')

print('range(1,5)')

for i in range(1,5):
    print(i, end=' ')

print('\n\nrange(1,6,2)')

for i in range(1,6,2):
    print(i, end=' ')

print('\nNote how the step (2) is added starting at the first number (1)')
```

3) Another method is to iterate over an object with discrete units, like a string or a list (which will be covered later).

In [8]:
```
#run this cell to iterate over a string
mystring = 'abcdefghijklmnopqrstuvwxyz'

for i in mystring:
    print(i, end=' ')
```

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
```

Try, except

The try/except statement is meant for catching and preventing errors. If an error occurs during the block of code under try, the code will continue to the except block instead of giving an error. The except block can be used to print your own statement related to the error or it can be used to skip to the next block of code or break a loop. *This statement is not to be used lightly.*

See an example below

In [9]:
```
#run this cell to see the error
for i in range(20):
    print(1.0/i)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-9-e7e9da001896> in <module>
      1 #run this cell to see the error
      2 for i in range(20):
----> 3     print(1.0/i)

ZeroDivisionError: float division by zero
```

In [10]:
```
#run this cell to see how the try statement works
for i in range(7):
    try:
        print(1.0/i)

    except:
        #continue causes the program to go to the next iteration
        continue
```

```
1.0
0.5
0.3333333333333333
0.25
0.2
0.16666666666666666
```

You can also use 'break' and 'pass' which stop the loop and ignore the error respectively.

https://docs.python.org/3/reference/compound_stmts.html#try

Practice Problems

Problem 1

Write a command to divide one number by another. Use an if statement to prevent dividing that number by 0. Be sure to test several different numerators and denominators, and note the accuracy of your answers.

In [ ]:

```
##Write a code below to answer the above question
#Be sure to comment out your code (explain what each section is doing in comments)
numerator = 1
denominator = 0
```

Problem 1

Write a command to divide one number by another. Use an if statement to prevent dividing that number by 0. Be sure to test several different numerators and denominators, and note the accuracy of your answers.

In [12]:

```
##Write a code below to answer the above question
#Be sure to comment out your code (explain what each section is doing in comments)
numerator = 1
denominator = 0

if not denominator == 0:
    answer = numerator/denominator
else:
    answer = 'cannot divide by zero'

print(answer)
```

```
cannot divide by zero
```

In [13]:

```
##Write a code below to answer the above question
#Be sure to comment out your code (explain what each section is doing in comments)
numerator = 1
denominator = 20

if not denominator == 0:
    answer = numerator/denominator
else:
    answer = 'cannot divide by zero'

print(answer)
```

```
0.05
```

Problem 2

Alawiah has two baskets of apples. She never wants Basket 1 to have two or more apples than Basket 2. Write a code that will "add" or "remove" apples from the baskets until Basket 1 has no more than one extra apple compared to Basket 2. For example if Basket 2 has three apples, Basket 1 cannot have more than four apples. Print the starting and ending number of apples for each basket.

In [17]:

```
##Write a code below to answer the above question
#Be sure to comment out your code (explain what each section is doing in comments)

#use these variables to test different numbers
basket1 = 20
basket2 = 5
```

Problem 2

Alawiah has two baskets of apples. She never wants Basket 1 to have two or more apples than Basket 2. Write a code that will "add" or "remove" apples from the baskets until Basket 1 has no more than one extra apple compared to Basket 2. For example if Basket 2 has three apples, Basket 1 cannot have more than four apples. Print the starting and ending number of apples for each basket.

In [16]:

```
##Write a code below to answer the above question
#Be sure to comment out your code (explain what each section is doing in comments)

#use these variables to test different numbers
basket1 = 20
basket2 = 5

while basket1 > basket2+1:
    basket1-=1
    basket2+=1

print('Basket 1:', basket1)
print('Basket 2:',basket2)
```

```
Basket 1: 13
Basket 2: 12
```

Problem 3

On the way to school Adrian and Charlie decide to sing 99 Bottles of Pop on the Wall. Print the last ten verses without typing out 'bottles of pop on the wall' more than once in your code. **Hint**: when using range() use a negative number to reverse the range. E.g. range(10,0,-1).

In [ ]:

```
##Write a code below to answer the above question
#Be sure to comment out your code (explain what each section is doing in comments)
```

Problem 3

On the way to school Adrian and Charlie decide to sing 99 Bottles of Pop on the Wall. Print the last ten verses without typing out 'bottles of pop on the wall' more than once in your code. **Hint**: when using range() use a negative number to reverse the range. E.g. range(10,0,-1).

In [20]:

```
##Write a code below to answer the above question
#Be sure to comment out your code (explain what each section is doing in comments)

verse = '%d bottle of pop on the wall'

for v in range(99,89,-1):
    print(verse %v)

```

```
99 bottle of pop on the wall
98 bottle of pop on the wall
97 bottle of pop on the wall
96 bottle of pop on the wall
95 bottle of pop on the wall
94 bottle of pop on the wall
93 bottle of pop on the wall
92 bottle of pop on the wall
91 bottle of pop on the wall
90 bottle of pop on the wall
```

Let's Get Iterable - Python Tutorial Part 3

By: Heather S. Deter

Objects that are iterable have discrete units that can be counted or looped through using a for loop.

Lists

A list is exactly what it sounds like, a list of objects. Lists are initiated by square brackets [].

```
#we can initiate an empty list using []
mylist = []

#or we can put stuff in our list
mylist2 = [1,3,4,7,9]
```

In [1]:
```
#run this cell to see lists in action

#define two lists
mylist = ['bananas']
mylist2 = [3,7,10,19,31]

#append the lists to each other
mylist3 = mylist + mylist2

print('mylist:', mylist)
print('mylist2:', mylist2)
print('mylist3:', mylist3)
```

```
mylist: ['bananas']
mylist2: [3, 7, 10, 19, 31]
mylist3: ['bananas', 3, 7, 10, 19, 31]
```

Take note of how the list we printed above is shown in square brackets [].
There are a variety of useful features when working with lists.

In [2]:

```
#we can add a number to our list using append
mylist.append(10)

#we can combine list using +
mylists = mylist + mylist2

print('mylist:',mylist)
print('mylists:',mylists)
```

```
mylist: ['bananas', 10]
mylists: ['bananas', 10, 3, 7, 10, 19, 31]
```

In [3]:

```
#You can also do things with lists like sorting them or get the length of the list.
sortlist2 = sorted(mylist2)
mylist2len = len(mylist2)

print('Sorted:',sortlist2)
print('Length:', mylist2len)
```

```
Sorted: [3, 7, 10, 19, 31]
Length: 5
```

For more information on what you can do with lists see: https://docs.python.org/2/tutorial/datastructures.html

---

Indexing lists

Similarly to strings, lists have indeces and can be measured by the number of objects they contain.
To get a particularly index use square brackets [ ] to slice the list.

In [4]:

```
#run this cell to see indexing in action

#to get the length of a list use len()
length = len(mylist3)

#get first object (remember to start counting at 0)
first = mylist3[0]

#get last object (use - to start at the end, but when counting from the end start at 1)
last = mylist3[-1]

#get a range of objects (remember the end of the range will stop at 1 less the given number)
middle = mylist3[1:-1]

print(mylist3)
print('Length:', length)
print('First:',first)
print('Last:',last)
print('Middle:',middle)
```

```
['bananas', 3, 7, 10, 19, 31]
Length: 6
First: bananas
Last: 31
Middle: [3, 7, 10, 19]
```

Example

Make a list of every member in your family in birth order. Print the names of the oldest, youngest and a person in the middle using your list.
Hint: use len() to get the length of your list and help you find the middle position. To convert a float to an integer use int().

In [ ]:

```
1 #write your own code and run the cell
2
3
4
```

Example

Make a list of every member in your family in birth order. Print the names of the oldest, youngest and a person in the middle using your list.
Hint: use len() to get the length of your list and help you find the middle position. To convert a float to an integer use int().

