Python

**Virtual Environment**

Try to create a virtual environment whenever starting a new project

Creating a virtual environment in python

* Python -m venv venv
  + 1st venv calls the venv package, and the second names it venv
* Source venv/bin/activate
  + Activates it

Ending virtual Environment

* Ctrl + C

**Terminal Commands**

* Cd
  + List current directory
* Ls
  + List directory
* Pwd
  + Print working directory
* Clear
  + Clears terminal screen
* Q
  + Quits a program
* Quit()
  + Quits python

**Python Commands**

* ;
  + Use to add multiple things on a single line of code
* \
  + Extends code to a new line
* = assigns a variable; == test if they have the same value

**Python Generally Understood**

Classes imported are usually named with uppercase

Methods are usually lowercase

Packages usually imported as lowercase

**Python Basics**

For loops, if then, while loops

* For loops: Iterate over a list
* While: Iterate until something is no longer true

Variables, methods, class, functions, objects

Lists, Tuples, Dictionaries, sets

* List
  + Ordered
  + [ ]
* Tuples
  + Immutable
  + ( )
* Dictionaries
  + Key pair value
  + { }
* Sets
  + Unordered list
  + { }

**Python advanced**

Decorators, lambda function, closures

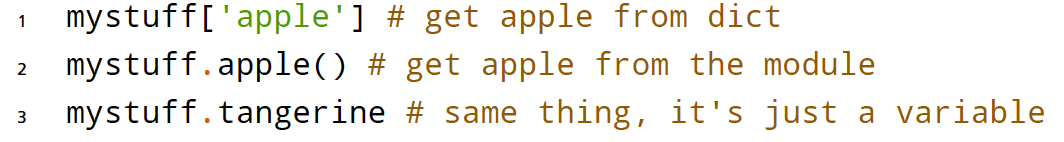
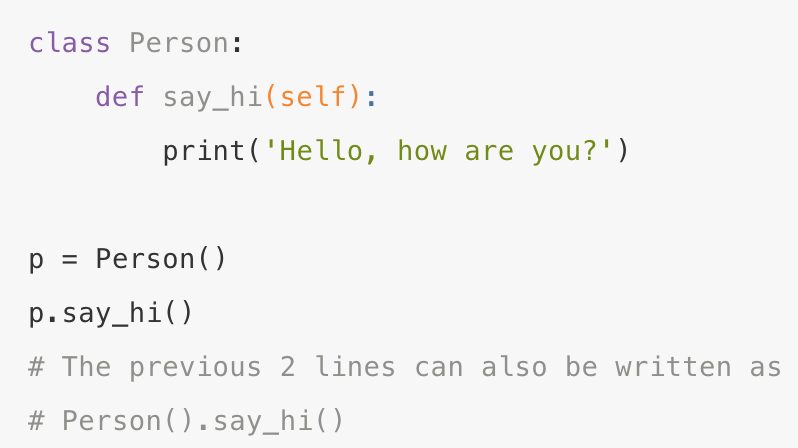
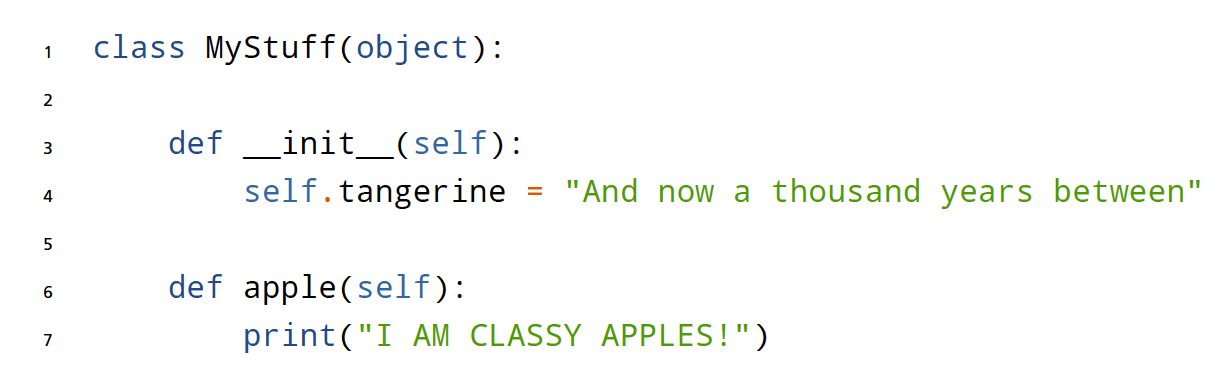
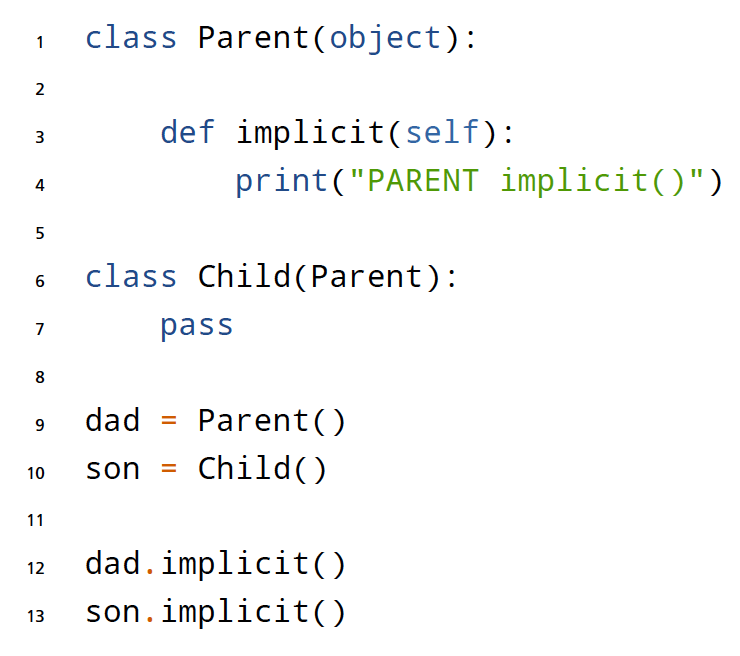
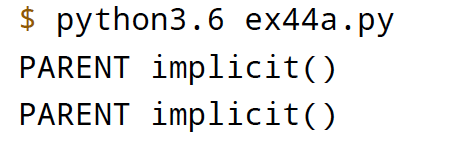
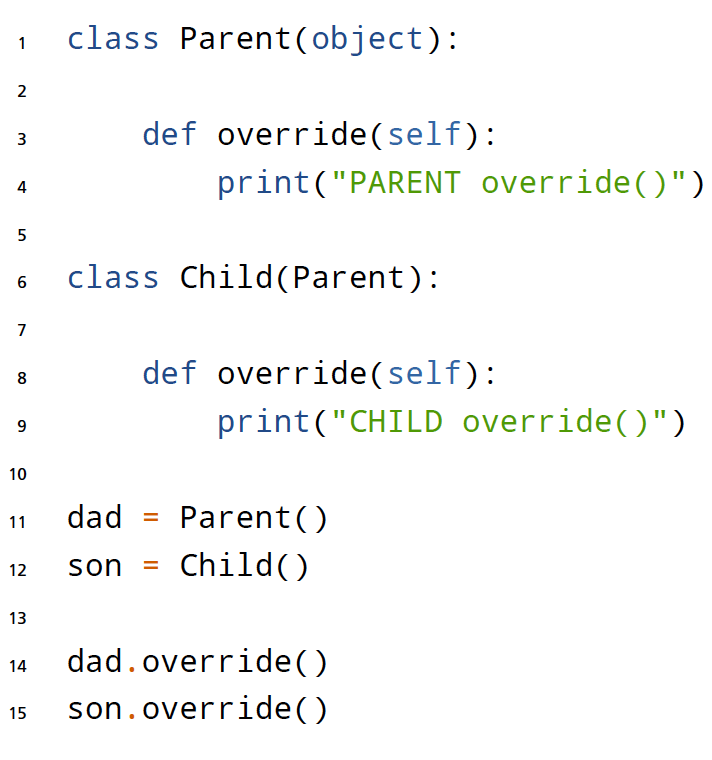
* Decorators
* Lambda function
* Methods
  + A function that belongs to a class
* Closures
  + Function that remembers the value from the enclosing lexical scope, even when the program flow is no longer in that scope
  + 

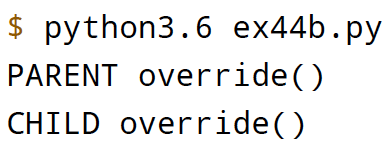
**Posting on Forum for help**

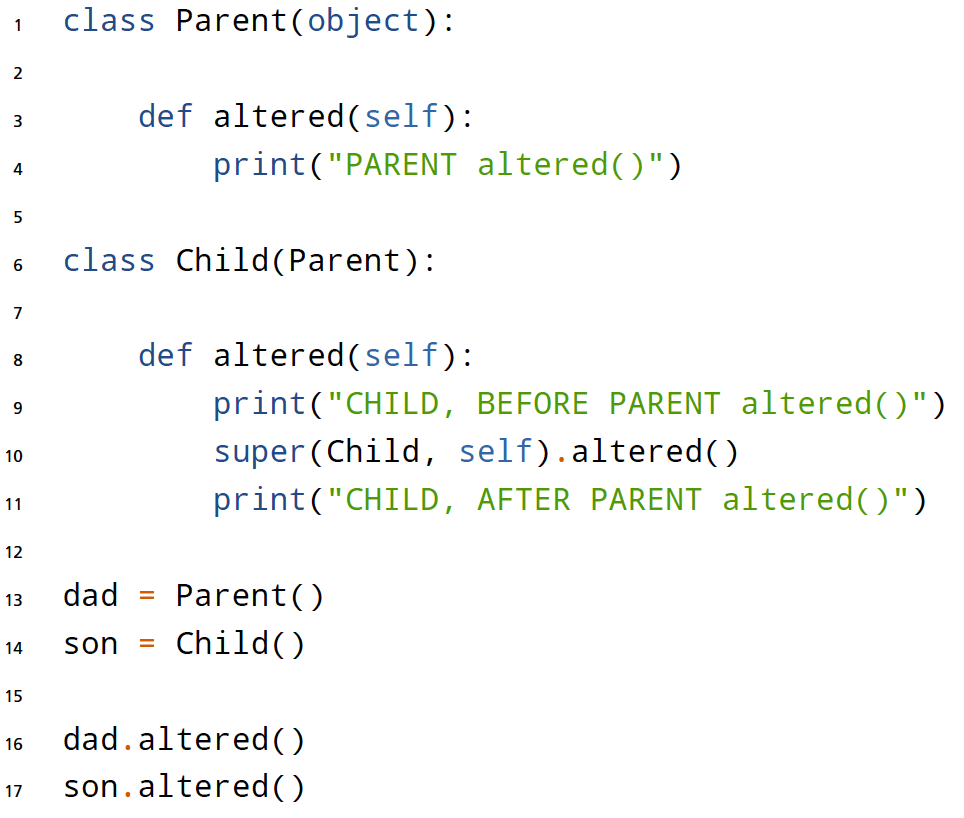
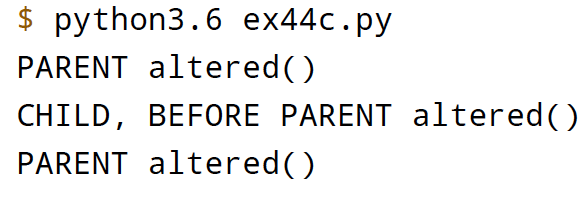
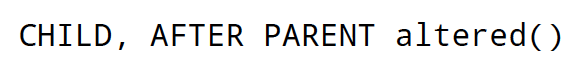
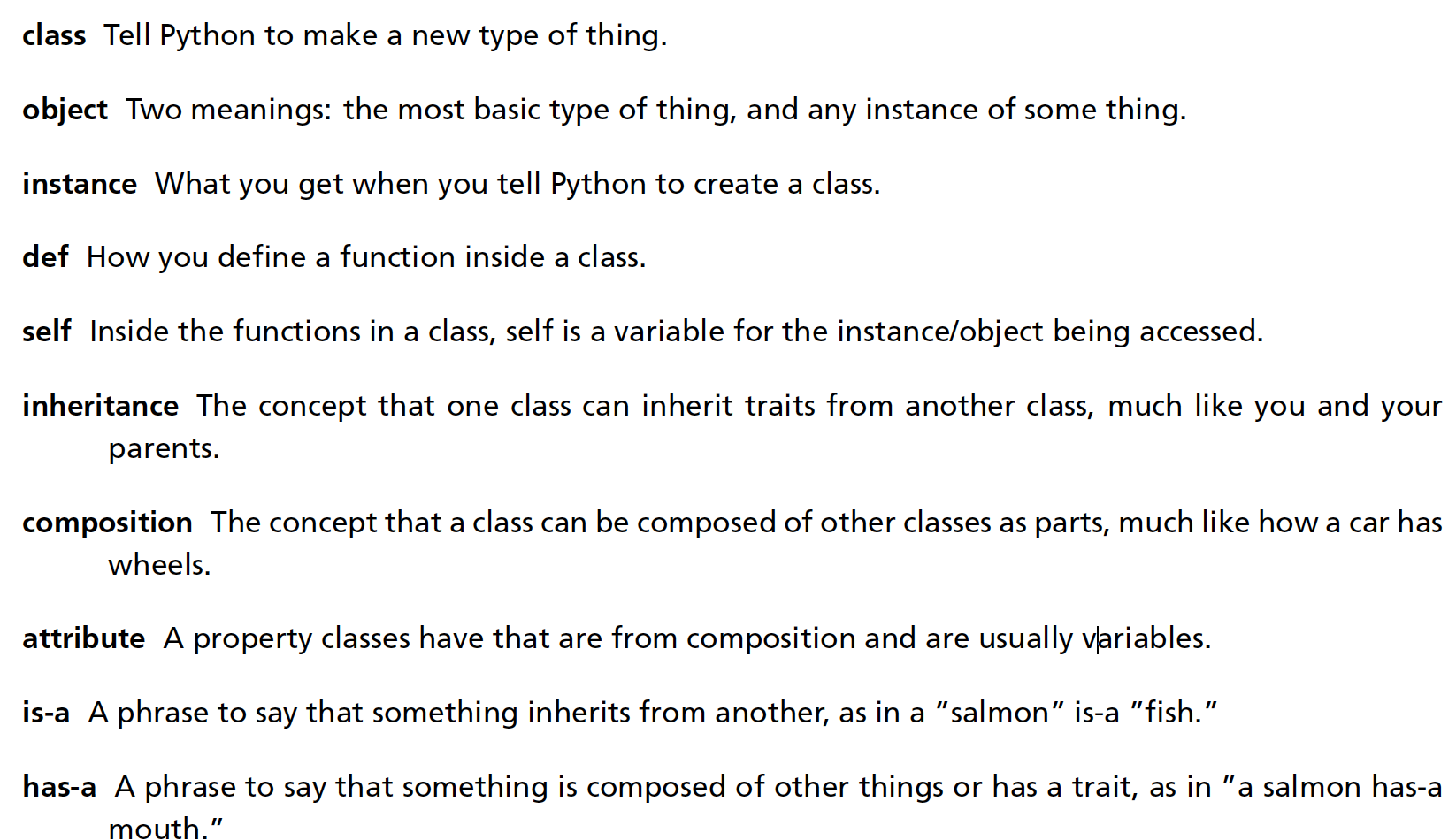
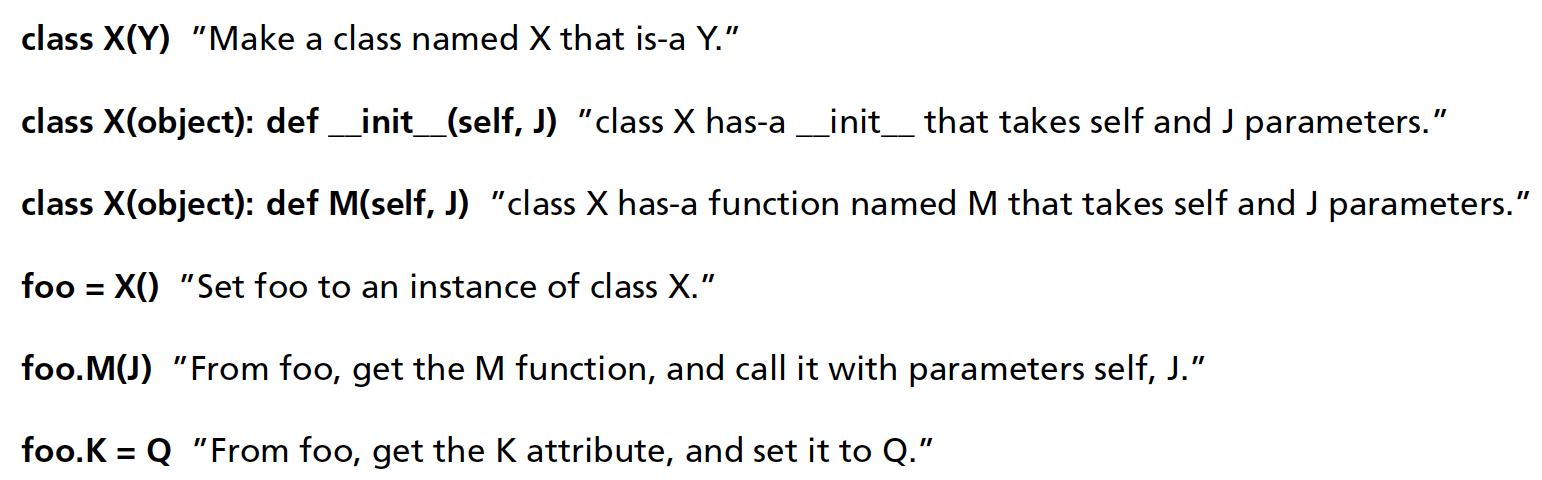
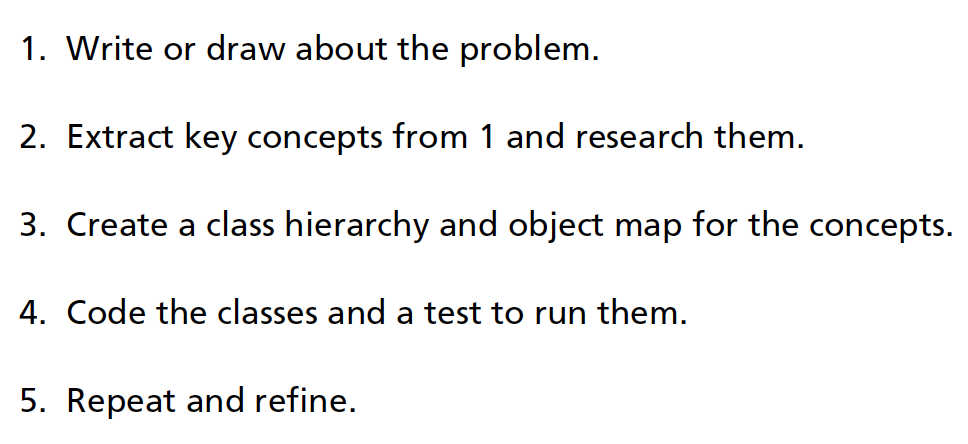
* Share specific examples
* Explains which aspects of the example you understand and what parts you don’t
* Try to explain to your best ability what is going on in the example even if what you say is inaccurate

