**Functions**

1. **Python Basics**
2. Square root function

import math

root = math.sqrt(99)

1. Floor Function to round off to integer

import math

flr = math.floor(89.9)

1. Variables Within Modules

import math

print(math.pi)

a = math.sqrt(math.pi)

b = math.ceil(math.pi)

c = math.floor(math.pi)

1. Csv reader function

import csv

f = open("nfl.csv", 'r')

csvreader = csv.reader(f)

nfl = list(csvreader)

1. Classes and objects

class Dataset:

def \_\_init\_\_(self):

self.type = "csv"

class Dataset:

def \_\_init\_\_(self, data):

self.data = data

f = open("nfl.csv", 'r')

csvreader = csv.reader(f)

nfl\_data = list(csvreader)

nfl\_dataset = Dataset(nfl\_data)

dataset\_data = nfl\_dataset.data

1. Enumerate function in modules

class Dataset:

def \_\_init\_\_(self, data):

self.header = data[0]

self.data = data[1:]

def column(self, label):

if label not in self.header:

return None

index = 0

for idx, element in enumerate(self.header):

if label == element:

index = idx

column = []

for row in self.data:

column.append(row[index])

return column

nfl\_dataset = Dataset(nfl\_data)

year\_column = nfl\_dataset.column('year')

player\_column = nfl\_dataset.column('player')

1. Set() function for unique elements

unique\_animals = set(["Dog", "Cat", "Hippo", "Dog", "Cat", "Dog", "Dog", "Cat"])

print(unique\_animals)

1. Try and except block

try:

int('')

except Exception:

print("There was an error")

try:

int('')

except Exception:

pass

1. Add a column in dataset

for row in legislators:

y = row[2].split('-')[0]

try:

birth\_year = int(y)

except Exception:

birth\_year = 0

row.append(birth\_year)

print(legislators[0:10])

1. Filling missing values in column with last value

last\_value = 1

for row in legislators:

if row[7] == 0:

row[7] = last\_value

last\_value = row[7]

1. Enumerate function to use multiple list in tandem

animals = ["Dog", "Tiger", "SuperLion", "Cow", "Panda"]

viciousness = [1, 5, 10, 10, 1]

for i, animal in enumerate(animals):

print("Animal")

print(animal)

print("Viciousness")

print(viciousness[i])

things = [["apple", "monkey"], ["orange", "dog"], ["banana", "cat"]]

trees = ["cedar", "maple", "fig"]

for i, thing in enumerate(things):

thing.append(trees[i])

1. Short for loops

apple\_prices = [100, 101, 102, 105]

apple\_prices\_doubled = [apple\*2 for apple in apple\_prices]

apple\_prices\_lowered = [apple-100 for apple in apple\_prices]

1. Highest value with None

values = [-50, -80, -100]

max\_value = None

for i in values:

if max\_value is None or i > max\_value:

max\_value = i

max\_value = None

for k in name\_counts:

count = name\_counts[k]

if max\_value is None or count > max\_value:

max\_value = count

1. Items() function for finding highest items

highest\_male\_count = None

for name in male\_name\_counts:

count = male\_name\_counts[name]

if highest\_male\_count is None or count > highest\_male\_count:

highest\_male\_count = count

top\_male\_names = []

for male, count in male\_name\_counts.items():

if count == highest\_male\_count:

top\_male\_names.append(male)

1. For text-mining, regex function to find peculiar words

strings = ["bat", "robotics", "megabyte"]

regex = ""

regex = "b.t"

import re

# to Find text with “of reddit” in the question

of\_reddit\_count = 0

for row in posts:

if re.search("of Reddit", row[0]) is not None:

of\_reddit\_count += 1

# To find both capital and lowercase

import re

of\_reddit\_count\_old = 0

for row in posts:

if re.search("of Reddit", row[0]) is not None:

of\_reddit\_count\_old += 1

of\_reddit\_count = 0

for row in posts:

if re.search("of [Rr]eddit", row[0]) is not None:

of\_reddit\_count += 1

import re

serious\_count = 0

for row in posts:

if re.search("\[Serious\]", row[0]) is not None:

serious\_count += 1

re.sub() to convert all **serious** tags to the format "[Serious]"

import re

for row in posts:

row[0] = re.sub("[\[\(][Ss]erious[\]\)]", "[Serious]", row[0])

1. Time Function

import time

current\_time = time.time()

print(current\_time)

import time

current\_time = time.time()

current\_struct\_time = time.gmtime(current\_time)

current\_hour = current\_struct\_time.tm\_hour

print(current\_hour)

# Datetime module for better date operations

import datetime

current\_datetime = datetime.datetime.utcnow()

current\_year = current\_datetime.year

current\_month = current\_datetime.month

print(current\_datetime)

#Finding dates at later point

import datetime

kirks\_birthday = datetime.datetime(year=2233, month=3, day=22)

diff = datetime.timedelta(weeks = 15)

before\_kirk = kirks\_birthday – diff

# Formatting dates

import datetime

mystery\_date\_formatted\_string = mystery\_date.strftime("%I:%M%p on %A %B %d, %Y")

print(mystery\_date\_formatted\_string)

#The datetime.datetime.strptime() function allows us to convert a *string* to a datetime instance:

import datetime

mystery\_date = datetime.datetime.strptime(mystery\_date\_formatted\_string, "%I:%M%p on %A %B %d, %Y")

print(mystery\_date)

 #We can convert each Unix time stamp into datetime object using the [datetime.datetime.fromtimestamp()](https://docs.python.org/3/library/datetime.html" \l "datetime.datetime.fromtimestamp)function:

import datetime

for row in posts:

float\_stamp = float(row[2])

day = datetime.datetime.fromtimestamp(float\_stamp)

row[2] = day

march\_count = 0

for row in posts:

if row[2].month == 3:

march\_count += 1

def count\_posts\_in\_month(month):

count = 0

for row in posts:

if row[2].month == month:

count += 1

return count

feb\_count = count\_posts\_in\_month(2)

aug\_count = count\_posts\_in\_month(8)