In [7]:
```
#write your own code and run the cell

fam = ['Dad','Mom','me','sister']

#index the list based on an integer of half the length of your list
##if you have an odd number of family members you don't have to convert to an integer
print(fam[int(len(fam)/2)])

```

```
me
```

Dictionaries

- Rather than indexing by position (as in strings and lists), dictionaries index objects by *keys*.
  Curly brackets { } are used to initiate and empty dictionary. Then keys are defined before a colon followed by the value and seperated by commas.
  E.g. key:value
- When iterating over a dictionary, the code will actually iterate over the keys of the dictionary.
  To iterate over both the keys and the values (keys,values) use mydict.items().
- To call a particular value you can use the key in square brackets.
  E.g. mydict[key]

In [8]:

```
##run this cell to see dictionaries in action
#an empty dictionary
mydict = {}

#fruit colors
fruits = {'apple':'red','orange':'orange','lemon':'yellow','pear':'green','banana':'yellow'}

print('mydict:', mydict)
print('fruits:',fruits)
print('fruits[\'apple\']:', fruits['apple'])
print('\nYou can also make a list of keys in a dictionary using list(dict.keys()).')
print('fruits keys:', list(fruits.keys()))
```

```
mydict: {}
fruits: {'apple': 'red', 'orange': 'orange', 'lemon': 'yellow', 'pear': 'green', 'banana': 'yellow'}
fruits['apple']: red

You can also make a list of keys in a dictionary using list(dict.keys()).
fruits keys: ['apple', 'orange', 'lemon', 'pear', 'banana']
```

Example

Write your own dictionary for the table below. Is it better to use names or grades as the keys? Why?

Class Grades	
Student	**Grade**
Greg	95
Sarah	95
Colleen	87
Lenard	84
Lydia	91
Johnny	89

In [ ]:

```
#Write a dictionary for the class grades.
```

Example

Write your own dictionary for the table below. Is it better to use names or grades as the keys? Why?

Class Grades	
Student	**Grade**
Greg	95
Sarah	95
Colleen	87
Lenard	84
Lydia	91
Johnny	89

In [10]:
```
#Write a dictionary for the class grades.

mydict = {'Greg':95,'Sarah':95,'Colleen':87,'Lenard':84,'Lydia':91,'Johnny':89}

print(mydict)



```

```
{'Greg': 95, 'Sarah': 95, 'Colleen': 87, 'Lenard': 84, 'Lydia': 91, 'Johnny': 89}
```

Keys have to be unique (you can't have two of the same key), therefore in the above example using names is better because two students have the same grade.
For more on dictionaries see: https://docs.python.org/3/tutorial/datastructures.html#dictionaries

Practice problems

Problem 1

Hint: use some functions from https://docs.python.org/2/tutorial/datastructures.html.

a) You are hosting a party and will be drawing prizes. In the first hour the following people walk in the door: Jonathan, Karly, Sharon, Ted, Amrit, Ben, Vanessa, Ugochi, and Sindhura. Make a list of all the people who are at the party. Draw prizes for the 5th person to show up, the 4th person alphabetically and the second to last person to show up. Print out the winners names and state why they won.

b) A few more people walk in the door: Cameron, Greg, Rachel, Renato and Tina. However, it is getting pretty lat so Ben, Karly and Sharon leave. Adjust your original list to account for the people coming and going. Draw another round of prizes for the people still in the room: the 7th person to show up, the 5th person alphabetically and the 7th from the last person to show up.

In [ ]:

```
##Write a code below to answer the above question
#Be sure to comment out your code (explain what each section is doing in comments)
```

Practice problems

Problem 1

Hint: use some functions from https://docs.python.org/2/tutorial/datastructures.html.

The list data type has some more methods. Here are all of the methods of list objects:

`list.`**`append`**(*x*)
Add an item to the end of the list; equivalent to `a[len(a):] = [x]`.

`list.`**`extend`**(*L*)
Extend the list by appending all the items in the given list; equivalent to `a[len(a):] = L`.

`list.`**`insert`**(*i*, *x*)
Insert an item at a given position. The first argument is the index of the element before which to insert, so `a.insert(0, x)` inserts at the front of the list, and `a.insert(len(a), x)` is equivalent to `a.append(x)`.

`list.`**`remove`**(*x*)
Remove the first item from the list whose value is *x*. It is an error if there is no such item.

`list.`**`pop`**([*i*])
Remove the item at the given position in the list, and return it. If no index is specified, `a.pop()` removes and returns the last item in the list. (The square brackets around the *i* in the method signature denote that the parameter is optional, not that you should type square brackets at that position. You will see this notation frequently in the Python Library Reference.)

`list.`**`index`**(*x*)
Return the index in the list of the first item whose value is *x*. It is an error if there is no such item.

`list.`**`count`**(*x*)
Return the number of times *x* appears in the list.

`list.`**`sort`**(*cmp=None*, *key=None*, *reverse=False*)
Sort the items of the list in place (the arguments can be used for sort customization, see `sorted()` for their explanation).

`list.`**`reverse`**()
Reverse the elements of the list, in place.

Problem 1

Hint: use some functions from https://docs.python.org/2/tutorial/datastructures.html.

a) You are hosting a party and will be drawing prizes. In the first hour the following people walk in the door: Jonathan, Karly, Sharon, Ted, Amrit, Ben, Vanessa, Ugochi, and Sindhura. Make a list of all the people who are at the party. Draw prizes for the 5th person to show up, the 4th person alphabetically and the second to last person to show up. Print out the winners names and state why they won.

b) A few more people walk in the door: Cameron, Greg, Rachel, Renato and Tina. However, it is getting pretty lat so Ben, Karly and Sharon leave. Adjust your original list to account for the people coming and going. Draw another round of prizes for the people still in the room: the 7th person to show up, the 5th person alphabetically and the 7th from the last person to show up.

In [15]:
```
##Write a code below to answer the above question
#Be sure to comment out your code (explain what each section is doing in comments)

guests = ['Jonathan','Karly','Sharon','Ted','Amrit','Ben','Vanessa','Ugochi','Sindhura']

print('5th guest:',guests[4])
print('4th alphabetically:',sorted(guests)[3])
print('2nd to last:',guests[-2])

guests.extend(['Cameron','Greg','Rachel','Renato','Tina'])

for name in ['Ben','Karly','Sharon']:
    guests.remove(name)


print('7th guest:',guests[6])
print('5th alphabetically:',sorted(guests)[4])
print('7th to last:',guests[-7])



```

```
5th guest: Amrit
4th alphabetically: Karly
2nd to last: Ugochi
7th guest: Cameron
5th alphabetically: Rachel
7th to last: Ugochi
```

Problem 2

It's poker night! It's your job to keep track of how many chips everyone has as you control the cash for the night. Make a dictionary of the starting chip counts for everybody from the table below. Then keep track as the night goes on.

Name	Chips
Rachel	50
Scott	75
Kiana	60
Jeremiah	50
Haley	80
Job	45

1) Haley wins 10 chips the first hand and everyone else looses two chips.
2) Scott wins 5 chips from Jeremiah in the second hand.
3) Job and Jeremiah each loose 15 chips to Rachel in the third hand.
4) Jeremiah wins 20 chips from everyone in the fourth hand (4 chips each).
5) Job looses all of his remaining chips to Kiana in the fifth hand.

Print how many chips each person currently has left.

In [ ]:

```
##Write a code below to answer the above question
#Be sure to comment out your code (explain what each section is doing in comments)
```

In [11]:

```python
##Write a code below to answer the above question
#Be sure to comment out your code (explain what each section is doing in comments)
#make dictionary
poker = {'Rachel':50,'Scott':75,'Kiana':60,'Jeremiah':50,'Haley':80,'Job':45}

#use for loop for #1
for key in poker:
    if key != 'Haley':
        poker[key] = poker[key] - 2
    else:
        poker[key] = poker[key] + 10

#2
poker['Scott'] = poker['Scott'] + 5
poker['Jeremiah'] = poker['Jeremiah'] - 5

#3
poker['Job'] = poker['Job'] - 15
poker['Jeremiah'] = poker['Jeremiah'] - 15
poker['Rachel'] = poker['Rachel'] + 30

#4
for key in poker:
    if key != 'Jeremiah':
        poker[key] = poker[key] - 4
        poker['Jeremiah'] = poker['Jeremiah'] + 4

#5
poker['Kiana'] = poker['Kiana'] + poker['Job']
poker['Job'] = 0

#print answers
print(poker)

```

```
{'Rachel': 74, 'Scott': 74, 'Kiana': 78, 'Jeremiah': 48, 'Haley': 86, 'Job': 0}
```

How to function - Python Tutorial Part 4

By: Heather S. Deter

A function is a block of reusable code that you can call for a specific task. In fact, you already know how to use functions! print() is a function. You can also make your own functions by defining them in a block of code.

```
def MyFunction(input):
    return(output)
```

In [1]:
```
#let's define a function to calculate the percent error
##run this cell to store the function
def getPercError(calculated,expected):

    #calculate the difference
    diff = calculated - expected

    #make sure number is positive
    if diff < 1:
        diff = diff*-1

    percerror = diff/expected * 100

    return percerror
```

In [2]:
```
#run this cell to test our function
calculated = 90
expected = 100

percerror = getPercError(calculated,expected)
print(percerror, '%')
```

```
10.0 %
```

Functions can return muliple values simply by placing a comma between the values you are returning. For example, if we wanted to also return the difference in our function above we could change it as follows.