**Python Hot Keys**

===========================================================================================

* + Lists
    - When to use a list
      * Maintain order
        + Listed order, not sorted order
      * Need to access the contents randomly by a number
      * Go through content linearly
  + Dictionary
    - Like a database for storing & organizing data
    - Maps things to a key
    - DO NOT have order
      * 
    - When to use a dictionary
      * When you have to take one value and “look up” another value
  + Modules
    - All python files are modules/libraries
      * You can call other python files into your current script
        + Acess Python file with functions or variables in it
      * Called by . 🡺 dot
        + 
    - Similar to a dictionary in that…
      * Take a key to get something out of that key
  + Class (<https://python.swaroopch.com/oop.html>)
    - Def: A way to take a grouping of functions and data and place them inside a container so you can access them with the . (dot) operator.
      * Like a blueprint or definitions for creating new mini modules
      * Instantiation Def: How you create an object from the class
    - Fields 🡺 Variables that belong to an object or class
      * Class variable – Accessed by all instances of the class, affects all class variables
      * Object Variable – Owned by individual instance, affects only that object
    - Objects belong to a class
      * Methods 🡺 Functions of objects are called methods
    - Attributes 🡺 Fields & Methods of a class
    - Class methods MUST HAVE an extra first parameter
      * Usually referred to as self
      * i.E: You have a class and a object, you call a method of the object it will be converted to myclass.method(self,arg1,arg2)
        + Means you will also have a minimum of one argument; self
      * 
    - \_\_int\_\_ 🡺 set the initial attributes of the instance
      * Must add self
        + Without it, variables become ambiguous
        + Unsure if it means instance variable or local variable
      * If you’re using a class and you have a variable, you must add 🡺 self.variable
        + 
    - Decorators 🡺 shortcut to calling a wrapper function
      * i.e. a function that "wraps" around another function so that it can do something before or after the inner function
    - inheritance 🡺 implementing a type and subtype relationship between classes
      * I.E: School Members (base class or superclass)
        + Teacher (derived class or subclass)
        + Students (derived class or subclass)
      * Implicit inheritance
        +   
          
      * Override
        + 

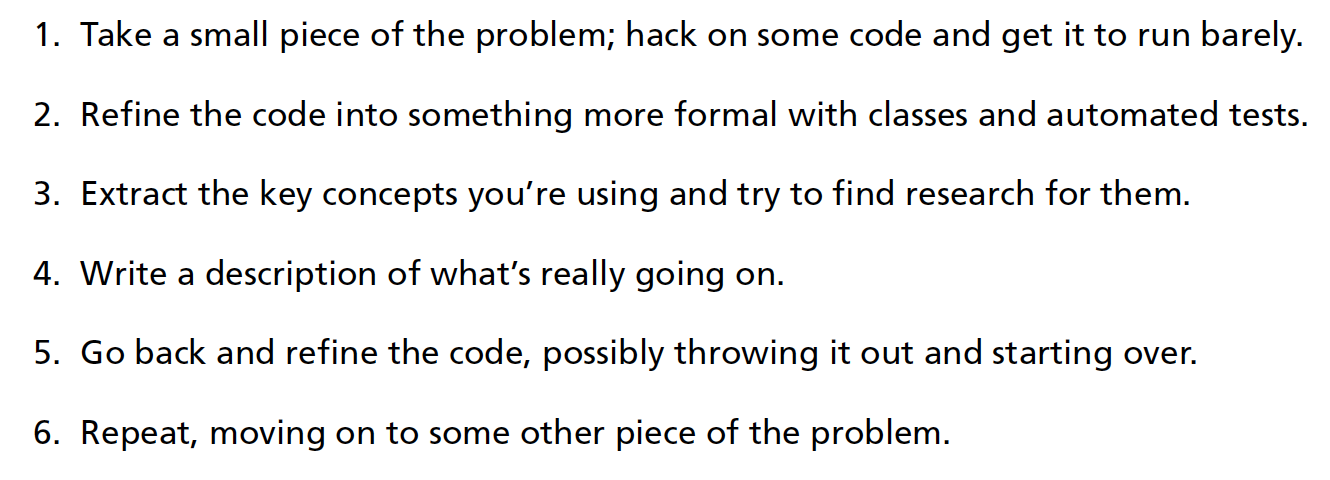


* + - * Altered
        +   
            
          
    - Inheritance vs Composition
      * AVOID multiple inheritance
      * Use Composition to package code into modules that are used in many different unrelated places and situations
      * Use Inheritance only when there are clearly related and reusable pieces of code that fit under a single common concept
    - “is-a”
      * when you talk about objects and classes being related to each other by a class relationship
    - “has-a”
      * When you talk about objects and classes that are related only because they reference each other
    - Class
      * Like a blueprint or template 🡺 manufacturing cars at a company
      * template used to bundle up data and methods whenever they depend on each other
  + Function style
    - Short and simple to be reused
  + Class Style
    - Camel case 🡺 SuperGoldFactory instead of super\_gold\_factory
    - Functions should use underscore format
    - Be consistent on how you organize function arguments
      * Make sure all class that deal with each argument are in the same order
        + (User, dog, cat) for all arguments / parameters in methods (functions) of a class
    - Always have class Name(object) format
  + Comment style
    - Comment why you are doing what you are doing
    - For functions write comments on what someone can do with that function
    - Comments relatively short and to the point
  + OOP Vocabulary
    - 
    - 
  + OOP Process
    - Top down approach
      * 
      * I.E
        + 1. Paragraph of the game
        + 1. Scenes / Key Concepts
        + 2. List all nouns

Classes & Subclasses

* + - * + 2. List all the verbs

Functions

* + - * + 3. Map the class and objects
        + 3. Make hierarchy
        + 3. Actions to transition
        + 4. Code
        + 4. Test
        + 5. Repeat & Refine
    - **Bottom up approach**
      * 
  + Tuple
    - List you can’t modify

ELI5

* Google things
  + Know how to filter good and bad data
* Make a module / library in python that teaches the course
* Basics of Python
  + Data Structure (1)
    - List
    - Dictionaries
  + Symbols / Operators (1)
  + OOP (2)
    - Class
    - Objects
    - Methods
    - Decorators
    - inheritance
  + Libraries (1)
  + IDE (1)
  + Differences in Python to R
    - Stuff[0:3]
      * Calls 0,1,2
      * Starts at 0 for Python when indexing
  + Pros of Python over R
    - Productionize
    - More software engineering toolset
    - Software programming use for statistical analysis
  + Cons of Python over R
    - Faster quicker analysis
    - More stats tool libraries
    - Statistical analysis tool used for software programming
  + Resources
    - Automate the boring stuff
    - Learn Python the hard way
    - Google
* Use case
  + Automation

**===================================================================**

**Basics**

**Jupyter Notebook (** [**https://www.dataquest.io/blog/jupyter-notebook-tips-tricks-shortcuts/**](https://www.dataquest.io/blog/jupyter-notebook-tips-tricks-shortcuts/) **)**

* SHIFT + ENTER to run a script
* ESCAPE to jump out, and ENTER to click back on
* 00 to cancel any code that’s running 🡸 aka infinite loops
* C – Copy
* B – Add cell below
* A – Add cell above
* V – Paste
* MARKDOWN to create space b/w codes 🡸 m while NOT in edit mode 🡸 --- + ENTER
  + Markdown is to add comments and descriptions
  + Regular text
* # Jupyter Notebook Shortcut Tips
* # 1. # for commenting Code
* # 2. Esc will take you to command mode
* # a. A ==> insert a new cell above the current cell,
* # b. B ==> insert a new cell below the current cell
* # c. C ==> Copy
* # d. V ==> Paste
* # e. M ==> change current cell to markdown
* # f. Y ==> To change back to code
* # g. D + D ==> To delete the current cell
* # h. Enter will take you back to edit mode
* # 3. 00 ==> Cancel any code thats running
* # 4. Shift + Enter ==> Run Cell and proceed to next Cell
* # 5. Ctrl + Enter ==> Run Selected Cell and stay
* # 6. Help ==> Keyboard Shorcuts
* # 7. Function help ==> tab or Shift tab
* # 8. Escape Character ==> \
* # 9. Command + ] For Indent ; Command + [ For Dedent
* # 10. ? For Function Arguments

More Basics

* Data Structures
  + List – []
  + Tuples – ()
  + Dictionaries – {}
  + Set – {}
    - No key value relationship & unique elements
* cd for command directory
* For objects already defined
  + Click . then tab to see all the functions
* Tab will give you function
* Shift tab will give you documentation string

**Python**

* Remember Python goes from 0 to number before 🡺 [0:2] 🡺 0, 1
* # hashtag for comments
* INDENTATION IS how loop works 🡸 IMPORTANT
* Help Keyboard shortcuts for more shortcuts
* None 🡸 Makes variable null 🡸 answer = none
* Elif 🡸 Else if statement
* Slicing an array - Not making another copy when you slice an array
  + A view of that array
* [ ] Used when you want to grab specific Rows or columns within a matrix or data frame
  + List – Mutable – You can change its contents
  + Dictionaries
* ( ) used when you are using a function
  + Tuples – Use to identify Parameters
* & for MULTIPLE filters
* Ctrl + Select the function 🡺 Function info/help
* Help(function) 🡺for help and info

