Python Tutorial

1. Google Collab: https://colab.research.google.com
   - Cloud solution for all your python and programming needs.
   - Recommend you start with: https://colab.research.google.com/notebooks/intro.ipynb#

2. Anaconda: https://www.anaconda.com/distribution/
   - Alternative to Collab
   - Runs Python on your own computer.
   - Python IDE
   - Download Python 3+.
   - Python 2 has better support, but will no longer be supported after 2020.
   - Jupyter
     - Integrated app with Anaconda
     - You must use Jupyter Notebook for this course
     - You will be submitting notebooks for all your project assignments
     - After installing Anaconda simply run Jupyter
     - Then create a notebook: https://data36.com/how-to-use-jupyter-notebook-basics-for-beginners/

3. Python Language: https://www.w3schools.com/python/default.asp
    - Advantages:
        - Readable
        - New lines to complete a command (no semicolons)
        - Indentation and whitespace to define loops, functions and classes
        - In-line compilation
        - Small set of reserved words
        - Open source. HUGE community support especially for ML
    - #comment
    - Variables
        - Var must start with a letter
        - Alphanumeric
        - Case sensitive
        - Don't need type defs
        - Types: int, float, complex (uses j), str
        - **type**(var) gives type
        - casting (force type): **int(), float(),str(), complex()**
    - string functions:
        ```
        a = "Hello, World!"
        a[2:5]
        len(a)
        a.strip('!')
        a.lower()
        a.upper()
        a.replace()
        a.split(' ')
        ```
    - Input/output:
        ```
        print("Enter your name:")
        x = input()
        print("Hello, " + x)
        ```
    - Arithmetic Operators
        ```
        x + y
        x - y
        x * y
        x / y
        x % y
        x ** y
        x // y          #floor division
        x = 5           #x = 5
        x += 3          #x = x + 3
        x -= 3          #x = x - 3
        x *= 3          #x = x * 3
        x /= 3          #x = x / 3
        x %= 3          #x = x % 3
        x //= 3         #x = x // 3
        x **= 3         #x = x ** 3
        x &= 3          #x = x & 3
        x |= 3          #x = x | 3
        x ^= 3          #x = x ^ 3
        ```

```
x >>= 3  #x = x >> 3
x <<= 3  # x = x << 3
```

- o Logical Operators

```
x == y
x != y
x > y
x < y
x >= y
x <= y
x < 5 and  x < 10
x < 5 or x < 4
not(x < 5 and x < 10)
x is y            #x is the same object as y
x is not y
```

- o Bitwise Operators

```
x &  y
x | y
x ^        y       #XOR
~x                 #NOT
x<<1
x>>3
```

- o Containers (Arrays)
  - List
    - Ordered and changeable.
    - Allows duplicate members.
    - Enclosed by [ ]
    - Functions:

```
thislist = ["apple", "banana", "cherry"]
len(thislist)
thislist.append("orange")
thislist.clear()
x = thislist.copy()
thislist.count("cherry")
del thislist[0]
enumerate(thislist)
thislist.extend(anotherlist)
thislist.index("cherry")
thislist.insert(1, "orange")
thislist.pop(1)
thislist.remove("banana")
thislist.reverse()
thislist.sort()
```