Functions can return muliple values simply by placing a comma between the values you are returning. For example, if we wanted to also return the difference in our function above we could change it as follows.

In [3]:
```
##run this cell to store the function
def getDiffError(calculated,expected):

    #calculate the difference
    diff = calculated - expected

    #make sure number is positive
    if diff < 1:
        diff = diff*-1

    percerror = diff/expected * 100

    #added diff to be returned
    return diff, percerror
```

In [4]:
```
#run this cell to test our function
calculated = 180
expected = 200

#we can still only assign the output to one variable
answer = getDiffError(calculated,expected)
print(answer)

#or we can give the function two variables to call
diff, percerror = getDiffError(calculated,expected)
print(diff, 'difference and ', percerror, '% error')
```

```
(20, 10.0)
20 difference and  10.0 % error
```

Importing libraries

A library, also known as a package, is a group of built-in modules that provide functions for common (and sometimes uncommon) use. They may be written in C to provide access to system functionality that would otherwise be inaccessible or in Python to provide standardized solutions for common problems. For more in-depth information see https://realpython.com/python-modules-packages/

To use a library, it must first be imported so that the program you are working on can access it. Some libraries have to be installed, but Anaconda provides many libraries in it's initial instillation, a few of which we will be using here.

In [5]:
```
##run this cell
#if the library is installed in your system you can import it by calling the name
import math

#you can designate a shortened name to call a specific library by using "as"
import numpy as np

#you can import a section of a library (rather than the whole thing) using "from"
from scipy import stats


print('Yay, you imported things')
```

```
Yay, you imported things
```

Math

A lot of mathmatical calculations are difficult to write out in python by hand since the operators are somewhat limited. Math takes care of that problem by providing several functions that are commonly used for math. See this link for details. https://docs.python.org/3.7/library/math.html

In [6]:

```
##run this cell

#you can get the square root of a number
print(math.sqrt(4))

#find the logarithm of a number
print(math.log(10))

#use constants like pi
print(math.pi)
```

```
2.0
2.302585092994046
3.141592653589793
```

Practice

Solve the follwing using math:
1) Find r if $r = \sqrt{29}$
2) Find y if $y = \pi r^2$
3) a = 8!
   b = 9!
   Find c if $a_2 + b_2 = c_2$

Hint: check out the documentation if you need help finding a specific function https://docs.python.org/3/library/math.html

In [ ]:

```
##solve the practice problem above in this cell

r =

y =

c =



print('1:',r)
print('2:',y)
print('3:',c)

```

`math.`**`factorial`**`(x)`

Return *x* factorial. Raises `ValueError` if *x* is not integral or is negative.

`math.`**`pi`**

The mathematical constant π = 3.141592…, to available precision.

`math.`**`sqrt`**`(x)`

Return the square root of *x*.

Practice

Solve the follwing using math:

1) Find r if r = $\sqrt{29}$
2) Find y if y = πr^2
3) a = 8!
   b = 9!
   Find c if $a_2 + b_2 = c_2$

Hint: check out the documentation if you need help finding a specific function https://docs.python.org/3/library/math.html

In [ ]:

```
##solve the practice problem above in this cell

r =

y =

c =



print('1:',r)
print('2:',y)
print('3:',c)

```

Practice

Solve the follwing using math:
1) Find r if $r = \sqrt{29}$
2) Find y if $y = \pi r^2$
3) $a = 8!$
   $b = 9!$
   Find c if $a_2 + b_2 = c_2$

Hint: check out the documentation if you need help finding a specific function https://docs.python.org/3/library/math.html

In [11]:
```
##solve the practice problem above in this cell

r = math.sqrt(29)

y = math.pi*r**2

c = (math.sqrt(math.factorial(8)**2 + math.factorial(9)**2))



print('1:',r)
print('2:',y)
print('3:',c)

```

```
1: 5.385164807134504
2: 91.10618695410399
3: 365113.12876970065
```

Numpy

If you thought Excel was your best friend, meet Numpy - your new best friend. Everything you could want in a spreadsheet with infinite dimensions and the power of Python. It also includes algebraic functions and other cool features for most of your number crunching needs. Check out the website for all the details http://www.numpy.org/. And see their quickstart tutorial if you have questions https://docs.scipy.org/doc/numpy/user/quickstart.html.

Making Arrays

You can think of an array as a spreadsheet, the only difference is that arrays can have n-dimensions. For now we'll just work with two dimensions.
You can make a numpy array out of a series of lists or import a csv file that you already have ready to go. We'll go over importing csv files later, for now we'll start with lists.

Our class from part 3 has gotten their final grades. Let's do some math using numpy to calculate the class grades.

Class Grades

Student	Midterm	Final
Greg	95	94
Sarah	95	97
Colleen	87	95
Lenard	84	88
Lydia	91	93
Johnny	89	85

Numpy

If you thought Excel was your best friend, meet Numpy - your new best friend. Everything you could want in a spreadsheet with infinite dimensions and the power of Python. It also includes algebraic functions and other cool features for most of your number crunching needs. Check out the website for all the details http://www.numpy.org/. And see their quickstart tutorial if you have questions https://docs.scipy.org/doc/numpy/user/quickstart.html.

Making Arrays

You can think of an array as a spreadsheet, the only difference is that arrays can have n-dimensions. For now we'll just work with two dimensions.
You can make a numpy array out of a series of lists or import a csv file that you already have ready to go. We'll go over importing csv files later, for now we'll start with lists.

Our class from part 3 has gotten their final grades. Let's do some math using numpy to calculate the class grades.

Class Grades

Student	Midterm	Final
Greg	95	94
Sarah	95	97
Colleen	87	95
Lenard	84	88
Lydia	91	93
Johnny	89	85

In [13]:

```
#make an empty array
empty = np.empty((1,3),dtype=int)
ones = np.ones((3,2))
zeros = np.zeros((2,3))

print('Empty\n', empty)
print('Ones\n', ones)
print('Zeros\n', zeros)

#you can also fill an array with a specified value
empty.fill(12)

print('Filled\n', empty)
```

```
Empty
 [[459619792       551         0]]
Ones
 [[1. 1.]
 [1. 1.]
 [1. 1.]]
Zeros
 [[0. 0. 0.]
 [0. 0. 0.]]
Filled
 [[12 12 12]]
```

Note that empty doesn't really mean empty, it just consists of random numbers and should be used with caution.

Working with Arrays

There are several key features of arrays to know when working with arrays. The most important thing is knowing the shape and type of your array.

In [14]:
```
# find the shape of our array using array.shape
print('classgrades.shape',classgrades.shape)
print('Each number corresponds to a dimension (row, columns, etc)\n')

#find the type of array (dtype) using array.dtype
print('classgrades.dtype',classgrades.dtype)
print('<U7 stands for a unicode string with 7 characters. An array must have the same dtype throughout.')
```

```
classgrades.shape (3, 6)
Each number corresponds to a dimension (row, columns, etc)

classgrades.dtype <U7
<U7 stands for a unicode string with 7 characters. An array must have the same dtype throughout.
```

When working with 2D arrays it is very important to keep in mind that the **first axis (0) corresponds to rows** and the **second axis (1) corresponds to columns.**

Since numpy arrays must be one type and our array has strings all of the values are considered strings.

Converting arrays

Notice how the dytpe of grades is unicode string even though they are all numbers. This is because arrays have a single dtype and the names were all strings. To convert an array use the command array.astype(dtype).

Now we can convert the array to a type that we can work with, floats. To do so we can do the following.

In [16]:
```
1 grades = grades.astype(float)
2
3 print(grades)
4 print(grades.shape)
5 print(grades.dtype)
```

```
[[95. 95. 87. 84. 91. 89.]
 [94. 97. 95. 88. 93. 85.]]
(2, 6)
float64
```

Using numpy for calculations

There are several useful features for common calculations.