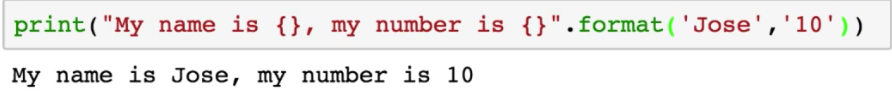
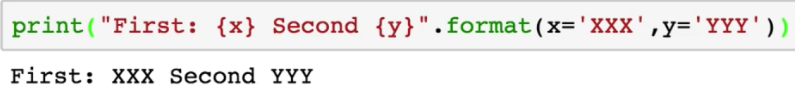
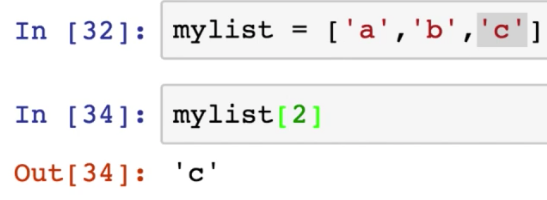
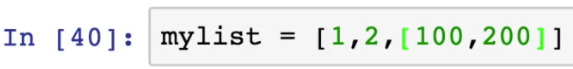
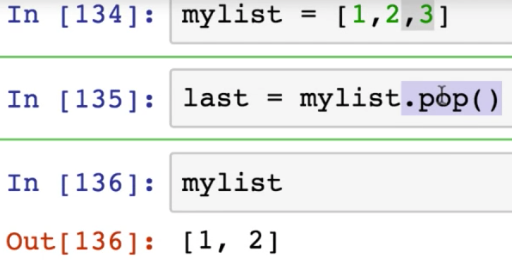
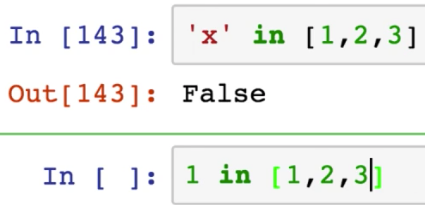
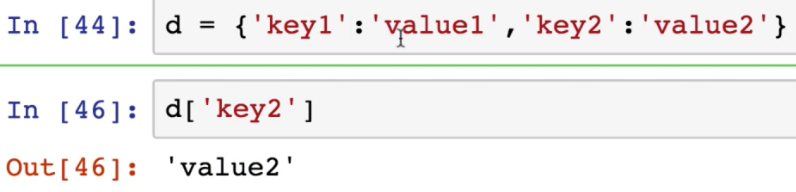
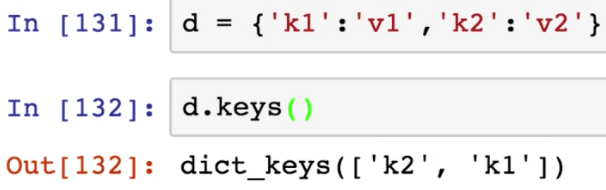
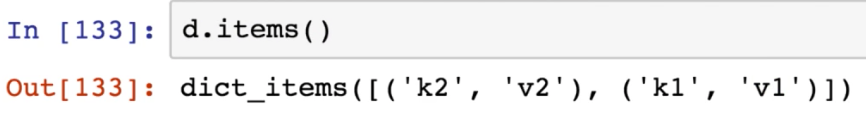
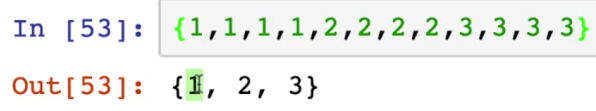
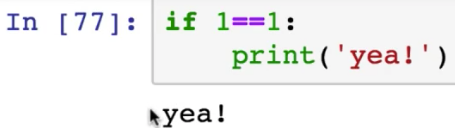
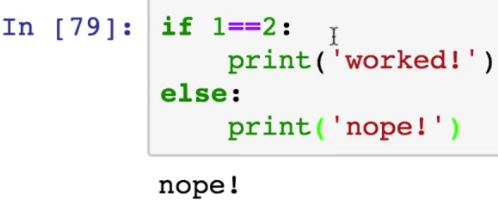
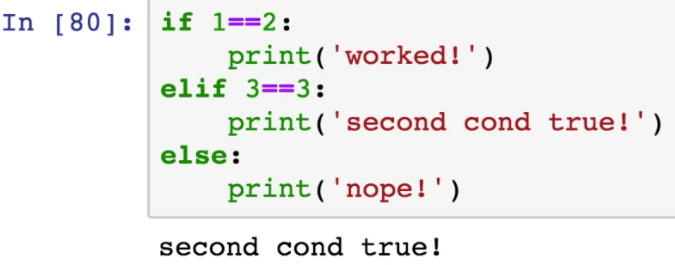
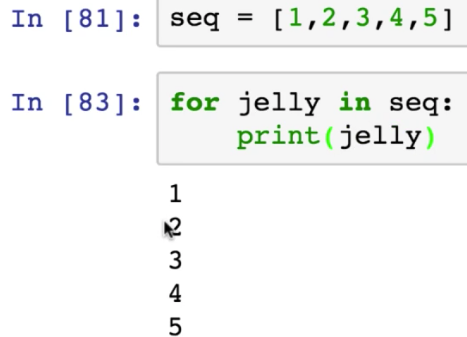
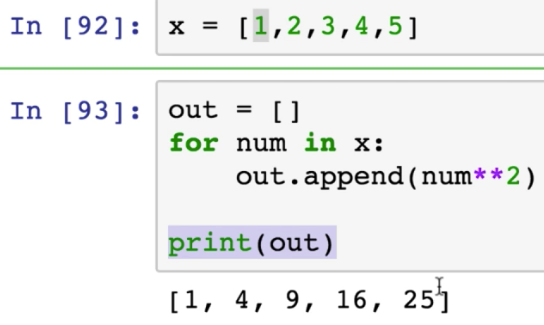
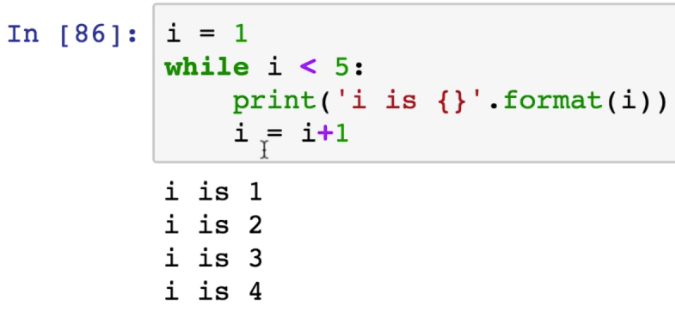
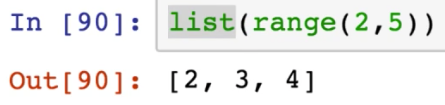
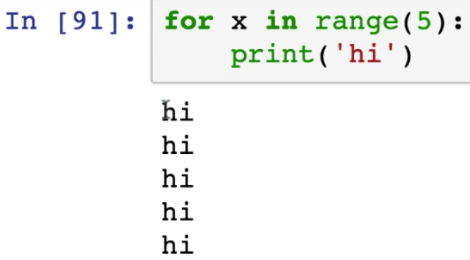
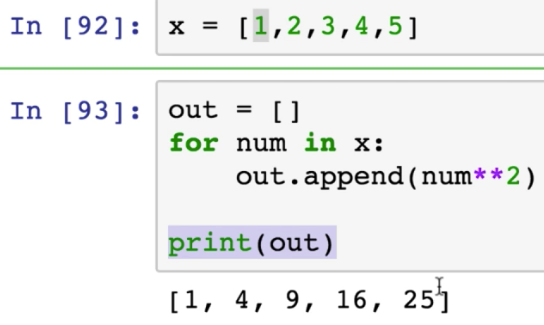
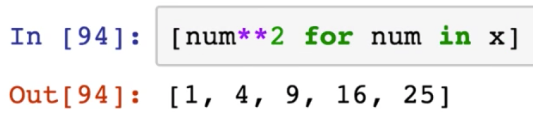
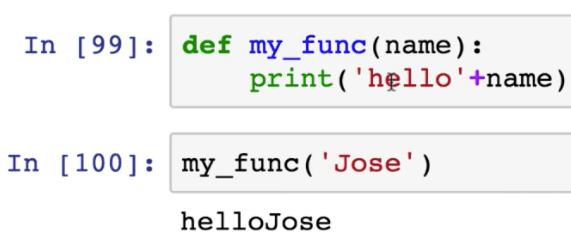
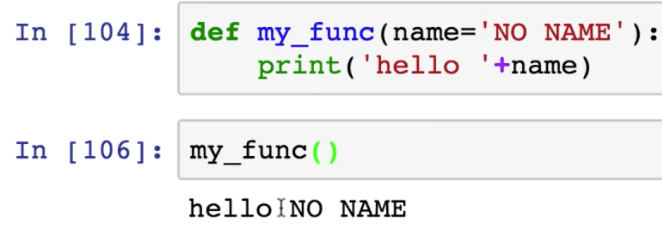
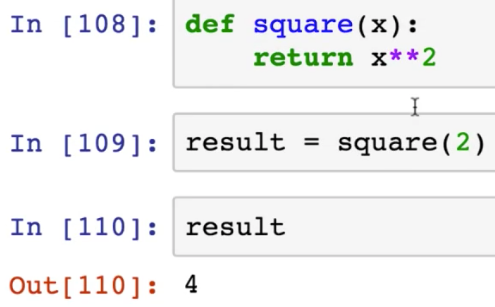
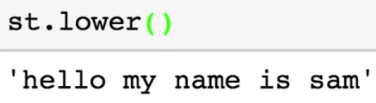
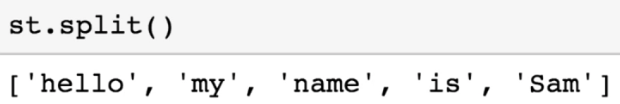
**Spyder**

* Cmd + I for help on objects
* Cmd + 1 to comment multiple lines
* To change the directory there is a directory toolbar…just change it there

**Packages**

* Pandas
  + Used for reading data
    - Read\_csv
* BeautifulSoup4
  + Data Cleansing & Text Preprocessing
* Re
  + Regular expressions to do a find-and-replace
* Nltk
  + Remove stop words / clean text data
    - Nltk & nltk.download()
* Scikit-learn
  + ML algorithims
* Numpy
  + Views data in arrays / vectors
* Word2vec
  + Text Analytics package
  + MUST install cython
    - <http://docs.cython.org/en/latest/src/quickstart/install.html>
  + Don’t remove stop words or numbers for word2vec

**Python Crash Course**

* Basics
  + Passing an object in a string
    - 
    - 
  + Lists:
    - Lists in python is known as arrays in other languages 🡺
    - Grab objects from a list is called:
      * Indexing
        + 
    - Adding to a list
      * 
        + adds do the end of the list
    - Nesting in a list
      * List inside of a list
        + 
      * Stack index calls
        + 
    - Removing an item from a list
      * Will remove the last item, but you can call which item you wish to remove
      * 
    - Checking an item in a list
      * 
  + If anything is highlighted when you type it out, it means it’s already a function
  + Dictionaries
    - Call a value based on a key
      * 
    - No order
    - To call all keys
      * 
    - To call all items
      * 
  + Tuples vs list
    - Tuples are immutable
      * You cant resign items inside of them
      * You don’t want user to change anything inside the function / variables
      * 
  + Set
    - Like dictionary but no key value pair
    - Unorder collection of unique elements
      * 
  + If statement
    - 
  + If else statement
    - 
  + Elif statement
    - Check multiple conditions
    - 
  + For loops
    - 
    - For loop to print out a list
      * 
  + While loops
    - Keep running while condition is true
      * 
  + Range function
    - 
    - Usually used in for loop
      * 
  + List comprehension
    - For loop to a list
      * Example 1 – For loop example
        + 
      * Example 2 – List comprehension example
        + 
  + Custom Function
    - Def to create a function
      * Parameters go in the myfunc() parenthesis
      * Have to add the () when calling the function
        + my\_func()
      * 
      * To add a defulat value
        + 
      * Print
        + Display value
      * Return
        + Saves the value to a variable
        + 
  + Triple quote to add comments
    - ‘’’ – open
    - ‘’’ – close
  + lambda function
    - anonymous function
      * Used a function one time
  + String manipulation
    - 
    - 
    - 
      * Splits on whitespace for default
        + To split on different character



**Types of Variables**

* Integer – Number
* Float / double – Number with Decimals
* String – Store Texts
* Boolean – Logical Operation

**Logical Expressions**

* # == Equal to
* # != Not Equal to or <>
* # <
* # >
* # <=
* # >=
* # and
* # or
* # not

**WHILE Statements**

*while condition:*

*executable code1*

*executable code2*

*executable code3*

*executable code4*

Indentation is very important in Python. That is how it knows which codes to execute

**FOR Statements**

for i in range(7):

print("Hello Python: ", i)

Like a WHILE STATEMENT but uses range 0123456 for range(7)

**IF Statements & ELSE IF**

#---- -2 ----- -1 ---- 0 ---- 1 ---- 2 ----

answer = None

x = randn()

if x > 1:

answer = "Greater than 1"

else:

answer = "Less than 1"

print(x)

print(answer)

Like a WHILE STATEMENT but only runs once, if the statement is true

#---- -2 ----- -1 ---- 0 ---- 1 ---- 2 ----

answer = None

x = randn()

if x > 1:

answer = "Greater than 1"

else:

if x >= -1:

answer = "Between -1 and 1"

else:

answer = "Less than -1"

print(x)

print(answer)

This is called a nested statement 🡸 Basically two IF statements within each other.

#---- -2 ----- -1 ---- 0 ---- 1 ---- 2 ----

#Chained Seatements

answer = None

x = randn()

if x > 1:

answer = "Greater than 1"

elif x >= -1:

answer = "Between -1 and 1"

else:

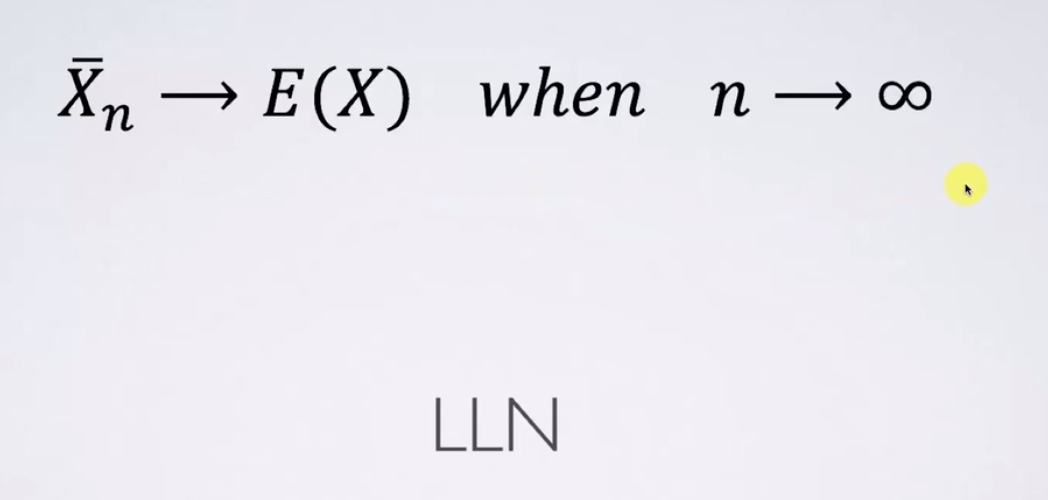
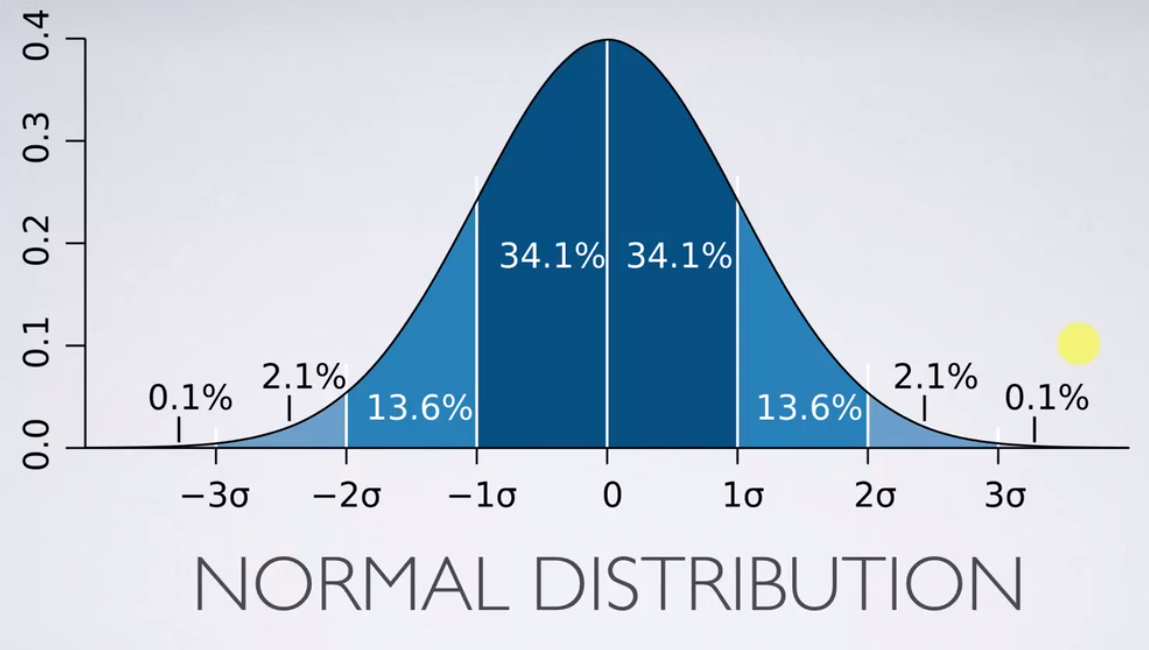
answer = "Less than -1"

print(x)

print(answer)

This is called a Chaining statement 🡸 Easier coding than a Nested statement for same results

**Theories**

* **LLN** -  
   
  + The average of the **actually measured value** converges to the **expected value** of that measurement **when n approaches infinity** (as your sample grows)
  + Coin toss => Heads will converge to .50
* **Normal Distribution –**
* 
* Standard deviation – how far from the mean it is.
* rnorm(5,10,8) 🡸 5 random variables will be around 10 but 68.2% will be from 2 to 18
  + 10 = mean
  + 8 = sd

**List**

* Storage Medium for data 🡸 Can be mixed
* List goes from 0-Number-1 🡸 R does it from 1 to number
* Pulls numbers up until the second number 🡸 z = list(range(1,8)) This will pull 1 to 7

w = list(range(100,111,2)) #by steps of 2

w

**[] Brackets**

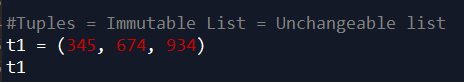
* + Pulls and selects specific in an array
  + len(w) 🡸 Pulls the length of a list
  + w[2] = 63 🡸 You can overwrite a list
  + w[-1] #different behaviour to R
    - ['a', 'b', 'c', 'd', 'e']
    - #0 1 2 3 4
    - #-5 -4 -3 -2 -1

**Slicing – Done inside the [ ] brackets**

* + Taking a subset out of the list
  + 2 : 9 : 2 🡸 Pulls subset of 2 to 8 by two. 🡸 2,4,6,8

**Tuples – Unchangeable Lists**

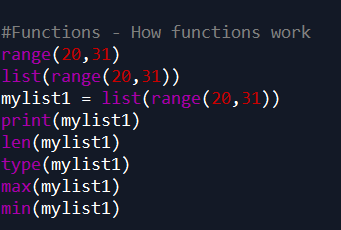
* + Usually done for programming where you don’t want anyone to change a list



**Function**

Like a blender, does something to the input to make an output

Only need to list the arguments you want.



**Packages**

Packages are collections of Python Functions

Library – Directory where packages are stored

Github – Where packages are online.