    - **Indexing:**

```
nums = list(range(5))    #nums = [0,1,2,3,4]
nums[-1]                  #[4]
nums[0]                   #[0]
nums[2:]                  #[2,3,4]
nums[:2]                  #[0,1]
nums[:]                   #[0,1,2,3,4]
```

```
        nums[:-1]                    #[0,1,2,3]
        nums[2:4]                    #[2,3]
```
- Tuple
  - ordered and unchangeable.
  - Allows duplicate members.
  - Enclosed by ( )
- Set
  - unordered and unindexed and changeable.
  - No duplicate members.
  - Enclosed by { }
  - Functions:
    - `add()`   Adds an element to the set
    - `clear()`   Removes all the elements from the set
    - `copy()`   Returns a copy of the set
    - `difference()`   Returns a set containing the difference between two or more sets
    - `difference_update()`   Removes the items in this set that are also included in another, specified set
    - `discard()`   Remove the specified item
    - `intersection()`   Returns a set, that is the intersection of two other sets
    - `intersection_update()`   Removes the items in this set that are not present in other, specified set(s)
    - `isdisjoint()`   Returns whether two sets have a intersection or not
    - `issubset()`   Returns whether another set contains this set or not
    - `issuperset()`   Returns whether this set contains another set or not
    - `pop()`   Removes an element from the set
    - `remove()`   Removes the specified element
    - `symmetric_difference()`   Returns a set with the symmetric differences of two sets
    - `symmetric_difference_update()` inserts the symmetric differences from this set and another
    - `union()`   Return a set containing the union of sets
    - `update()`   Update the set with the union of this set and others
- Dictionary
  - unordered, changeable and indexed.
  - No duplicate members.
  - Enclosed by { }
  - keys : values
- If-then-elseif-else:
  - Using indentation:
```
        if a > b or a > c:
            a = 5
            print("a is greater than b or c. a changed to",a)
        elif a == b:
            print("a and b are equal")
        else:
            print("a is not greater or equal to b, nor is it greater
        than c")
```

- ▪ Without indentation:
  ```
  if a > b or a > c: print("a is greater than b or c")
  elif a == b: print("a and b are equal")
  else: print("a is not greater or equal to b, nor is it greater
  than c")
  ```
- o While loop:
  - ▪ Indentation must match:
    ```
    i = 1
    while i < 6:
            print(i)
        if i==3:
            break
        i += 1
    ```
- o For loop:
  - ▪ for … in …:
    ```
    for x in "banana": print(x)
    ```
  - ▪ for … in range():
    ```
    for x in range(2, 30, 3): print(x)        #displays:
    2,5,8,11,…,26,29
    ```
  - ▪ Nested use indentation:
    ```
    adj = ["red", "big", "tasty"]
    fruits = ["apple", "banana", "cherry"]
    for x in adj:
        for y in fruits:
            print(x, y)
    ```
- o Functions:
  - ▪ Use **def** to define and indentation:
    ```
    def my_function(name = "bob"):
        print("My name is " + name)

    my_function("slim shady")
    ```
  - ▪ Recursion:
    ```
    def curs(k):
        if(k>0):
                result = k+curs(k-1)
                print(result)
        else:
            result = 0
        return result

    curs(6)
    ```
  - ▪ Lambda: small anonymous function:
    ```
    x = lambda a, b : a * b
    print(x(5, 6))
    ```
- o File handling:
  - ▪ Open file options:
    - • "r" - Read
    - • "a" - Append
    - • "w" - Write

- "x" - Create
- "t" - Text
- "b" – Binary

- Create a file: `f = open("myfile.txt", "x")`
- Read file: `f = open("demofile.txt","rt")`
  - Read until end of file: `f.read()`
  - Partial read from current curser position: `f.read(5)`
  - Read one line: `f.readline()`
- Write:
  - Append:

    ```
    f = open("demofile.txt", "a")
    f.write("Now the file has one more line!")
    ```

  - Overwrite:

    ```
    f = open("demofile.txt", "w")
    f.write("Woops! I have deleted the content!")
    ```

- File function from OS module:

  ```
  import os
  if os.path.exists("demofile.txt"):
  os.remove("demofile.txt")
  os.rmdir("myfolder")
  ```

4. Numpy module: https://numpy.org/
   o library for scientific computing
   o Quickstart tutorial: https://numpy.org/devdocs/user/quickstart.html
   o import module:

```
import numpy as np
```

   o Data types:
     ▪ int64
     ▪ float64
   o Create arrays and matrices:

```
b = np.array([[1,2,3],[4,5,6]])
a = np.array([1,2], dtype=np.int64)    #force data type to integer
print(b.shape)              #(2,3)
a = np.zeros((2,2))
b = np.ones((1,2))
c = np.full((2,2), 7)
d = np.eye(2)
e = np.random.random((2,2))
```

   o Array indexing:

```
a = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
print(a[0,0], a[1,1], a[2,2])           #1 5 9
row1 = a[0, :]              #row 0
row2 = a[1]                        #row 1
row2 = a[2:4,:]                    #rows 2,3
b = np.array([0, 2, 0, 1])
print(a[np.arange(4), b])       #1  6  7 11  =
e(0,0),(1,2),(2,0),(3,1)
boolix_gt_6 = (a>6)               #true/false for each element
```