In [17]:
```
#Calculate averages
MidtermAvg = np.mean(grades[0,:])
FinalAvg = np.mean(grades[1,:])

print('Averages\nMidterm:', MidtermAvg, '\nFinal:', FinalAvg)

#calculate standard deviation
MidtermStd = np.std(grades[0,:])
FinalStd = np.std(grades[1,:])

print('\nStandard deviations\nMidterm:', MidtermStd, '\nFinal:', FinalStd)

#calculate minimum
MidtermMin = np.min(grades[0,:])
FinalMin = np.min(grades[1,:])

print('\nMinimum\nMidterm:', MidtermMin, '\nFinal:', FinalMin)

#calculate minimum
MidtermMax = np.max(grades[0,:])
FinalMax = np.max(grades[1,:])

print('\nMaximum\nMidterm:', MidtermMax, '\nFinal:', FinalMax)
```

```
Averages
Midterm: 90.16666666666667
Final: 92.0

Standard deviations
Midterm: 4.017323597731316
Final: 4.163331998932265

Minimum
Midterm: 84.0
Final: 85.0

Maximum
Midterm: 95.0
Final: 97.0
```

In [18]:

```
#most formulas also allow you to apply them to a specific axis
Avgs = np.mean(grades,axis = 1)
print('Midterm, Final\n',Avgs)
```

```
Midterm, Final
 [90.16666667 92.        ]
```

In [19]:

```
#you can also do calculations elementwise (each cell is multiplied by the corresponding cell in the matrix)

#subtraction
Gradediff = grades[1,:] - grades[0,:]

print('Differences', Gradediff.astype(int))

#addition
Gradeproduct = grades[1,:] + grades[0,:]

#and divide by a consant
print('Averages', Gradeproduct/2)
```

```
Differences [-1  2  8  4  2 -4]
Averages [94.5 96.  91.  86.  92.  87. ]
```

Practice Problems

Problem 1

Write a function to calculate the average and SEM from a list. Use that function to solve the following problem.

How to calculate SEM: https://en.wikipedia.org/wiki/Standard_error#Population

In [ ]:

```
#define your function
##don't forget the run the cell afterwards to store the function

def myfunc(mylist):
    #calculate average


    #calculate SEM


    #return answers


```

Practice Problems

Problem 1

Write a function to calculate the average and SEM from a list. Use that function to solve the following problem.
How to calculate SEM: https://en.wikipedia.org/wiki/Standard_error#Population

In [22]:

```
#define your function
##don't forget the run the cell afterwards to store the function

def myfunc(mylist):
    #calculate average
    avg = np.mean(mylist)

    #calculate SEM
    SEM = np.std(mylist)/math.sqrt(len(mylist))

    #return answers
    return(avg,SEM)


```

Elise gets into the lab early in the morning to count the number of colonies on her plates (Table 1).
Courtney comes in that afternoon and counts the same plates (Table 2). Calculate and print your answers for the following problems in the cell below.

a. Calculate the average and SEM for both Elise's and Courtney's results. **Hint:** you can use the function we defined earlier.
b. Calculate the percent error.

Table 1	
Plate	**Colonies**
1	200
2	232
3	201
4	195
5	225
6	217

Table 2	
Plate	**Colonies**
1	193
2	238
3	195
4	199
5	229
6	213

In [ ]:

```
##Write a code below to answer the above question
#Be sure to comment out your code (explain what each section is doing in comments)
```

In [23]:

```
##Write a code below to answer the above question
#Be sure to comment out your code (explain what each section is doing in comments)

#make lists for both tables
Elise = [200,232,201,195,225,217]
Courtney = [193,238,195,199,229,213]

#part a
Eavg,ESEM = myfunc(Elise)
Cavg,CSEM = myfunc(Courtney)

#print the answers using \n for new lines
print('Elise\nAvg:',Eavg,'\nSEM:',ESEM)
print('\nCourtney\nAvg:',Cavg,'\nSEM:',CSEM)

#part b
PercError = (Eavg - Cavg)/Eavg

print('\nPercent Error:',PercError,'%')
```

```
Elise
Avg: 211.66666666666666
SEM: 5.645384873537887

Courtney
Avg: 211.16666666666666
SEM: 7.033320168497421

Percent Error: 0.002362204724409449 %
```

Problem 2

Initalize a new numpy array then assign it values according to the table below. Then calculate each exam average, each student's average, and the overall average.

Exam scores	Louisa	Kurt	Elsie
Exam 1	92	87	94
Exam 2	89	91	92
Exam 3	93	90	97

In [ ]:

```
#initialize a new array and assign the values



#calculate the overall average


#calculate the exam average (Hint: calculate along one axis)


#calculate the student averages (Hint: calculate along one axis)


#print out your answers


```

Problem 2

Initalize a new numpy array then assign it values according to the table below. Then calculate each exam average, each student's average, and the overall average.

Exam scores	Louisa	Kurt	Elsie
Exam 1	92	87	94
Exam 2	89	91	92
Exam 3	93	90	97

In [26]:

```
#initialize a new array and assign the values
exams = [[92,87,94],[89,91,92],[93,90,97]]

#calculate the overall average
Avg = np.mean(exams)
print('Overall avg:',Avg)

#calculate the exam average (Hint: calculate along one axis)
Examavgs = np.mean(exams,axis = 1)
print('Exam avg:',Examavgs)

#calculate the student averages (Hint: calculate along one axis)
Studentavgs = np.mean(exams,axis = 0)
print('Student avg:',Studentavgs)


```

```
Overall avg: 91.66666666666667
Exam avg: [91.         90.66666667 93.33333333]
Student avg: [91.33333333 89.33333333 94.33333333]
```

Statistics - Python Tutorial Part 5

By: Heather S. Deter

Python is a good tool for running statistics on a lot of data, and if designed properly, you can write a script to run on multiple datasets.

There are a lot of different options for how to run statistics. What's import to learn here is not so much the precise methodology (you can Google which function to use), but more about how to approach statistical analysis from a programming perspective.

Importing data

The first thing to do when you need to run statistics is to import your data. Usually in the form of a comma seperated values (csv). Numpy http://www.numpy.org/ and pandas http://pandas.pydata.org/ are two libraries that are key parts of importing csv files and handling data.

In [1]:
```
#run this cell

import numpy as np
import pandas as pd
```

We are going to use some data from a 2015 study on seedlings in Fish Creek. It is important to know the following when you need to import (or export) a file.
1) What is your working directory? Your working directory is the directory your Jupyter Home tab is running from.
2) Where is your file relative to your working directory?
If you are confused about directories check out this explanation: https://www.computerhope.com/jargon/d/director.htm

While there are a number of different ways to import data, one way is to use numpy.
https://docs.scipy.org/doc/numpy/reference/generated/numpy.genfromtxt.html
Another method is to use pandas, which is what we will be using.
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html

In [2]:

```
#import a csv file

#set to the path of your CSV file relative to the working directory
CSVname = 'C:/Users/owner/Documents/research/scripts/learn-python/FishCreekseedlings2015.csv'

#import the CSV file
##the delimeter indicates what seperates your values (commas)
##skip_header allows us to only import the data - the headers are string but we want our data to be float
DATA = pd.read_csv(CSVname)

#print the numpy array
print(DATA)
```

```
    Ecotype  Max Height (cm)
0     Local             21.0
1     Local             27.0
2     Local             27.5
3     Local             28.0
4     Local             30.5
5     Local             33.0
6     Local             37.0
7     Local             37.0
8     Local             39.0
9     Local             40.0
10    Local             41.0
11    Local             42.0
12    Local             44.0
13    Local             44.0
14    Local             44.5
15    Local             45.0
16    Local             47.0
17    Local             47.5
18    Local             49.0
19    Local             49.0
20    Local             49.5
```

T-tests

Now we can run a t-test using scipy, but first we have to import scipy.