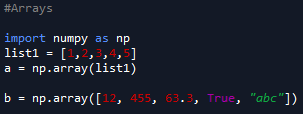
pandas, nympy, seaborn, matplotlib, matplotlib.pyplot, scripy.stats, scipy.stats.stats

* Plotting charts
* Processing large data sets
* Good for data frame

**Install & Run**

Inside of command anaconda 🡺 Conda install numpy or pip install numpy

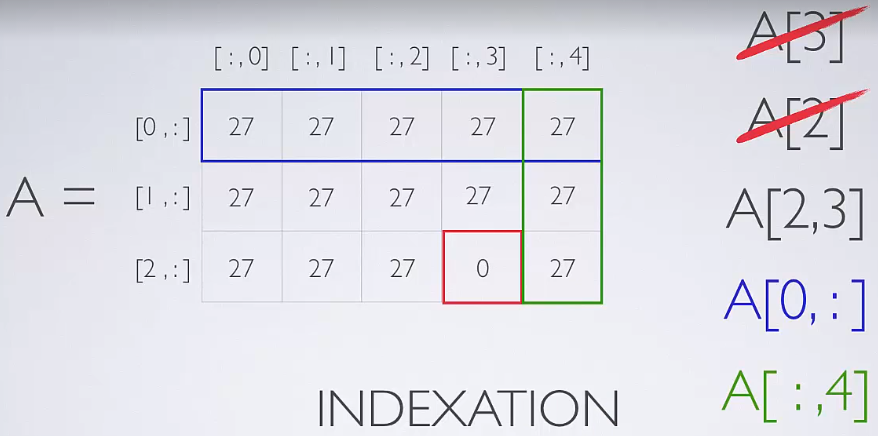
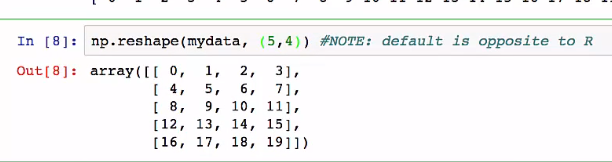
**Arrays**

* + **Numpy –** Uses arrays in Numpy
    - Use the Numpy array
      * A = Np.array(Pass list here)
      * Works like R arrays in numpy
      * ****

**Slicing Arrays**

* + **Numpy –** Uses arrays in Numpy
    - Not making another copy when you slice an array
      * A view of that array
    - So if you edit the new array you sliced, it will change the original which is different from a list

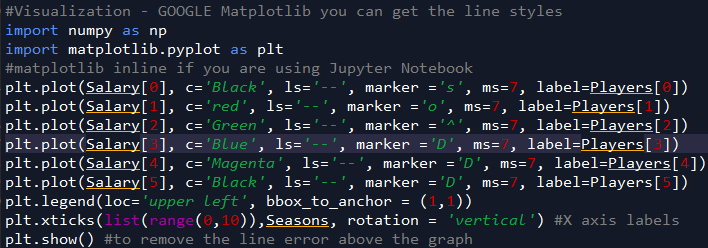
**Matrices**

* + **Indexation**
    - Pulls and selects specific in a matrices 🡸 A[1,2]
    - To pull a whole row or column 🡸 A[1, : ] or A[ : ,5]
    - 
    - Reshaping Matrix
    - 
    - Remember indexation is from 0-# not 1-#

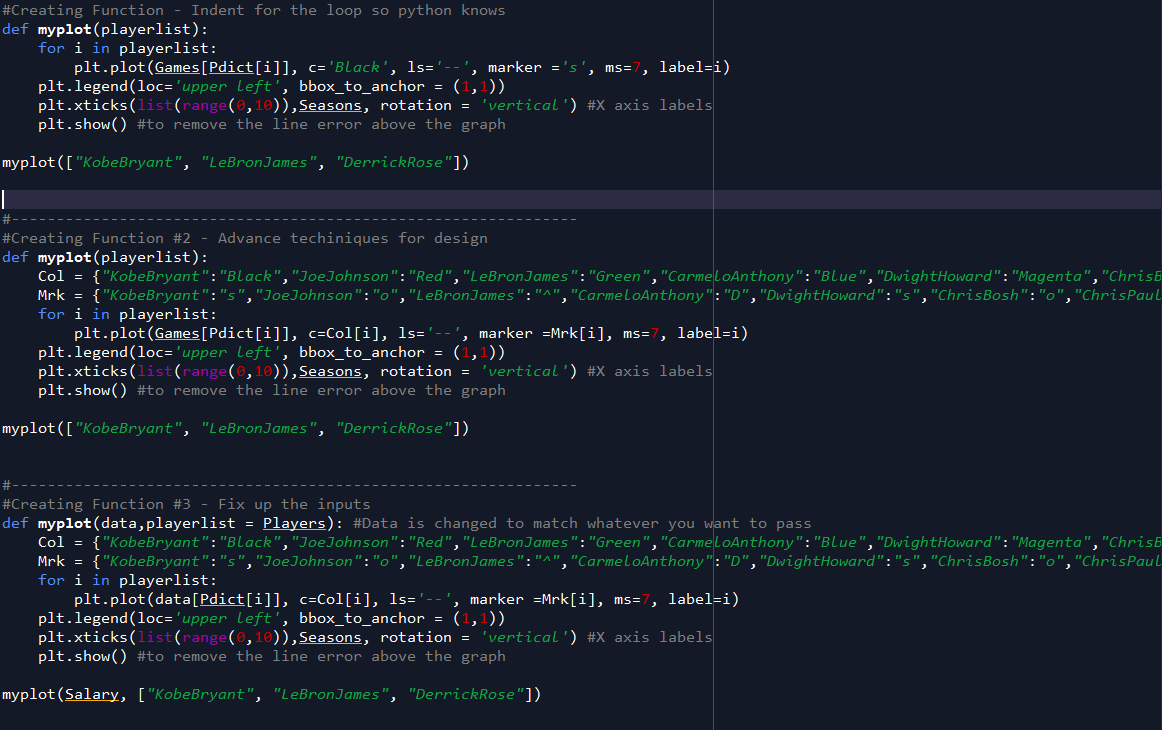
**Dictionaries**

* + - **Ordered by key**
      * Label of each value

**Visualization**

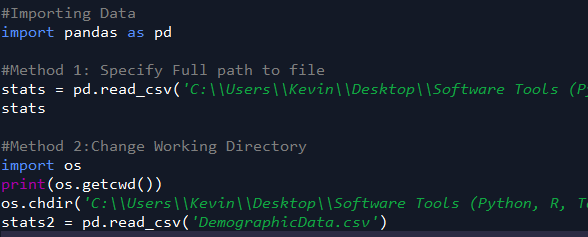
* + Matplotlib
* 

**Creating Functions**

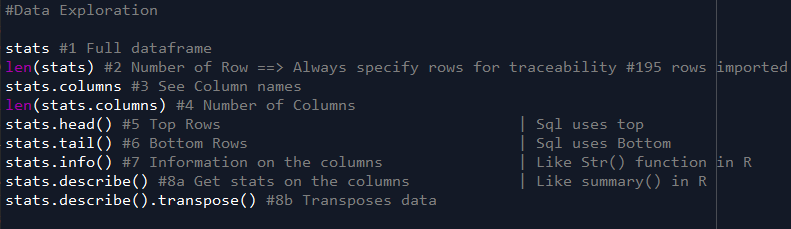
* Def a function and input the data and parameters
  + Data can be salary, points, fg%, etc
  + Parameter can be players or the rows
* 

This creates a function as “myplot” 🡸To use function you then type myplot()

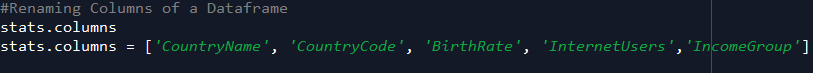
**Importing Data into Python**



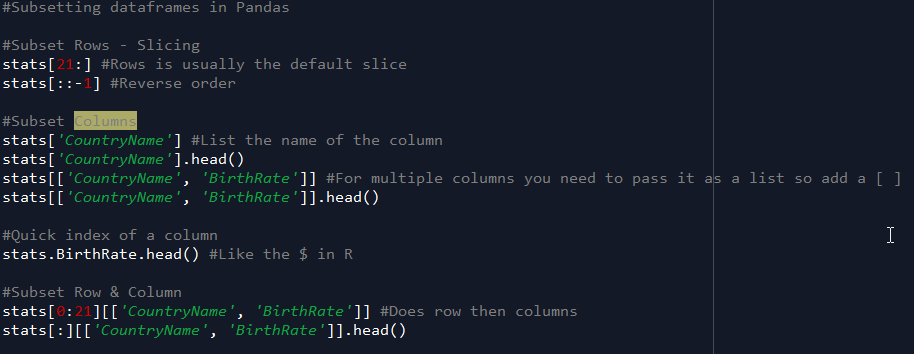
**Exploring Data in Pandas**

****

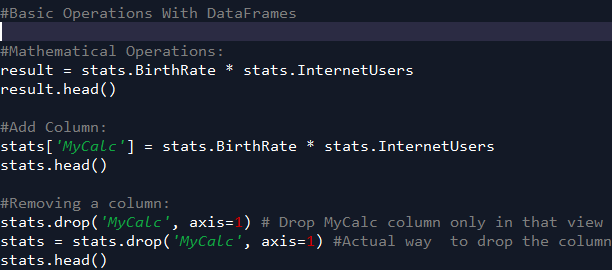
**Renaming Columns**

****

**Subsetting dataframe in Pandas& . is equivalent to $ Sign and the use of it**

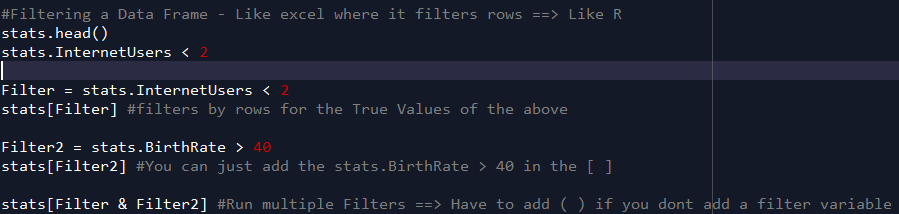
* **Used for pulling data from a column.**
* ****

**Basic Operations & Adding/Removing Columns**

****

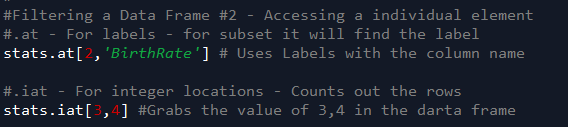
**Filters In Python**

* Have to use & for multiple filters
* Have to use ( ) if you don’t use a variable
* Have to use == for text/string

****

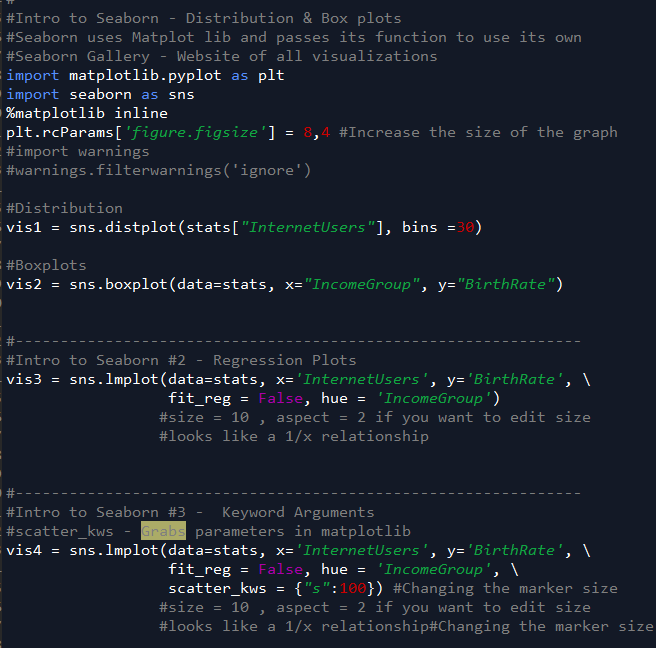
**Filter #2 – Specific elements**

* .at grabs the labels with column name
* .iat COUNTS the actual rows and columns
* Will grab different results for subsets

****

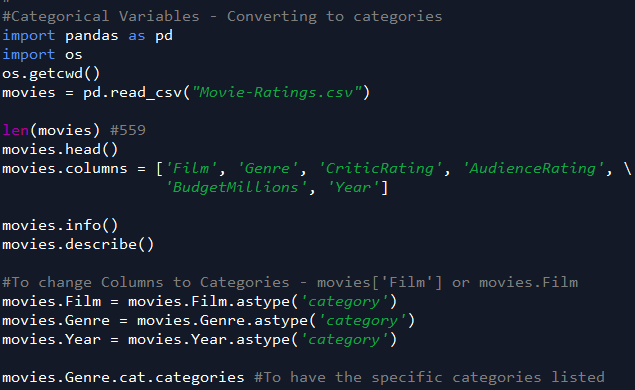
**Seaborn Visualization**

* Facet grid is a grid of multiple charts
* Uses Matplotlib as the base and builds on it



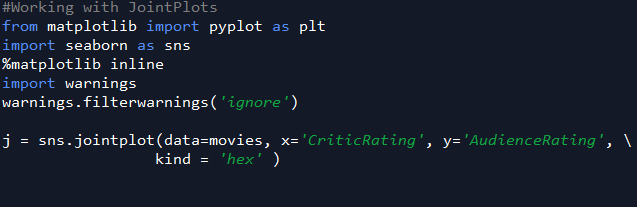
**Categorical Variables**

Changing the type for data 🡸 Year from # to categorical.