   o Array Ops:

```
x + y
np.add(x, y)
x - y
np.subtract(x, y)
x * y
np.multiply(x, y))
x / y
np.divide(x, y)
x ** (1/2)
np.sqrt(x)
v.dot(x)
np.dot(x,y)
np.sum(x)
np.sum(x, axis=0) # Compute sum of each column. axis=1 is each row
x.T              #transpose
```

   o Broadcasting examples:

```
x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
v = np.array([1, 0, 1])
y = np.empty_like(x)    #Create an empty matrix with the same shape as x
```

```python
#Add the vector v to each row of the matrix x:
for i in range(4): y[i, :] = x[i, :] + v

v = np.array([1, 0, 1])
vv = np.tile(v, (4, 1))     #4 copies of v on top of each other

x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
v = np.array([1, 0, 1])
y = x + v                          # Add v to each row of x

v = np.array([1,2,3])
w = np.array([4,5])
np.reshape(v, (3, 1)) * w  #4 5; 8 10; 12 15
```

- Readable
- Math functions: https://docs.scipy.org/doc/numpy/reference/routines.math.html
- Array padding: https://docs.scipy.org/doc/numpy/reference/generated/numpy.pad.html
- Polynomials: https://docs.scipy.org/doc/numpy/reference/routines.polynomials.html
- Sorting searching counting: https://docs.scipy.org/doc/numpy/reference/routines.sort.html

5. Matplotlib Module: https://matplotlib.org/
   o Plotting library
   o Import: `import matplotlib.pyplot as plt`
   o To show inline plots in jupyter: `%matplotlib inline`
   o Plot():

```
x = np.arange(0, 3 * np.pi, 0.1) #x = 0:0.1:3pi
y_sin = np.sin(x)
y_cos = np.cos(x)

plt.plot(x, y_sin)
plt.plot(x,y_cos,'r:')                    #red dotted curve

plt.xlabel('x axis label')
plt.ylabel('y axis label')
plt.title('Sine and Cosine')
plt.legend(['Sine', 'Cosine'])
plt.grid(True)
plt.show()                      #show a plot with 2 curves
```

   ▪ Options for plot: https://matplotlib.org/tutorials/introductory/pyplot.html

   o Scatter plots, Subplots():

```
plt.subplot(2, 1, 1)
plt.plot(x, y_sin)
plt.title('Sine')

plt.subplot(2, 1, 2)
plt.plot(x, y_cos)
plt.title('Cosine')
plt.show()
```

   o Images:
       ▪ Tutorial: https://matplotlib.org/tutorials/introductory/images.html#sphx-glr-tutorials-introductory-images-py
       ▪ Reading images: `img = plt.imread('location/pic.ext')`
       ▪ Display images: `plt.imshow(img, cmap='gray')`
           • 164 Maps help color 2D images that don't have a 3rd color dimension.
       ▪ Histogram: `h = plt.hist(im.ravel(), bins=256)`
           • ravel() converts 2D vector to 1D
           • h is a tuple: h[0] = y axis of histogram; h[1] = x-axis
   o Other plotting functions: https://matplotlib.org/api/pyplot_api.html

6. ImageIO Module: https://pypi.org/project/imageio/
   - Read and write a wide range of image data.
   - Import: `import imageio as io`
   - Read: `in = io.imread('location/file.ext')`
     - Location can online
   - Write: `io.imwrite(''location/file.ext', in)`

7. SKimage sub-Module: https://scikit-image.org/
   - From the Sci-Kit (science kit) module
   - Image processing algorithms
   - Tutorials: https://scikit-image.org/docs/dev/auto_examples/index.html
   - Contains a number of sample images for experimentation
     - Read the camera man image: `im = sk.data.camera()`
   - Convert color images to grayscale: `im = sk.color.rgb2gray(Im)`
   - Filtering: https://scikit-image.org/docs/dev/api/skimage.filters.html
     - Roberts filter: `filt = sk.filters.roberts(im)`
   - Functions (APIs): https://scikit-image.org/docs/dev/api/api.html