In [4]:
```
#run this cell

from scipy import stats
```

We are going to run a Student's t-test. https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html

In [5]:
```
#seperate out treated and untreated data
Southern = DATA[DATA['Ecotype']=='Southern']
Local = DATA[DATA['Ecotype']=='Local']

#run the t-test on the CFU/ml column
ttest = stats.ttest_ind(Local['Max Height (cm)'],Southern['Max Height (cm)'])
print(ttest)
```

```
Ttest_indResult(statistic=4.08490189729399, pvalue=5.518267558644035e-05)
```

One-way ANOVA

One way to preform a oneway anova is to use scipy f_oneway. https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.f_oneway.html

In [6]:
```
#run a oneway anova
statsanova = stats.f_oneway(Local['Max Height (cm)'],Southern['Max Height (cm)'])
print(statsanova)
```

```
F_onewayResult(statistic=16.68642351051606, pvalue=5.5182675586439454e-05)
```

Another method is to use statsmodels. https://www.statsmodels.org/devel/examples/notebooks/generated/ols.html

In [8]:
```
#run this cell

import statsmodels.api as sm
from statsmodels.formula.api import ols
```

In [9]:
```
#first we have to rename the columns to remove spaces
DATA.columns = ['Ecotype','Max_Height']

#now we run an ANOVA using the ordinary least squares method
results = ols('Max_Height ~ Ecotype', data=DATA).fit()

#print the ANOVA table
print(results.summary())

#retrieve the actual pvalue
print('\nPvalues')
print(results.pvalues)
```

One-way ANOVA

One way to preform a oneway anova is to use scipy f_oneway. https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.f_oneway.html

In [6]:
```
#run a oneway anova
statsanova = stats.f_oneway(Local['Max Height (cm)'],Southern['Max Height (cm)'])
print(statsanova)
```

```
F_onewayResult(statistic=16.68642351051606, pvalue=5.5182675586439454e-05)
```

Another method is to use statsmodels. https://www.statsmod

In [8]:
```
#run this cell

import statsmodels.api as sm
from statsmodels.formula.api import ols
```

In [9]:
```
#first we have to rename the columns to remo
DATA.columns = ['Ecotype','Max_Height']

#now we run an ANOVA using the ordinary leas
results = ols('Max_Height ~ Ecotype', data=D

#print the ANOVA table
print(results.summary())

#retrieve the actual pvalue
print('\nPvalues')
print(results.pvalues)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             Max_Height   R-squared:                       0.047
Model:                            OLS   Adj. R-squared:                  0.044
Method:                 Least Squares   F-statistic:                     16.69
Date:                Mon, 03 Jun 2019   Prob (F-statistic):           5.52e-05
Time:                        17:38:10   Log-Likelihood:                -1413.8
No. Observations:                 338   AIC:                             2832.
Df Residuals:                     336   BIC:                             2839.
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Intercept       61.2366      1.649     37.125      0.000      57.992      64.481
Ecotype[T.Southern]    -7.9141      1.937     -4.085      0.000     -11.725      -4.103
==============================================================================
Omnibus:                       21.865   Durbin-Watson:                   0.075
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               29.130
Skew:                           0.501   Prob(JB):                     4.73e-07
Kurtosis:                       4.031   Cond. No.                         3.58
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Pvalues
Intercept             6.103906e-121
Ecotype[T.Southern]     5.518268e-05
dtype: float64
```

Practice Problem

Problem 1

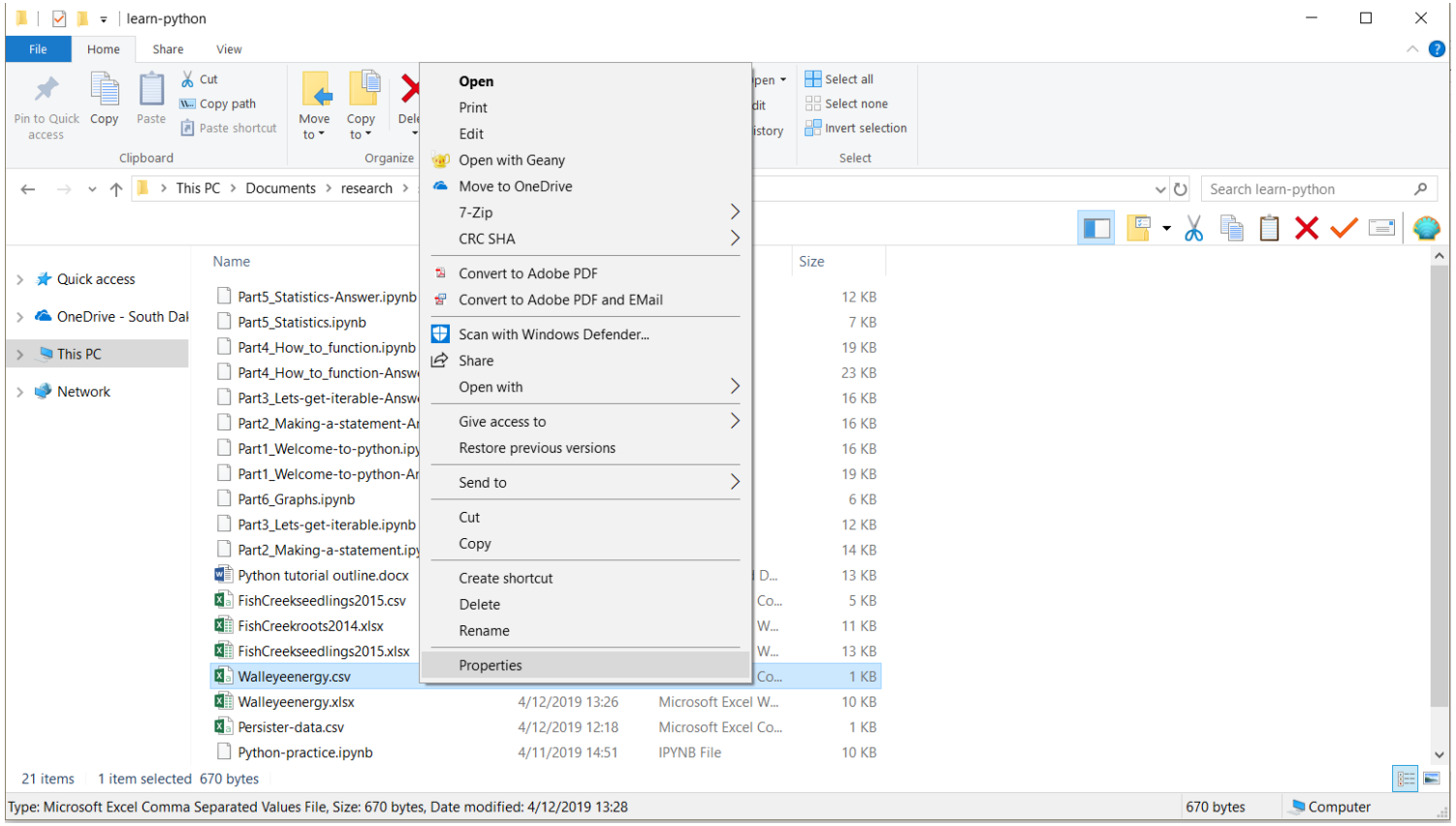
Go through another dataset, Walley energy. Import the csv file (Walleyenergy.csv) and calculate the average and standard deviation for both near-shore and open-water energy. Run a t-test between the two groups and print the p-value.

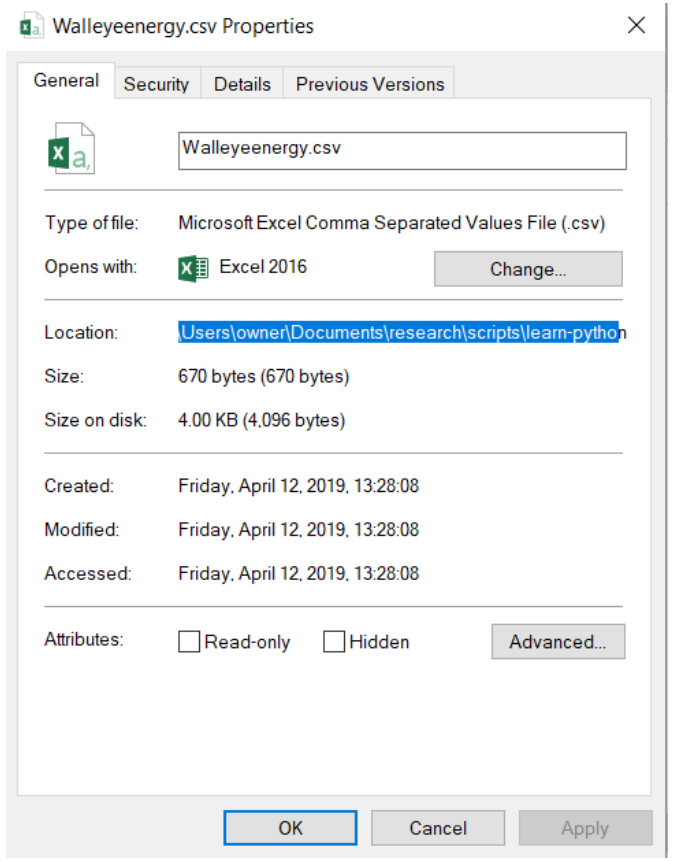
Here's some help for calculating the mean and standard deviation with pandas: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html **Hint:** try searching for mean and std under methods.