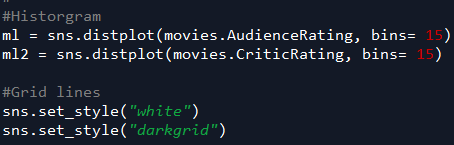


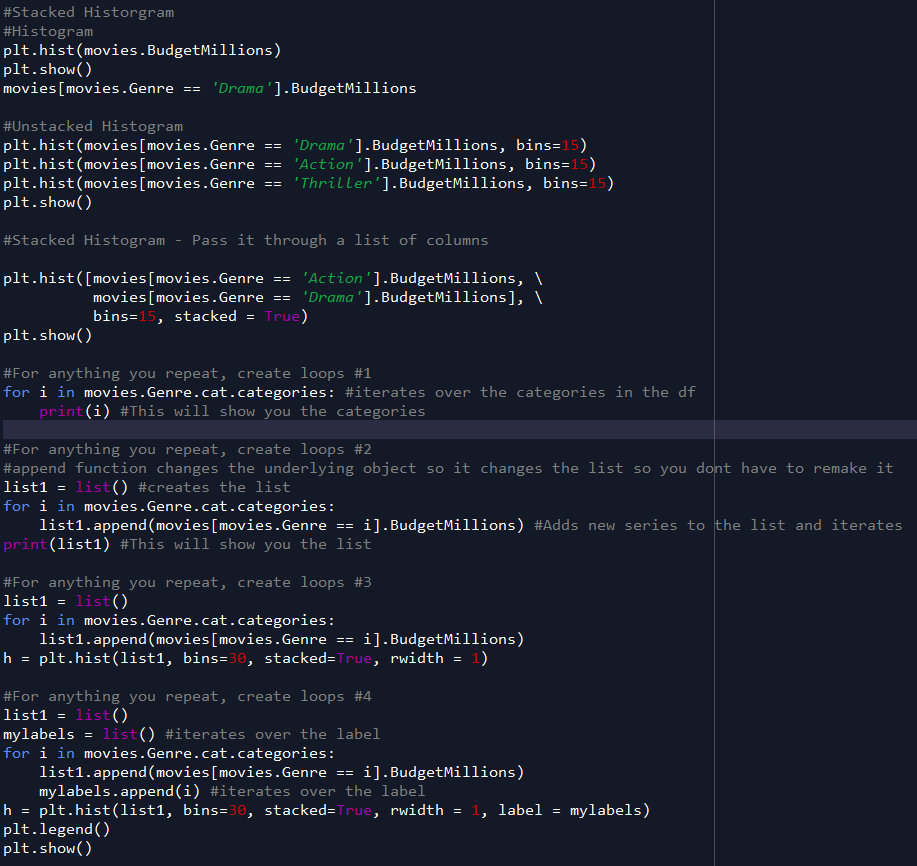
**Joint Plots**

* Scatterplot with cumulative plot

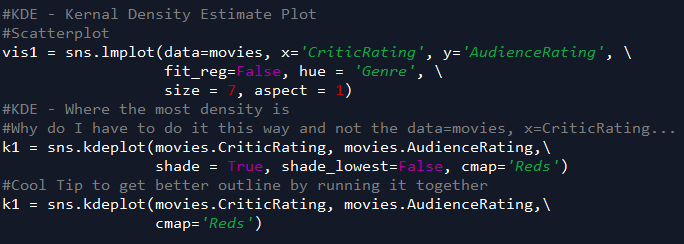


**Histogram & Stacked Histogram Plots**

****

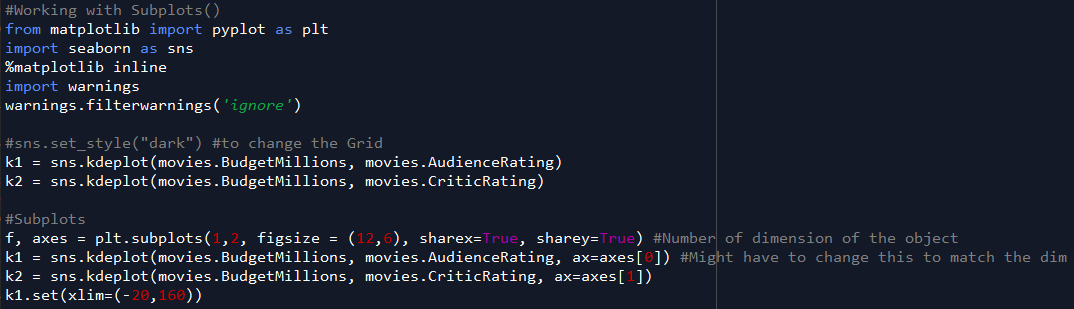
****

**KDE Plots –Kernal Density Estimate Plot**

****

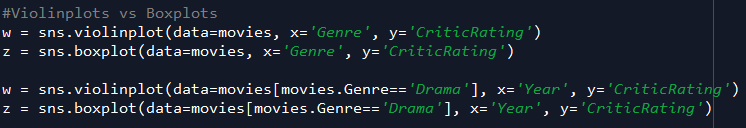
**Subplots()**

* Multiple plots



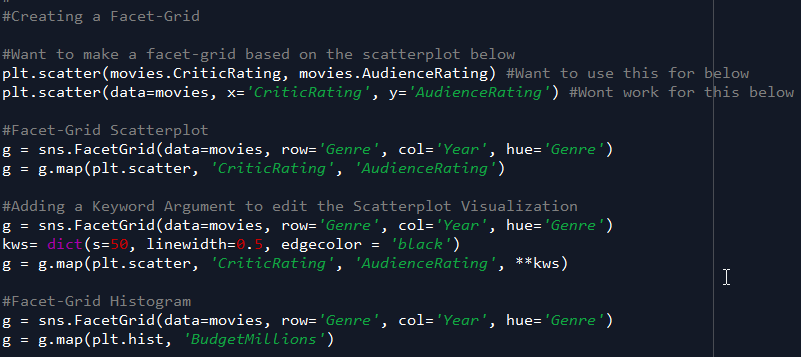
**Violin-plot vs Boxplots**

* Violinplot – the width tells you how many points are in that section
* Boxplots – show the amount of observations by quadrants
  + Smaller the section the more people are in there #density

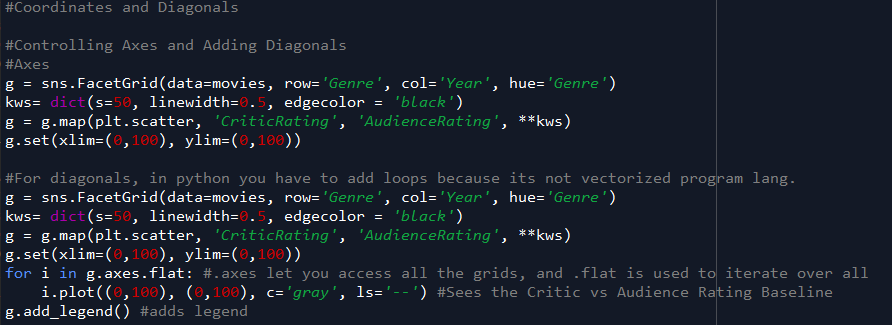
****

**Facet Grid**

* Multiple Graphs – Auto populates of your graphs

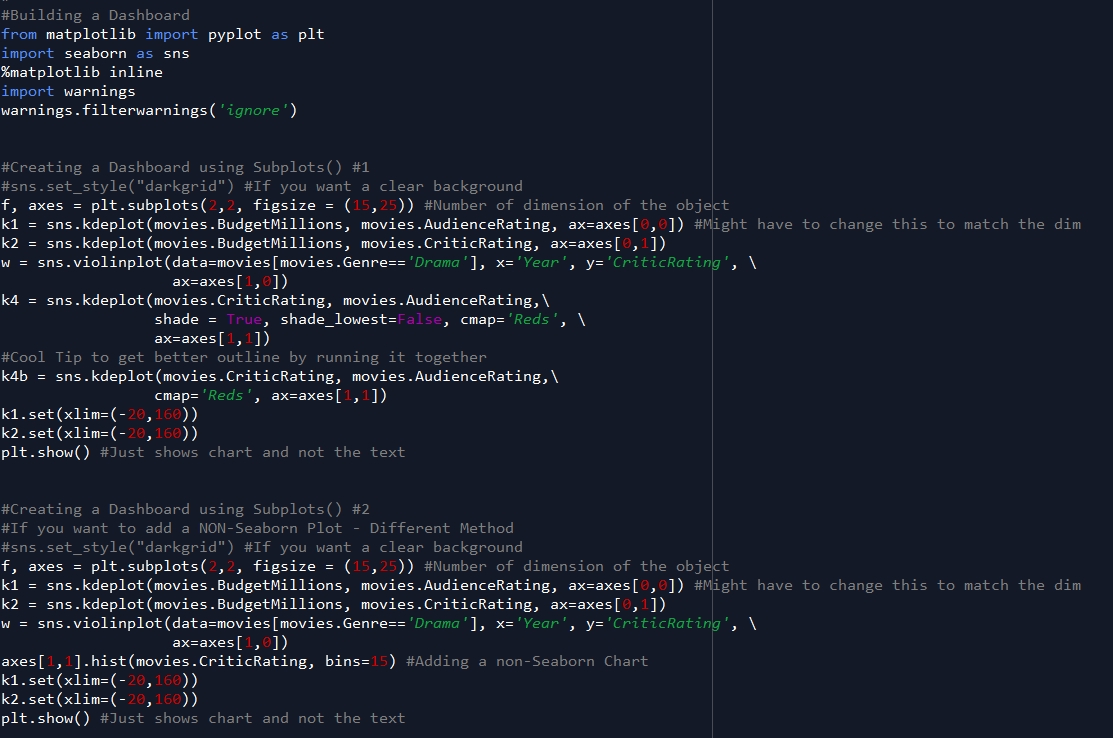


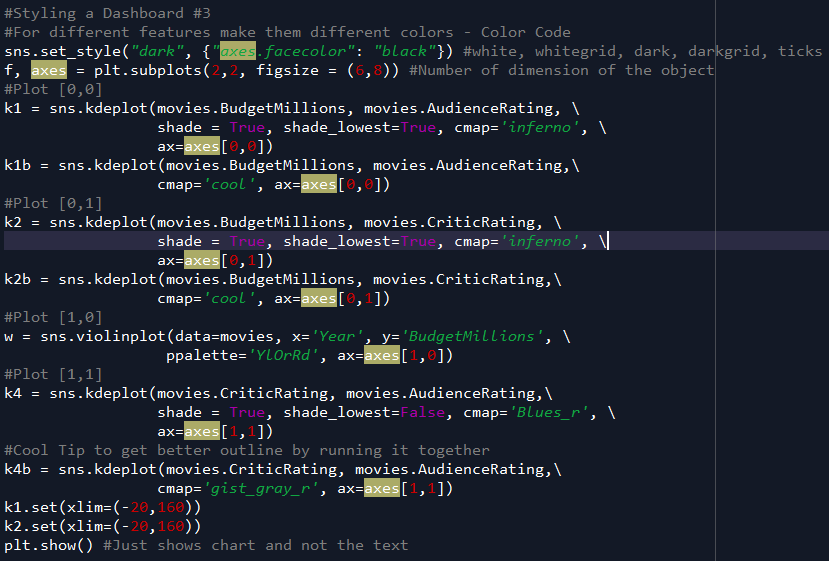
**Coordinates & Diagonals**



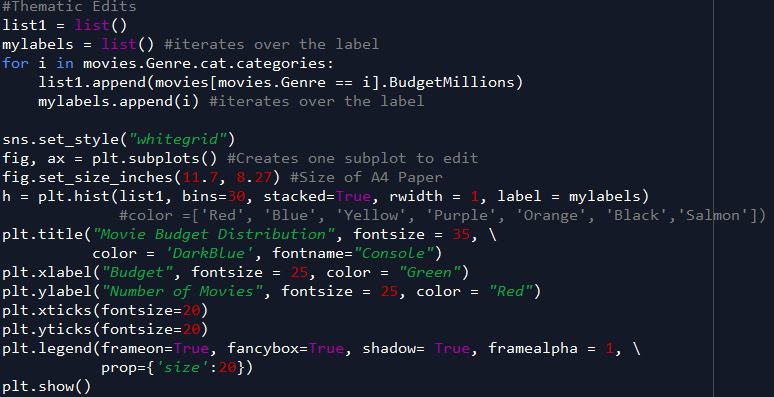
**Building a Dashboard in Python**

* Subplot() to make dashboard
* For different features make them different colors - Color Code