In [ ]:

```
##Write a code below to answer the above question
#Be sure to comment out your code (explain what each section is doing in comments)
```

- An easy way to find the file location is to right click on the file in your file explorer and go to “Properties”
- Copy and paste the location
- If you are on windows be sure to change all ‘\’ to ‘/’
- Also be sure to add the actual name of the file at the end
- e.g. C:/Users/user/Desktop/walleyeenergy.csv





pandas.DataFrame.mean

`DataFrame.`**`mean`**(*axis=None*, *skipna=None*, *level=None*, *numeric_only=None*, ***kwargs*) [source]

Return the mean of the values for the requested axis.

Parameters:

axis : *{index (0), columns (1)}*

Axis for the function to be applied on.

skipna : *bool, default True*

Exclude NA/null values when computing the result.

level : *int or level name, default None*

If the axis is a MultiIndex (hierarchical), count along a particular level, collapsing into a Series.

numeric_only : *bool, default None*

Include only float, int, boolean columns. If None, will attempt to use everything, then use only numeric data. Not implemented for Series.

****kwargs**

Additional keyword arguments to be passed to the function.

Returns:

mean : *Series or DataFrame (if level specified)*

pandas.DataFrame.std

`DataFrame.std(axis=None, skipna=None, level=None, ddof=1, numeric_only=None, **kwargs)` [source]

Return sample standard deviation over requested axis.

Normalized by N-1 by default. This can be changed using the ddof argument

Parameters:

axis : *{index (0), columns (1)}*

skipna : *boolean, default True*

Exclude NA/null values. If an entire row/column is NA, the result will be NA

level : *int or level name, default None*

If the axis is a MultiIndex (hierarchical), count along a particular level, collapsing into a Series

ddof : *int, default 1*

Delta Degrees of Freedom. The divisor used in calculations is N - ddof, where N represents the number of elements.

numeric_only : *boolean, default None*

Include only float, int, boolean columns. If None, will attempt to use everything, then use only numeric data. Not implemented for Series.

Returns:

std : *Series or DataFrame (if level specified)*

In [19]:

```
##Write a code below to answer the above question
#Be sure to comment out your code (explain what each section is doing in comments)

#get filename
filename = 'C:/Users/owner/Documents/research/scripts/learn-python/Walleyeenergy.csv'

#import csv with pandas
walleydata = pd.read_csv(filename)

#rename columns


#near shore energy
navg = walleydata['Proportion near-shore energy'].mean()
nstd = walleydata['Proportion near-shore energy'].std()

print('Near-shore avg:',navg)
print('Near-shore std:',nstd)

#line break
print('')

#near shore energy
wavg = walleydata['Proportion open-water energy'].mean()
wstd = walleydata['Proportion open-water energy'].std()

print('Open-water avg:',wavg)
print('Open-water std:',wstd)

```

```
Near-shore avg: 0.7580204605000002
Near-shore std: 0.06421102861679646

Open-water avg: 0.24197953949999998
Open-water std: 0.06421102861679646
```

In [24]:

```
#run t-test using scipy

walleye_ttest = stats.ttest_ind(walleyedata['Proportion near-shore energy'],walleyedata['Proportion open-water energy'])
print('p-value:',walleye_ttest[1])
```

```
p-value: 5.784794215729516e-18
```

Graphs - Python Tutorial Part 6

By: Heather S. Deter

There are a couple of different options for graphing in python. The most common one is matplotlib https://matplotlib.org/. We're going to be using numpy arrays to organize our data. http://www.numpy.org/.
Pandas also has a built in graph function that uses matplotlib. https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.html

Let's try it out.

1. Get some data.
We are going to be using data from an antibiotic survival test with and without induction of a synthetic circuit. The population is measured in colony forming units per millileter (CFU/ml).

In [1]:
```
#run this cell

import numpy as np
import matplotlib.pyplot as plt

# import some data using numpy
CSVname = 'C:/Users/owner/Documents/research/scripts/learn-python/Persister-data.csv'

#skip the header so that everything we import is a number
DATA = np.genfromtxt(CSVname, delimiter = ',',skip_header = 1)

print(DATA)
```

```
[[1.00e+00 0.00e+00 9.48e+05]
 [2.00e+00 0.00e+00 1.81e+06]
 [3.00e+00 0.00e+00 1.36e+06]
 [4.00e+00 0.00e+00 1.26e+06]
```

2. Separate the data and do some calculations.

In [2]:
```
#let's separate out the data according to treatment
##we could just slice it, but a more flexible method is to use np.where

#first make a copy of the data so that we keep the orginial
data = np.copy(DATA)

#get the treated location
treatedloc = np.where(data[:,1] == 100)[0]

#slice the data using treatedloc
treated = data[treatedloc,2]
untreated = np.delete(data[:,2],treatedloc)

print(treated)
print(untreated)
```

```
[22700000. 43600000. 44200000. 78700000. 62200000. 48200000. 24600000.
 51300000. 19900000. 91400000. 58000000. 88900000.]
[ 948000. 1810000. 1360000. 1260000. 1850000. 1020000. 3940000. 4410000.
 3500000. 4080000. 2480000. 4520000.]
```

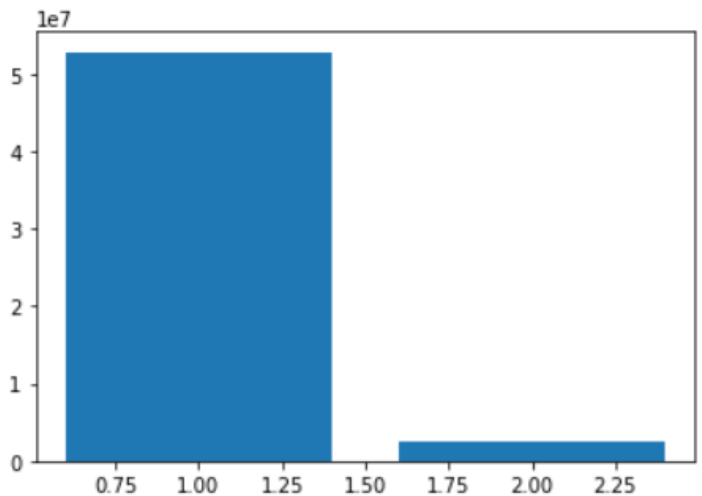
In [3]:
```
#calculate mean and std
tmean = np.mean(treated)
umean = np.mean(untreated)
tstd = np.std(treated)
ustd = np.std(untreated)

print('treated',tmean,tstd)
print('untreated',umean,ustd)
```

```
treated 52808333.333333336 23354245.093534686
untreated 2598166.6666666665 1339096.8245633158
```

3. Plot a bargraph.

In [4]:
```
#now we make a bargraph
plt.figure()
plt.bar([1,2],[tmean,umean])
```

Out[4]: <BarContainer object of 2 artists>



4. Make it pretty. This is where referring to the documentation is super helpful: https://matplotlib.org/api/_as_gen/matplotlib.pyplot.bar.html

In [5]:

```
#first we give the figure a size (x,y)
plt.figure(figsize = (3,5))

#add some errorbars when we plot the bargraph (yerr), make it green (color), and give the errorbars caps (capsize)
plt.bar([1,2],[tmean,umean],yerr = [tstd,ustd],color = ['g'],capsize = 5)

#add axis titles
plt.xlabel('Group')
plt.ylabel('CFU/ml')

#give the plot a title
plt.title('Survival')

#label the x-axis ticks
plt.xticks([1,2],['Treated','Untreated'])

#save the figure
##to save the figure uncomment the line below and change the path as you desire

# plt.savefig('BarGraph.png')
```

matplotlib.pyplot.bar ¶

matplotlib.pyplot.bar(*x*, *height*, *width=0.8*, *bottom=None*, *, *align='center'*, *data=None*, ***kwargs*) [source]

Make a bar plot.

The bars are positioned at *x* with the given *align*ment. Their dimensions are given by *width* and *height*. The vertical baseline is *bottom* (default 0).

Each of *x*, *height*, *width*, and *bottom* may either be a scalar applying to all bars, or it may be a sequence of length N providing a separate value for each bar.

Parameters:

x : sequence of scalars

The x coordinates of the bars. See also *align* for the alignment of the bars to the coordinates.

height : scalar or sequence of scalars

The height(s) of the bars.

width : scalar or array-like, optional

The width(s) of the bars (default: 0.8).

bottom : scalar or array-like, optional

The y coordinate(s) of the bars bases (default: 0).

align : {'center', 'edge'}, optional, default: 'center'

Alignment of the bars to the *x* coordinates:

- 'center': Center the base on the *x* positions.
- 'edge': Align the left edges of the bars with the *x* positions.

Other Parameters:

color : scalar or array-like, optional

The colors of the bar faces.

edgecolor : scalar or array-like, optional

The colors of the bar edges.

linewidth : scalar or array-like, optional

Width of the bar edge(s). If 0, don't draw edges.

tick_label : string or array-like, optional

The tick labels of the bars. Default: None (Use default numeric labels.)

xerr, yerr : scalar or array-like of shape(N,) or shape(2,N), optional

If not *None*, add horizontal / vertical errorbars to the bar tips. The values are +/- sizes relative to the data:

- scalar: symmetric +/- values for all bars
- shape(N,): symmetric +/- values for each bar
- shape(2,N): Separate - and + values for each bar. First row

  contains the lower errors, the second row contains the upper errors.

- *None*: No errorbar. (Default)

See Different ways of specifying error bars for an example on the usage of `xerr` and `yerr`.

ecolor : scalar or array-like, optional, default: 'black'

The line color of the errorbars.

capsize : scalar, optional

The length of the error bar caps in points. Default: None, which will take the value from `rcParams["errorbar.capsize"]`.

error_kw : dict, optional

Dictionary of kwargs to be passed to the `errorbar` method. Values of *ecolor* or *capsize* defined here take precedence over the independent kwargs.

log : bool, optional, default: False

If *True*, set the y-axis to be log scale.

orientation : {'vertical', 'horizontal'}, optional

This is for internal use only. Please use `barh` for horizontal bar plots. Default: 'vertical'.

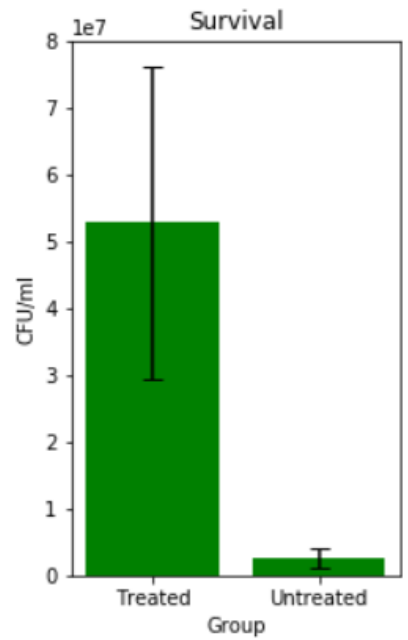
4. Make it pretty. This is where referring to the documentation is super helpful: https://matplotlib.org/api/_as_gen/matplotlib.pyplot.bar.html

In [5]:

```
#first we give the figure a size (x,y)
plt.figure(figsize = (3,5))

#add some errorbars when we plot the bargraph (yerr), make it green (color), and give the errorbars caps (capsize)
plt.bar([1,2],[tmean,umean],yerr = [tstd,ustd],color = ['g'],capsize = 5)

#add axis titles
plt.xlabel('Group')
plt.ylabel('CFU/ml')

#give the plot a title
plt.title('Survival')

#label the x-axis ticks
plt.xticks([1,2],['Treated','Untreated'])

#save the figure
##to save the figure uncomment the line below and change the

# plt.savefig('BarGraph.png')
```

Out[5]:

```
([<matplotlib.axis.XTick at 0x17f320e69b0>,
  <matplotlib.axis.XTick at 0x17f320e6390>],
 <a list of 2 Text xticklabel objects>)
```



Practice Problem

We are going to make a scatterplot. It's just like making a bargraph, but instead of plt.bar() we will use plt.scatter(). See https://matplotlib.org/api/_as_gen/matplotlib.pyplot.scatter.html#matplotlib.pyplot.scatter. Use the Walleye energy dataset (Walleyenergy.csv), plot both the open-water and near-shore on the same graph in two different colors. Try adding a legend. https://matplotlib.org/api/_as_gen/matplotlib.pyplot.legend.html

In [ ]:

```
##Write a code below to answer the above question
#Be sure to comment out your code (explain what each section is doing in comments)
```

matplotlib.pyplot.scatter

matplotlib.pyplot.scatter(*x*, *y*, *s=None*, *c=None*, *marker=None*, *cmap=None*, *norm=None*, *vmin=None*, *vmax=None*, *alpha=None*, *linewidths=None*, *verts=None*, *edgecolors=None*, *, *plotnonfinite=False*, *data=None*, ***kwargs*) [source]

A scatter plot of *y* vs *x* with varying marker size and/or color.

Parameters:

x, y : array_like, shape (n,)

The data positions.

s : scalar or array_like, shape (n,), optional

The marker size in points**2. Default is `rcParams['lines.markersize'] ** 2`.

c : color, sequence, or sequence of color, optional

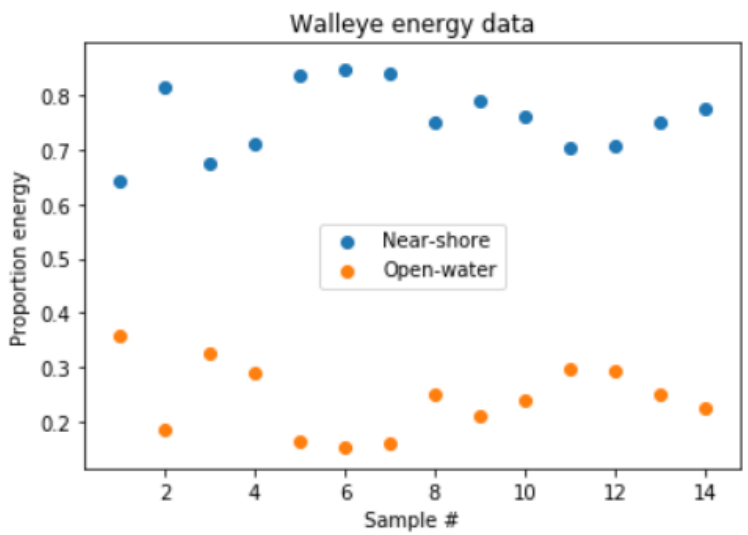
The marker color. Possible values:

- A single color format string.
- A sequence of color specifications of length n.
- A sequence of n numbers to be mapped to colors using *cmap* and *norm*.
- A 2-D array in which the rows are RGB or RGBA.

In [14]:

```
##Write a code below to answer the above question
#Be sure to comment out your code (explain what each section is doing in comments)

import pandas as pd

#get filename
filename = 'C:/Users/owner/Documents/research/scripts/learn-python/Walleyeenergy.csv'

#import csv with pandas
walleyedata = pd.read_csv(filename)

plt.scatter(walleyedata['Sample #'],walleyedata['Proportion near-shore energy'])
plt.scatter(walleyedata['Sample #'],walleyedata['Proportion open-water energy'])

plt.legend(['Near-shore','Open-water'],loc = 'center')

plt.title('Walleye energy data')
plt.ylabel('Proportion energy')
plt.xlabel('Sample #')
```

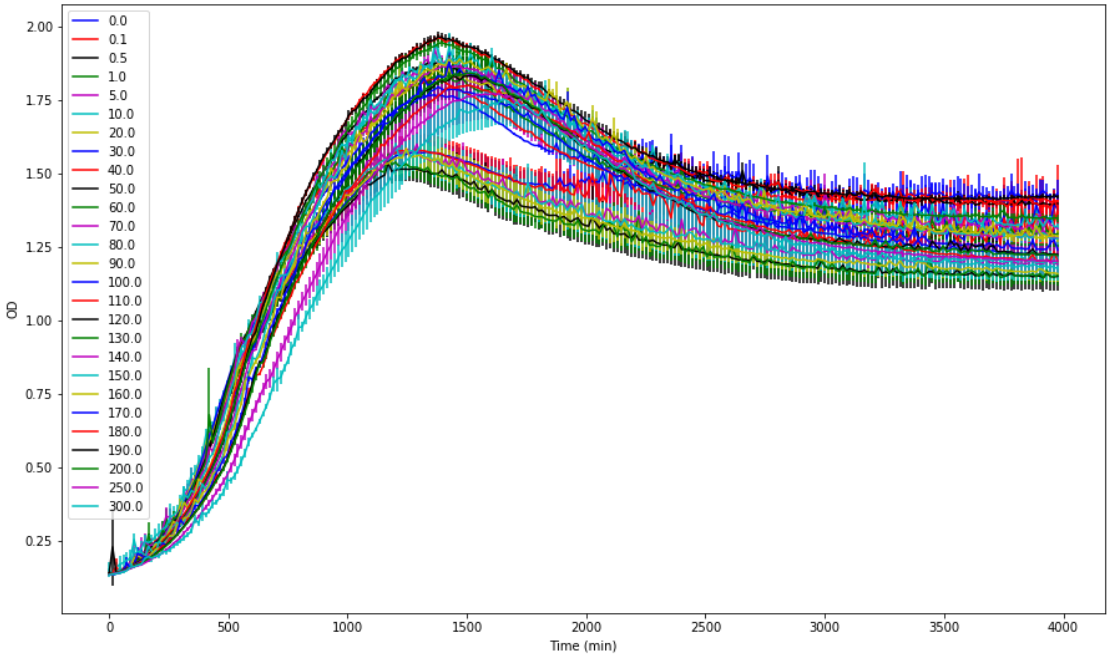
Out[14]: Text(0.5, 0, 'Sample #')