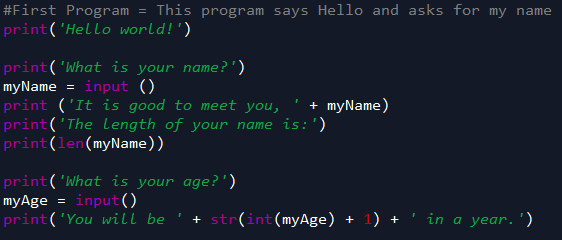
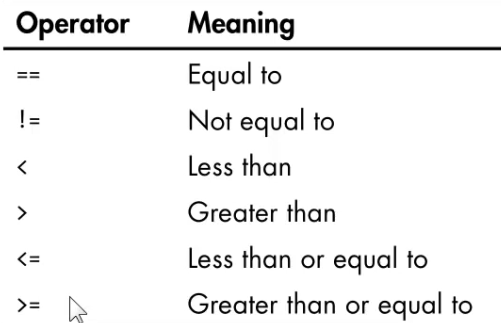
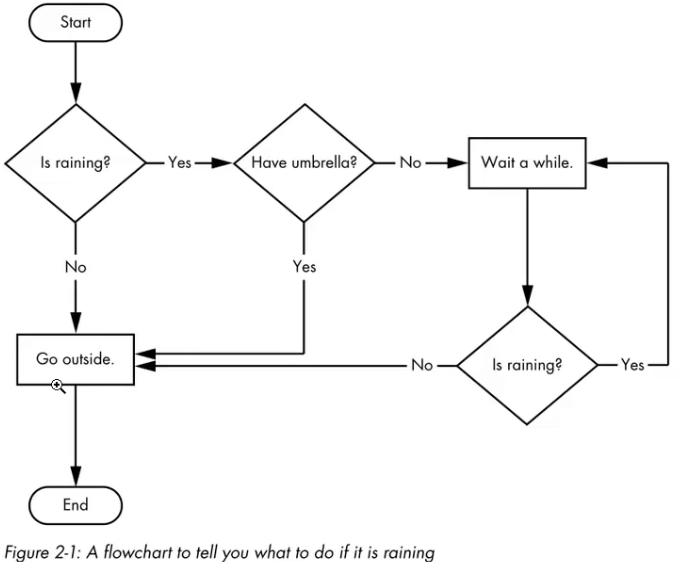
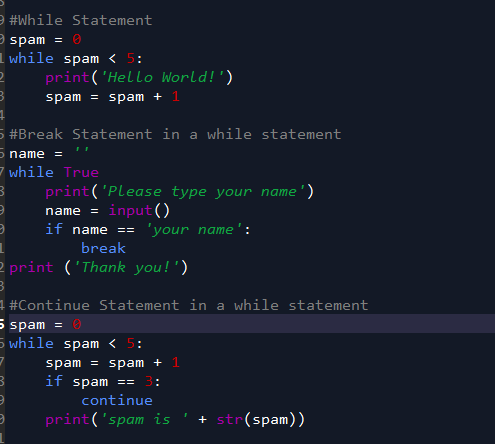
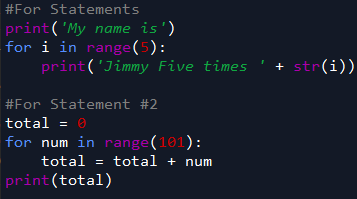
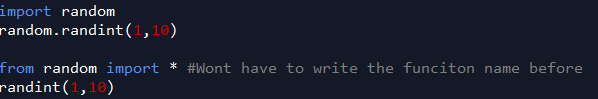
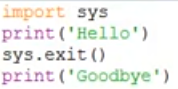
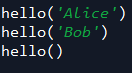
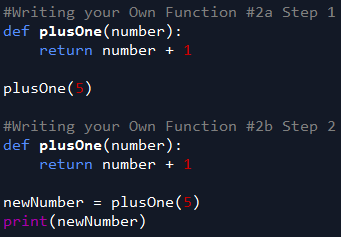
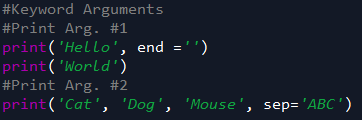
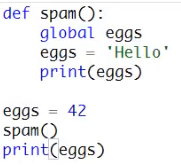
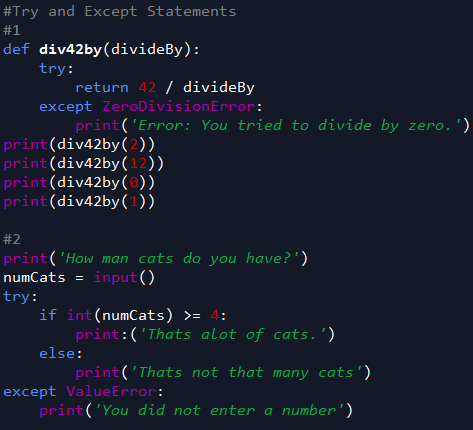
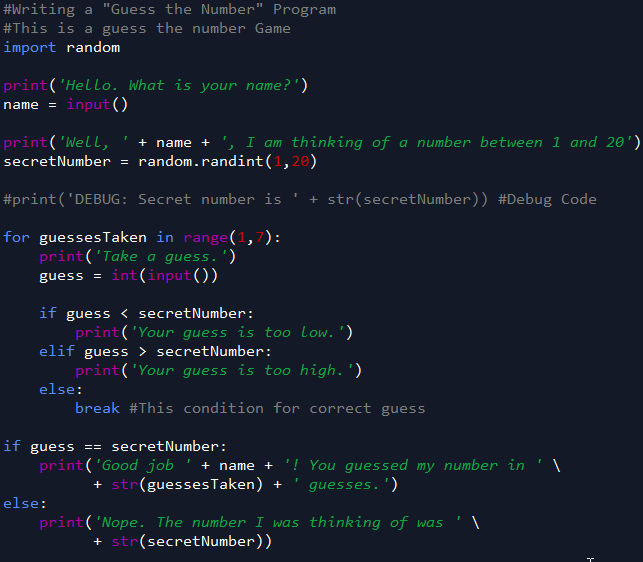
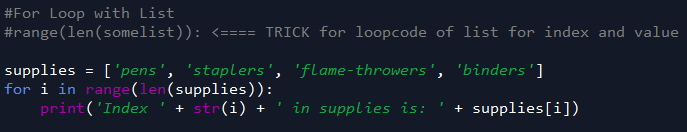
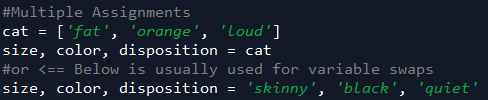
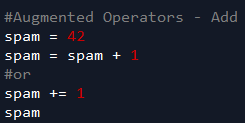
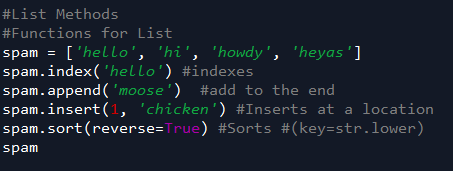
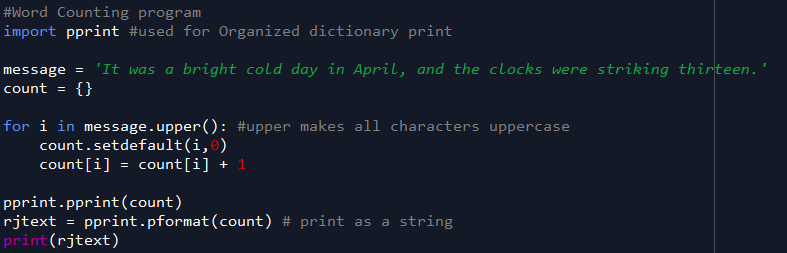
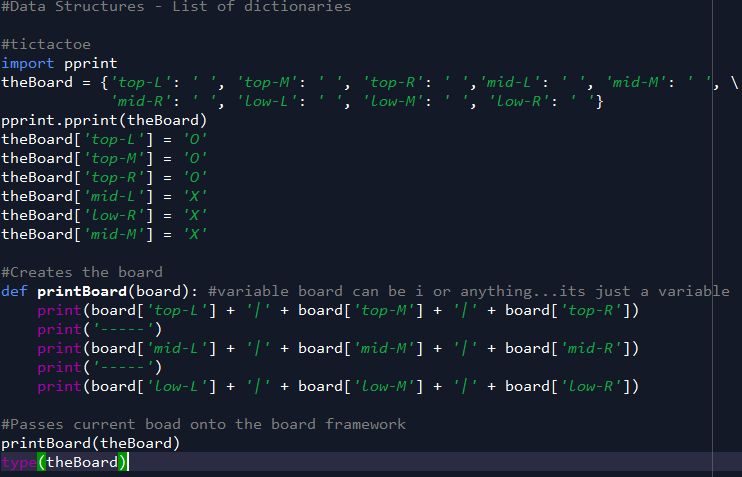
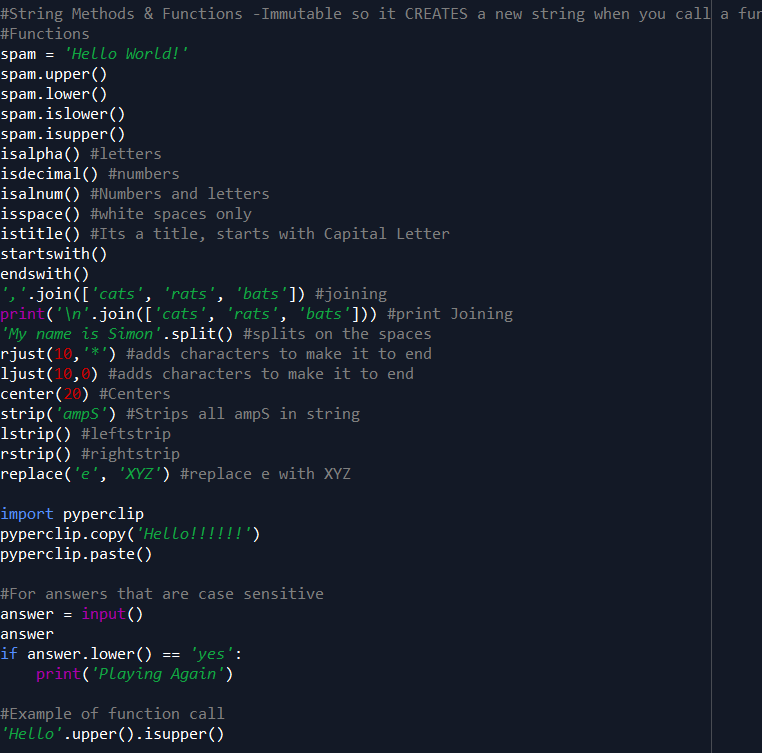
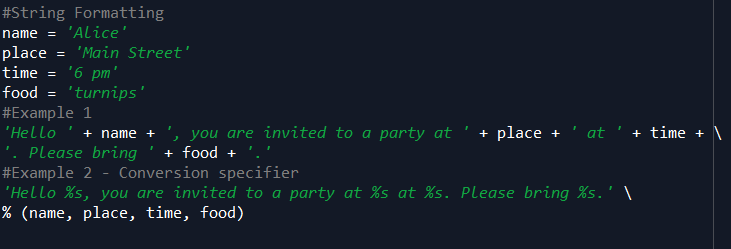
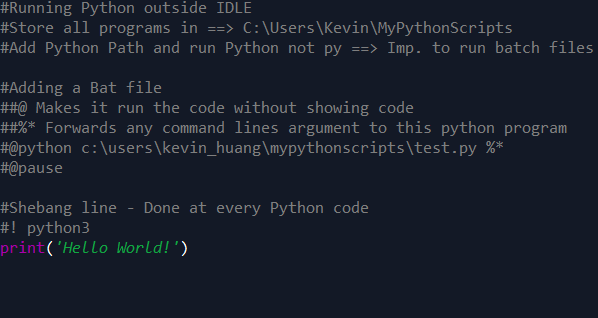
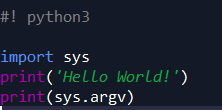
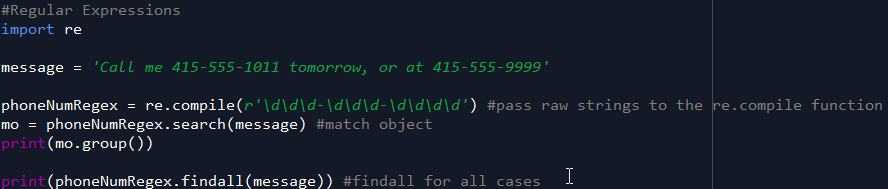
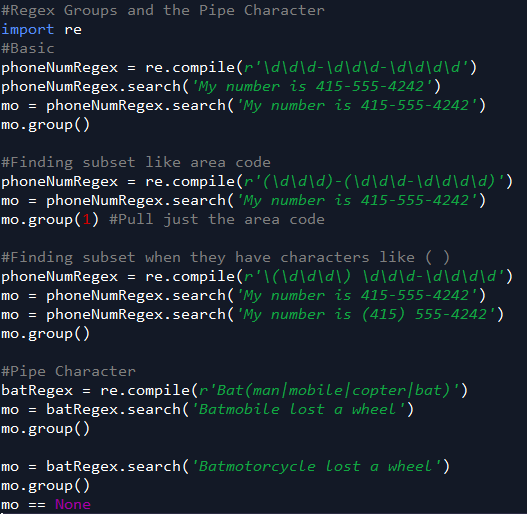
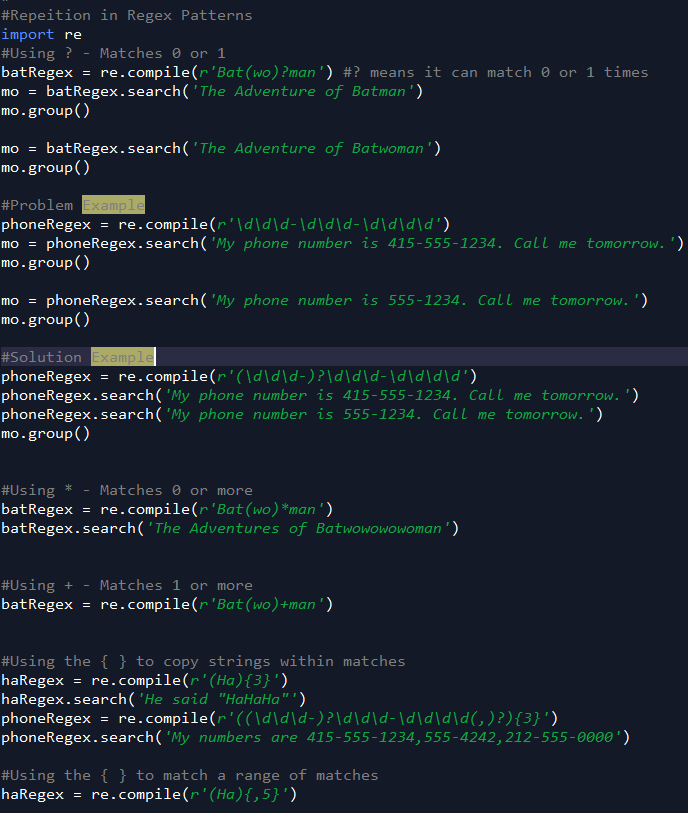
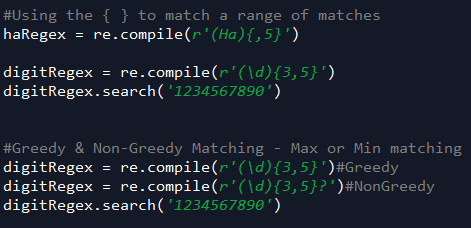
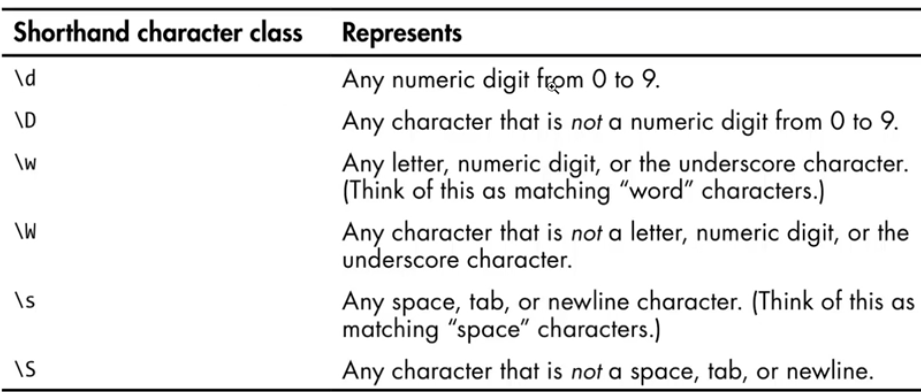
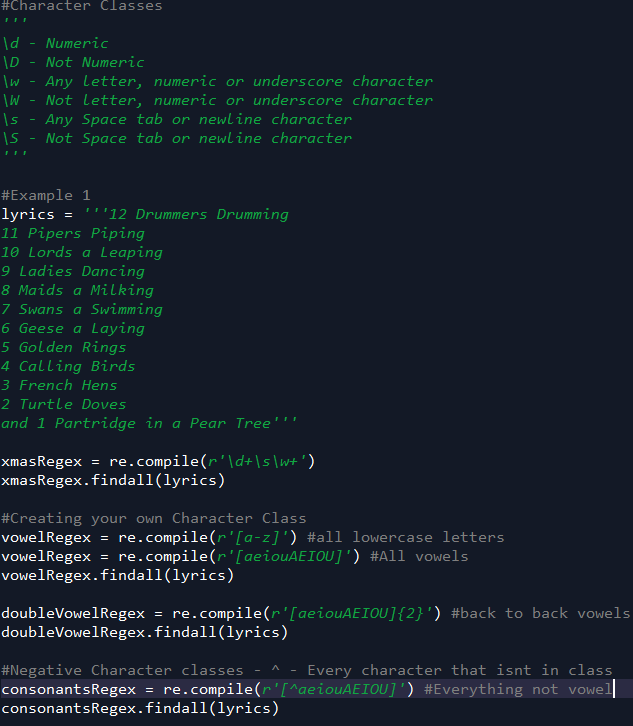
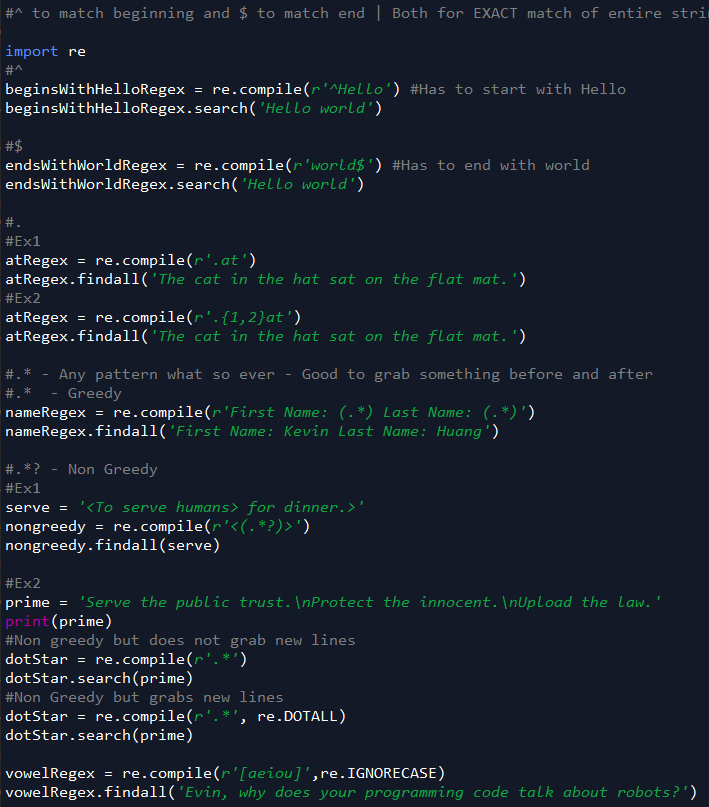
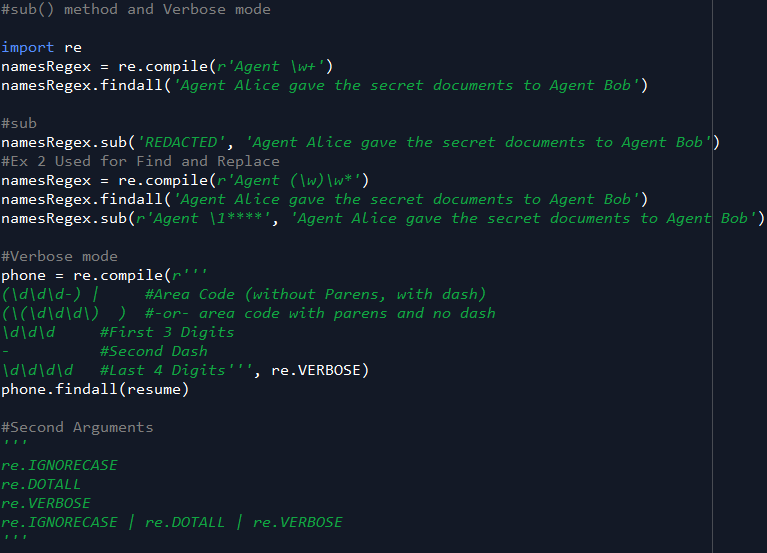
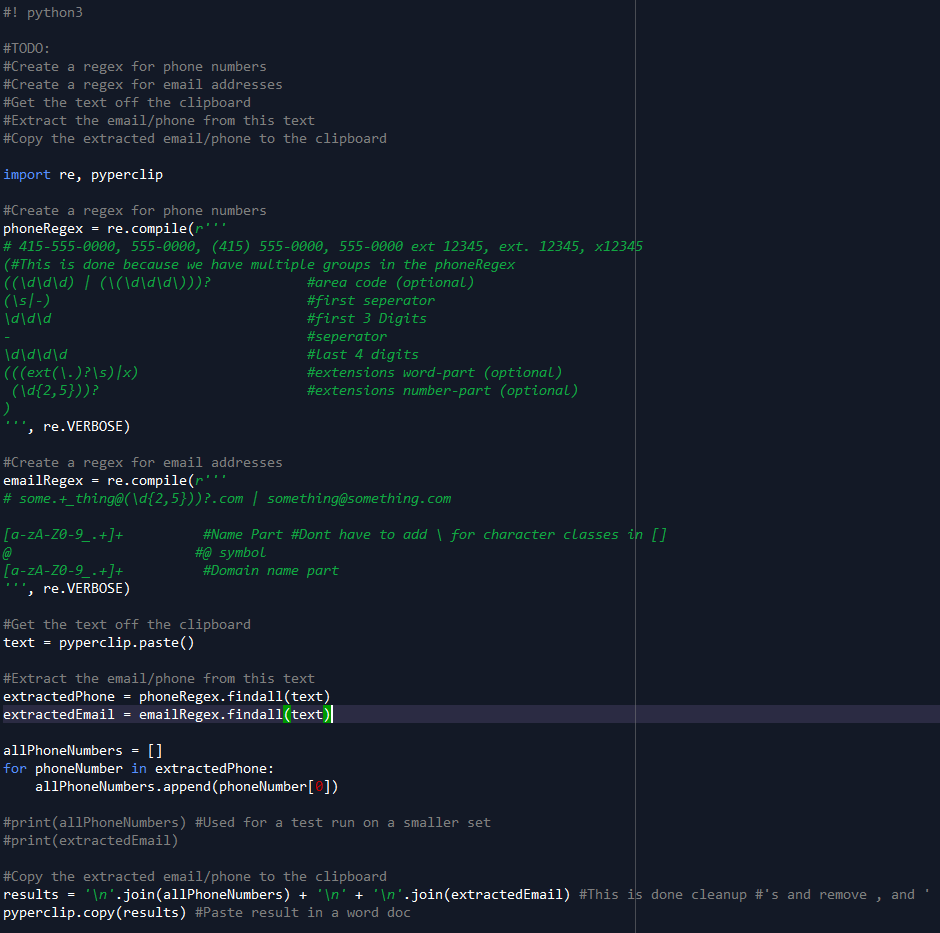
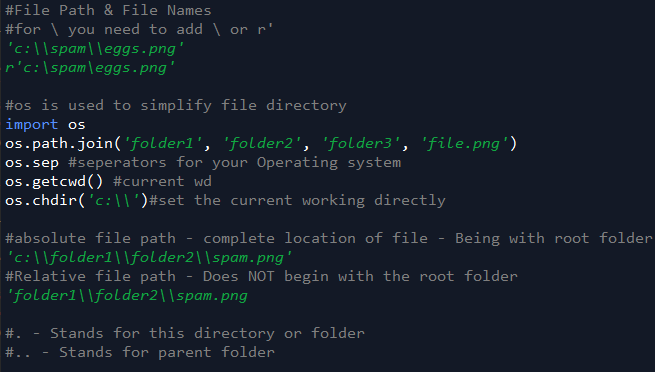
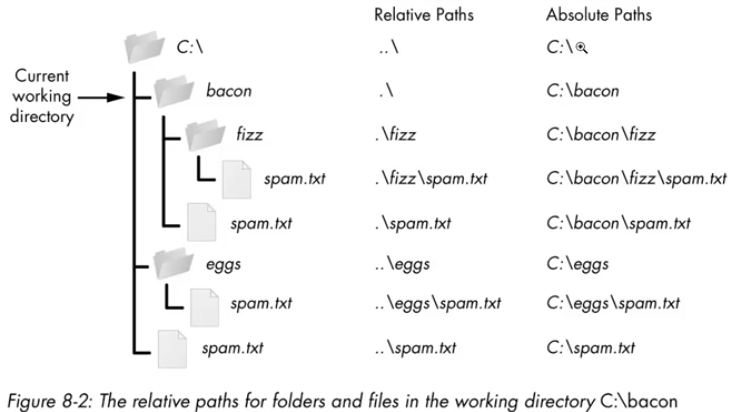
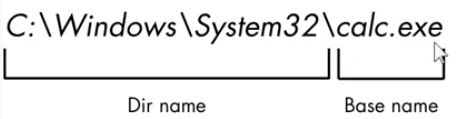
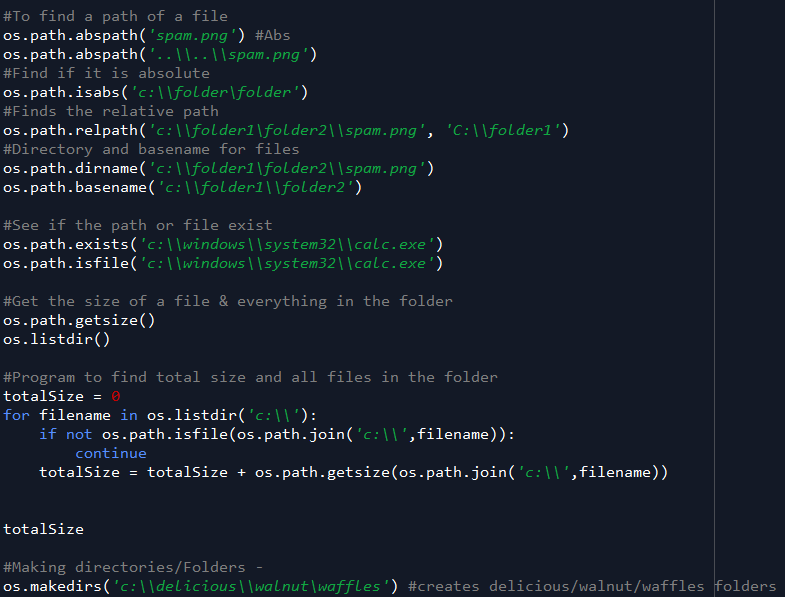
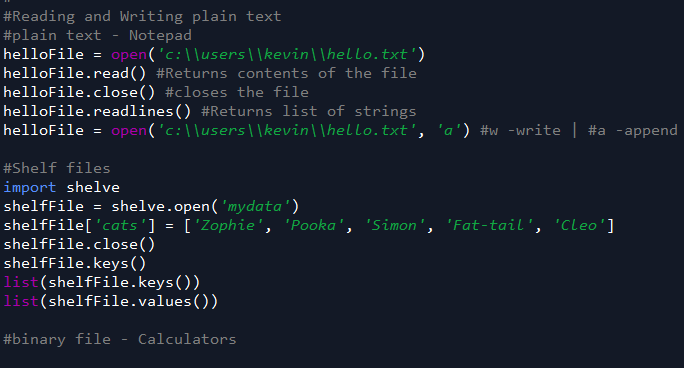
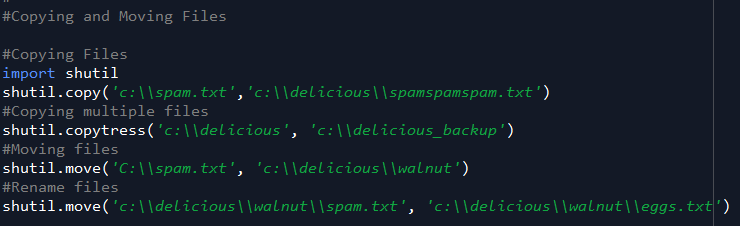
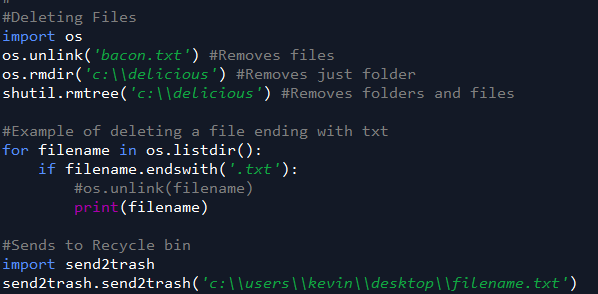
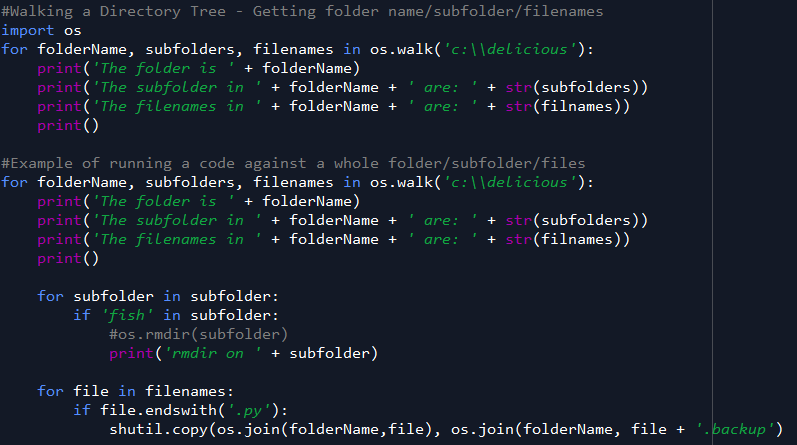
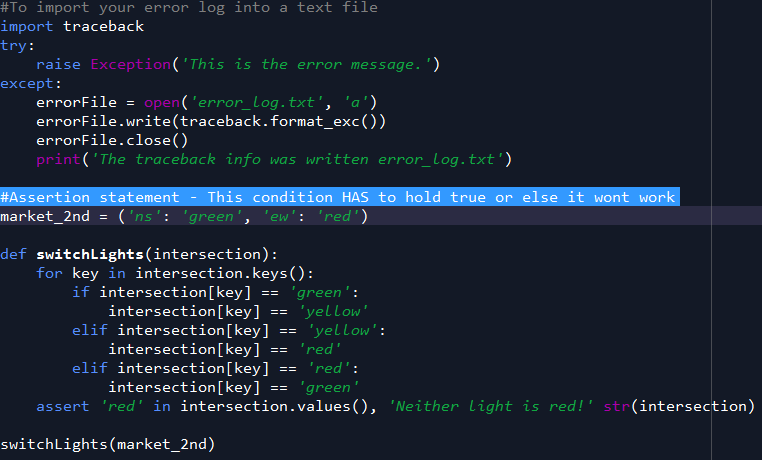
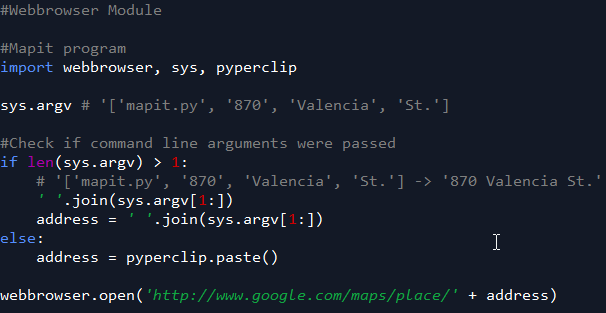
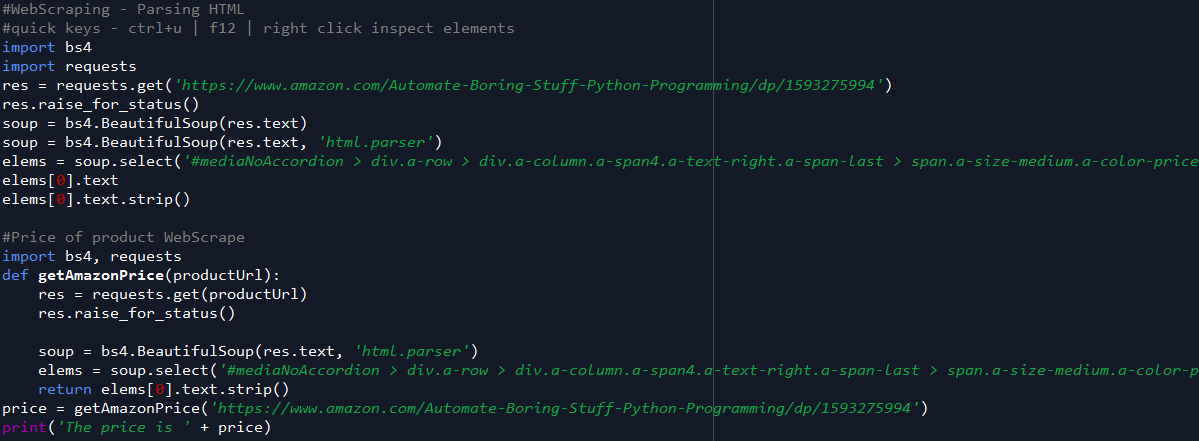
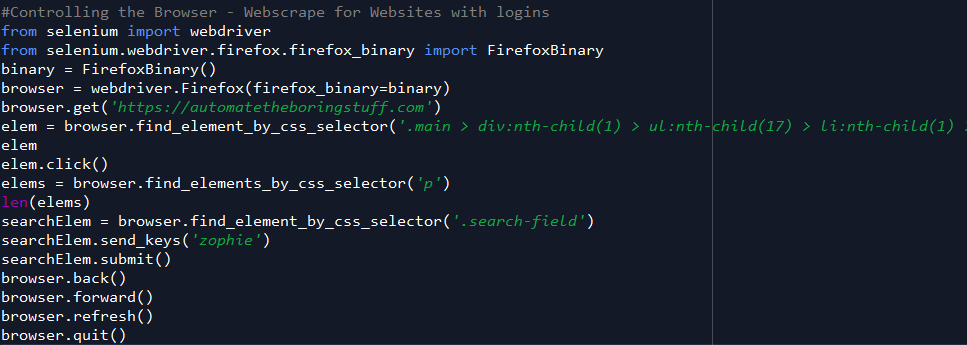
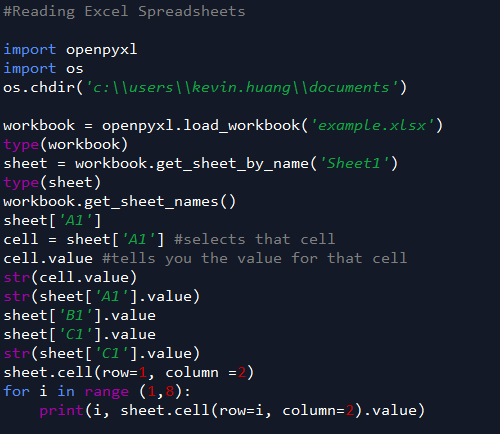
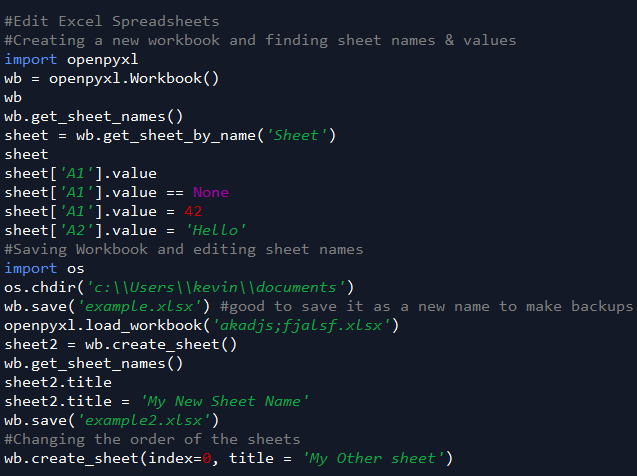
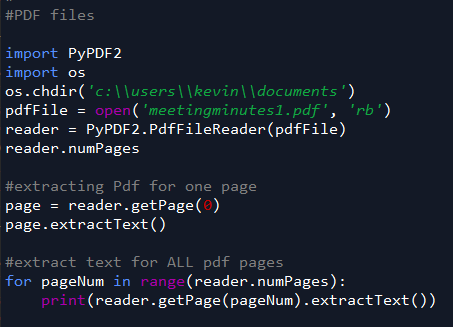
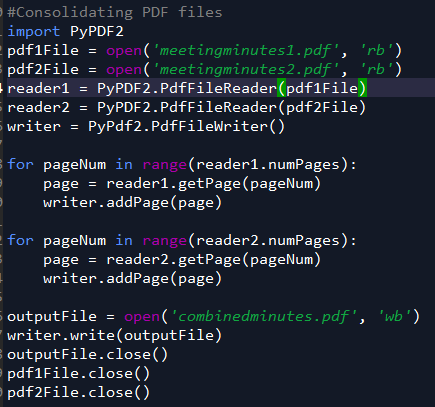
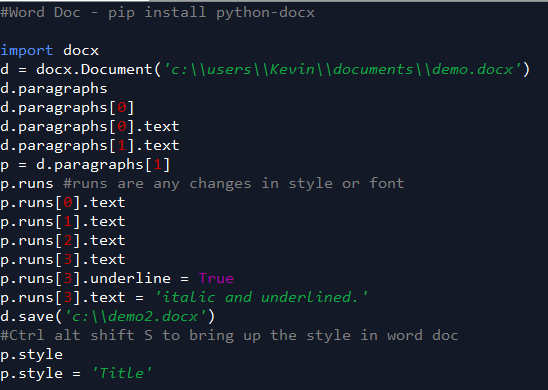
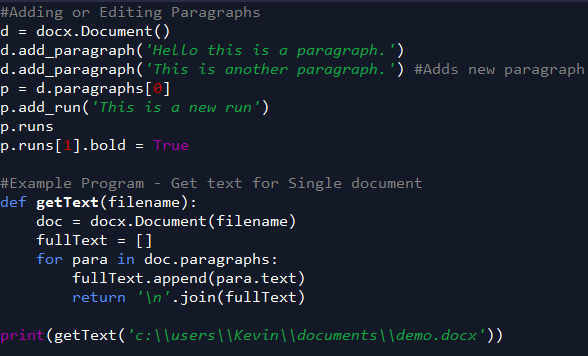
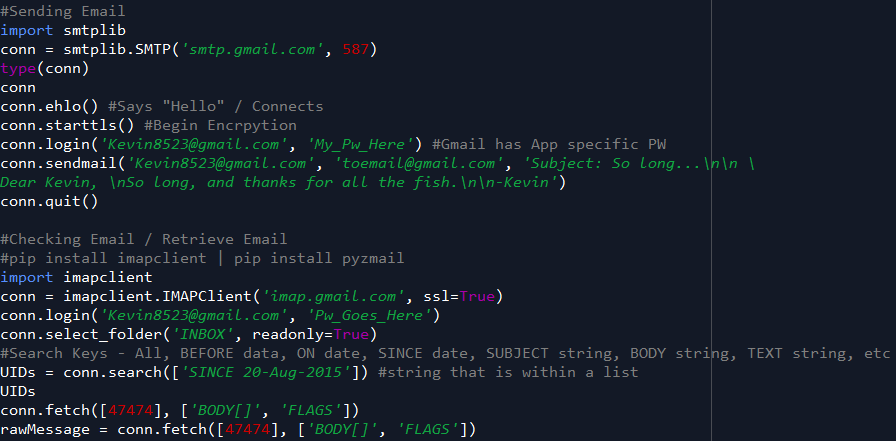
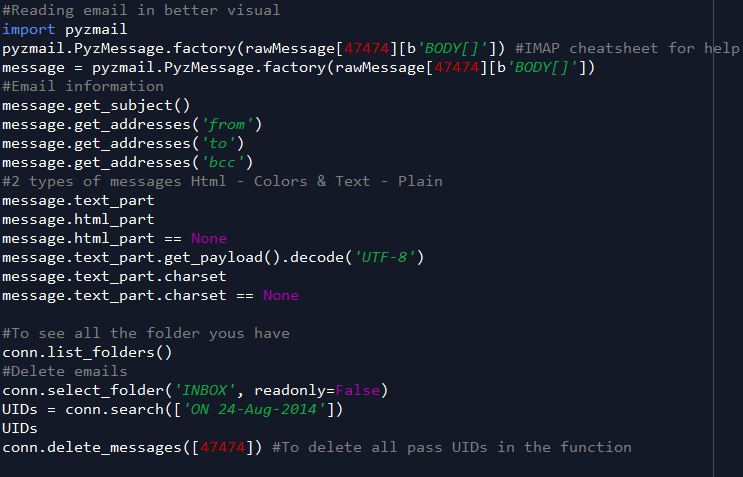
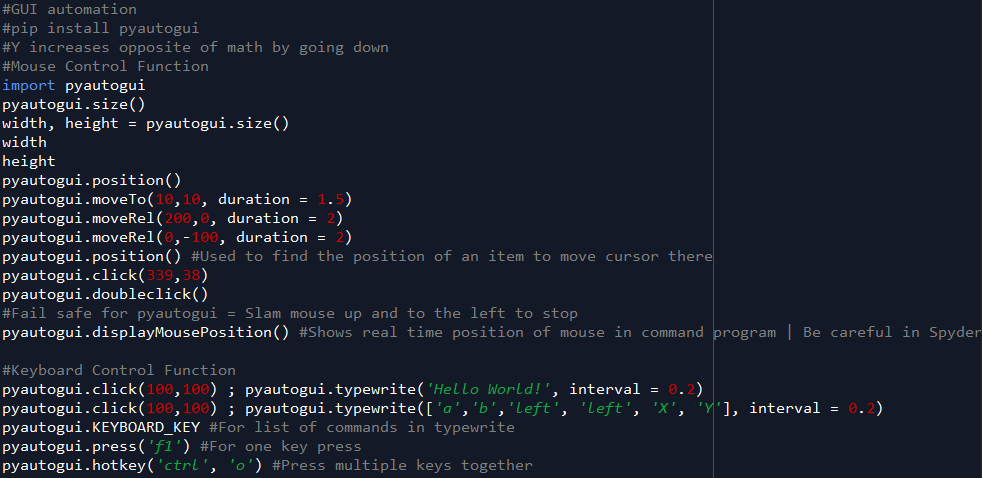
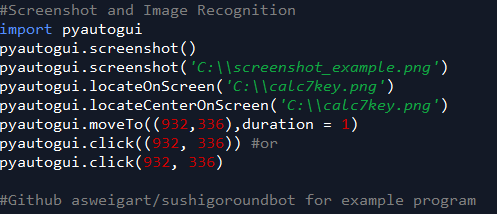
****