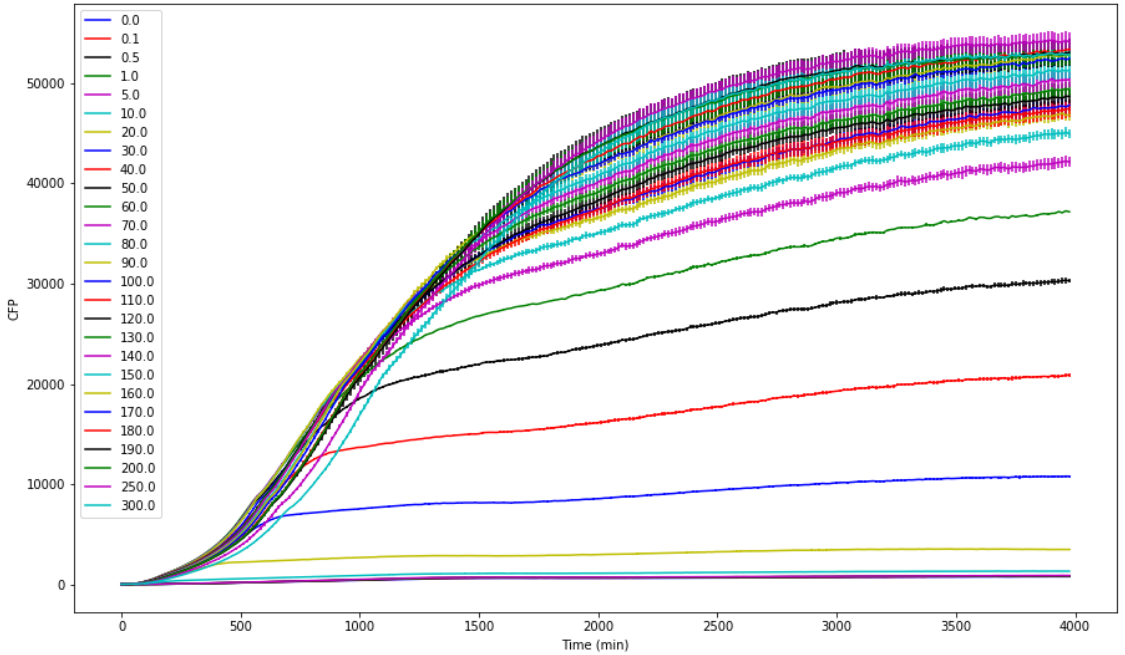


What's next? - Apply what you know

- Statistical analysis of large datasets
- Graphing large datasets

- Example: Plate reader data – over 45,000 datapoints





What’s next? - Expand on what you know

- Learn more about statistics
  - https://scipy-lectures.org/packages/statistics/index.html
- Learn more about graphing
  - https://plot.ly/python/

Plotly Python Open Source Graphing Library

Plotly's Python graphing library makes interactive, publication-quality graphs online. Examples of how to make line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, heatmaps, subplots, multiple-axes, polar charts, and bubble charts.

- There are plenty of other tutorials available online

What’s next? - Learn some new skills

- Machine learning
  - https://machinelearningmastery.com/machine-learning-in-python-step-by-step/
  - https://realpython.com/tutorials/machine-learning/
- Image analysis
  - https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_table_of_contents_imgproc/py_table_of_contents_imgproc.html
- Bioinformatics
  - https://biopython.org/
- And more!

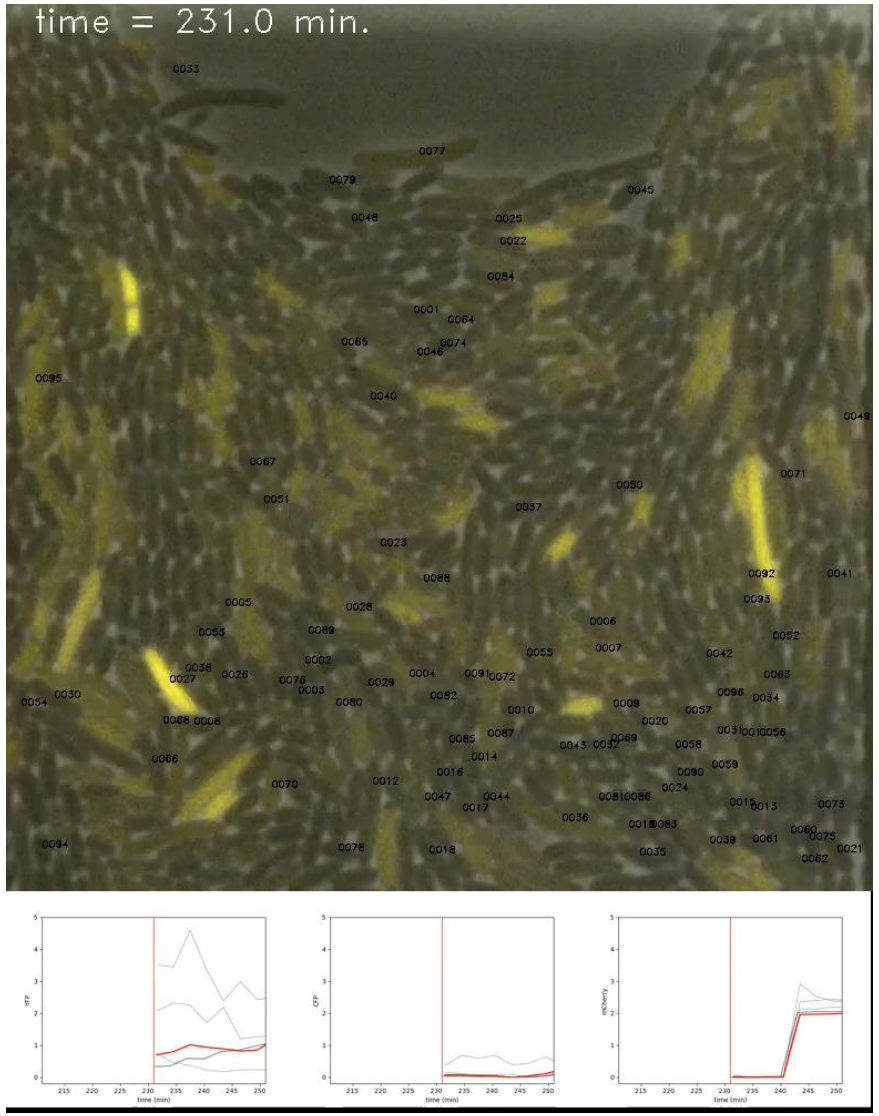
How I use Python

Image Analysis

- Cell tracking
- Colony counting
- Fluorescence microscopy

Data analysis

- RNA-seq and ribosome profiling data
- Bioplate reader data
- Other experimental data



GitHub

https://github.com/hdeter

Tips for your Python journey

- Use your resources
  - Search the web for solutions to the problems you’re facing
    - Chances are you are not the first person to face the problem
  - Ask the community
- Forget about being perfect
  - So long as you reach your end goal you have been successful
- Try new things
  - You never succeed at something you never tried

PLOS | COMPUTATIONAL BIOLOGY

EDITORIAL

Ten simple rules for biologists learning to program

Maureen A. Carey[1], Jason A. Papin[2]*

1. Begin with the end in mind
2. Baby steps are steps
3. Immersion is the best learning tool
4. Phone a friend
5. Learn how to ask questions
6. Don’t reinvent the wheel
7. Develop good habits early on
8. Practice makes perfect
9. Teach yourself
10. Just do it

The End - Go forth and code

```
print "Hello World"
```