**Thematic edits**



applied forecasting techniques such as Exponential Smoothening, ARIMA, ARIMAX, UCM, Kalman Filtering, ARCH/GARCH and other regression based, econometric causal modeling etc

**Automating the Boring Stuff Python**

* Correct Syntax Error
  + Exlian what you are trying to do, not just what you did
  + Specify the point at which the error happens
  + Copy and paste the error message and your code to a pastebin site like
    - Pastebin.com
    - Gist.github.com
  + Explain what you’ve already tried to do to solve yor problem
  + List your version of Python
    - Also what windows
* Basic
  + Expression – Reduce down to a single value
    - Expression = Value + operators
  + Statement – Doesn’t reduce to a single value
    - Code
  + Ctrl + C to stop infinite loop
* Data Types
  + Integers = Ints
  + Floating Point Numbers = Floats
  + Strings = Text
  + Variables = Box that can store values
* Terms
  + Concatenation = Joining
  + String Replication = Multiplying
  + Functions = Mini program in a code
    - Function name followed by paranethesis
  + Arguments = Values passed in functions
* Functions
  + Input()
    - Waits until you put an input into the function
  + Len()
    - Takes a string argument and tells you how many letters is in the function
  + Str()
    - Return String Values
    - Used to convert data types
  + Int()
    - Return Integer Values
    - Used to convert Data Types
* Hello world Program
* 
* Boolean data types
  + Boolean Value
    - True
    - False
    - Integers and string WILL NOT equal eachother
    - Float and strings CAN equal each other
  + Boolean Operators
    - And = both true
    - Or = Either true
    - Not = Opposite
      * Not True = False
      * Not False = True
  + Comparison Operators
    - 
* If/Else/Elif
  + If = A condition statement
    - If true it will run the indented statement
    - If false it will skip the indented code
  + Else = When a condition is false
    - Will run the else statement
  + Elif = Else if Statement
    - Runs as many conditions as you would like
  + Truthy/Falsey
    - 0 or Blank false
    - Everything else is true
    - bool() function
* Flow Chart
  + 
* While
  + Runs when condition is true
  + Break statement – Several different places in a while loop to break/be ininfite
    - Jumps out of a loop
  + Continue statement -
    - Jumps back to the while loop
  + 
  + 
* For Loops
  + Runs for a specified # of times
  + Break and Continue statement can be used in for loops
  + 
* Functions
  + Must import functions first – Must have . to call it
    - Random
      * Random.randint(1,10)
      * 
    - Sys.exit()
      * Exit a program immediately
  + Installing packages
    - Pip install pyperclip
    - 
* Writing your own function
  + Creating your own function is to get rid of duplicate codes
  + De-Duplicating Code
    - Making code more clean and smaller
  + Parameter – Variable inside the function
  + Argument – The value passed in the function call
  + When passing a function
    - Strings – Print
    - Return - number
  + Example 1
    - Parameter is what’s between the parenthesis
      * name
    - Argument is what is passed
      * Alice
      * Bob
    - 
  + Example 2
    - 
* Functions Background information
  + None Value
    - Print Function - Returns as a none value
    - Every function call has a return
  + Keyword Argument
    - 
* Global and local scopes
  + Global scope – Global variable
    - Within the whole code
    - Until Whole code ends
    - Global scope CANT use local variables
  + Local scope – Local variable
    - Created when a function is called
    - Temporary
    - Local scope CAN use Global Variables
  + Rules
    - Global scope CANT use local variables
    - Local scope CAN use Global Variables
    - Local variables CANT use other local variables
    - Can use same name for different variables if they are in different scopes
  + Creating a global scope in a local scope
    - Create global function in variable scope
    - 
* Try and Except Statements
  + Doesn’t know how to divide by 0
  + Value error
  + Used to let the program keep going if there is an error
  + 
* Writing a Guess the number Program
  + Thought Process
    - Import the random function
    - Ask what their name is
    - Want an input function for name
      * Store the function
    - Print a response saying hello and asking the question of random number
    - Create a secret-number function for them to guess
    - Now we are creating the loop for them to guess
      * Print the take the guess
      * Create a guess input for the users
        + Make sure to convert to integer
      * Create if scenario for too low and too high
      * Create scenario if they guess right or wrong too many times
        + Print the good job you got it in X guesses
        + Print the else statement if they didn’t get it right
    - 
* List
  + Value that contain values
    - Contains multiple values in a sequence
    - List values = items
    - Usually starts with [ ]
    - Can contain lists of lists
    - Negative index – goes backward from your list
    - List is Mutable value
      * Uses as a reference
  + Index
    - Evaluates to single value
  + Slice
    - Gets multiple values
    - Spam[1:3]
    - Evaluates to new list value
    - Slice shortcut
    - Spam [ : 3 ]
      * Beginning to 2
    - Spam [ 3 : ]
      * 3 up to last one but not last one
  + del statement
    - To delete things within a list
* For Loops with list
  + 
* Multiple Assignments
  + 
* Augmented Operators
  + 
* List Methods
  + Methods are like functions
  + Methods for lists
    - Index
      * For duplicates it will return the first value it sees
    - Append
      * Will add it at the end
    - Insert
      * Will insert in that location, requires 2 arguments
    - Remove
      * Removes value
      * Will remove the first instance
      * Can remove by string 🡺 ‘bat’
    - Sort
      * Sorts Uppercase first then lowercase after
      * reverse=True
        + Reverse order
      * True alphabetical order
        + key=str.lower
      * 
* Similarities between list and string
  + List – mutable data type
    - Things can be added and removed
  + String = Immutable
    - Things cannot be changed
  + Difference between immutable and mutable comes up with “references”
    - For lists
      * When you make changes to spam or cheese it will change BOTH references
    - For Strings
      * When you make changes to either it will ONLY make changes to that string
    - Mutable value – list
      * references
    - Immutable values – Strings & Tuples
  + Deep copy
    - Makes a BRAND new copy of the list
      * NOT a reference
  + \ is to add codes
* Dictionary
  + Collection of many values
  + { }
  + Dictionaries are unordered
  + Mutable
  + Methods
    - Keys() – Lists the Dictionary item
    - Values() – List the dictionary values
    - Items() – Shows both
    - Get() – Grabs data, if it doesn’t exists it will input something you listed
    - Setdefault() – Sets a default if it does not exists it will call that
    - Upper() – Makes all uppercase letters for string
    - 
  + Triple quotes is to make it a string across multiple lines
    - ‘’’
    - ‘’’
  + Pprint module
    - For nicer formatting
    - 
* Data Structures
  + List of dictionaries
  + Type function
    - Tells you the type of function
  + 
* Advance string syntax
  + \’ to add ‘ in a string
  + \n for a new line
  + r’ – raw string
    - useful for many backslashes
  + ‘’’ or “””
    - Multi line code
* String Methods
  + Strings are immutable
  + Functions
    - Upper()
    - Lower()
    - Isupper()
    - Islower()
    - Isalpha()
    - Isalnum()
    - Isdecimal()
    - Isspace()
    - Istitle
    - Startswith()
    - Endswith()
    - 
* String formatting
  + %s
    - Conversion specifiers
    - 
* Running Python outside IDLE
  + Add the Python PATH in systems
    - Important to add this so it reads the batch file
    - Have to find the Python.exe name
      * Python.Exe
  + Shebang line
    - #! python3
    - Win key + R to use the run dialog
      * Cmd
      * 
  + Batch Files
    - Known as Shell Scripts
    - Text file with commands saved to file that ends in .bat
  + Run file by just inputting the file in the command prompt once the PATH is set up
    - If path is not set you can list the area
  + 
  + Arguments
    - Can add it after the running of the code
    - Import sys and print sys.argv
    - 
* Regular Expressions
  + Mini language Done for text pattern recognition
    - Finding a phone number
  + The Re Module
    - 1. Compile what you’re searching for
    - 2. Create where you are searching for it
    - 3. Create a match object where you will store what you find from the search
    - \d is for digits
    - 
* Reggex Groups and the Pipe Character
  + Reggex Groups
    - Adding parenthesis to group specific parts of match object and finding just the parenthesis info
  + Pipe Character
    - Finding something with a prefix word
    - If it doesn’t find anything it will be None
    - 
* Repeition in Regex Patterns
  + ? for match objects containing 0 or 1 of a prefix
    - if you want to find a ? just add a \?
  + \* match 0 or more times
  + + match 1 or more
  + \ For literal match for the above regex patterns
  + { } to copy over strings for matches
  + 
* Greedy vs Non Greedy Match
  + Greedy – Default
    - Grabs the most possible
  + Non-greedy – Add a ? after the { } 🡺 {}?
    - Grabs the minimum
    - 
* Findall() method
  + Returns a list of strings if there is no group
  + If there are groups it sends a list of tuples of strings
  + 
  + Character classes
    - 
    - 
* Dot-Star and Caret/Dollar Character
  + ^
    - Beginning match
  + $
    - End Match
  + ^ and $
    - EXACT match
  + .
    - To get matches of anything with the letters after
  + .\*
    - Any pattern match to grab
    - Greedy
  + .\*?
    - Not Greedy
  + 
* Sub() Method and Verbose Mode
  + Sub() method used to find and replace
  + Verbose mode for easier reading of code
  + 
* Regex Program phone and email scraper
  + 
* Files
  + Root Folder
    - C:\
  + File Extension
    - Tells you what type of file it is
      * Jpg
  + Folder directories
    - Parent and relative working directory
    - 
    - . for current
    - .. for parent folder
  + 
  + Directly & Base name
    - 
    - 
* Reading and Writing Plain Text
  + 
* Copying and Moving Files
  + 
* Deleting Files
  + 
* Walking a directory tree
  + 
* Debugging
  + Raising your own exception
    - A rule you need to follow in your code
  + The error message is a traceback
    - Gives you the line of code that causes error
    - Also where the code is called
    - 
  + Assertion method
    - Programmer errors
    - Assertion statement - This condition HAS to hold true or else it wont work
    - 
* Logging
  + Great way to understand what’s happening in what order
  + Record of custom methods
  + Breadcrumb trail for basic config
* Using the debugger
  + Execute the program one line at a time
  + Make sure these are checked
    - Stack
    - Source
    - Locals
    - Globals
  + Over – Step over the current line of code and pauses to the enxt one
  + Step Button – step into a function call
  + Out button – step out of the current function you are in
  + Go button – continue the program until the next breakpoint or the end of the program
  + Quit button – terminate the program
  + Breakpoint – right click and set breakpoint
* Web browser Function
  + 
* Request Module
  + 
* WebScraping
  + Right click 🡺 View page source
  + F12
    - Right click inspect elements to get specific elements
  + 
* Webscrape Controlling the browser
  + 
* Reading Excel Spreadsheets
  + 
* 
* PDF Files
  + 
  + 
  + 
  + 
* Sending Email
  + 
  + 
* GUI Automation
  + 
  + 

=============================================================================================

Overview Summary

* Functions
  + Int() – Changes to integer
  + Str() – Changes to string
  + List() – Changes to list
  + Range() - Creates a list like function
* Statement type function
  + In 🡺 Find if something is in something
* Quick Keys in python - Windows/Preference->Keys
  + Ctrl + D
    - Delete line
  + Ct
* Fundamentals
  + For loop is to iterate over a list of things