Functions, Scoping, Data Collections 1 & List Comprehensions

Tasks Today:

Monday Additions (or, and ... if statements)

1) String Manipulation
   a) strip()
   b) title()
2) Working With Lists
   a) min()
   b) max()
   c) sum()
   d) sort()
   e) Copying a List
   f) 'in' keyword
   g) 'not in' keyword
   i) Checking an Empty List
   j) Removing Instances with a Loop
3) List Comprehensions
4) Tuples
   a) sorted()
5) Functions
   a) User-Defined vs. Built-In Functions
   b) Accepting Parameters
   c) Default Parameters
   d) Making an Argument Optional
   e) Keyword Arguments
   f) Returning Values
   g) *args
   h) Docstring
   i) Using a User Function in a Loop
6) Scope

Monday Refreshers

Using 'or' in if statements

In [ ]:
```
# covered yesterday
```

Using 'and' in if statements

In [ ]:
```
# covered yesterday
```

Using both 'or' and 'and' in if statements

In [ ]:
```
# covered yesterday
```

String Manipulation

.lstrip()

In [2]:
```
# string.lstrip()
name = "    John"
print(name.lstrip())
```

```
John
```

.rstrip()

In [3]:
```
# string.rstrip()

name = " John                "
print(name.rstrip())
```

```
 John
```

.strip()

In [4]:
```
name = "   John     "
print(name.strip())
```

```
John
```

.title()

In [5]:
```
# string.title()

name = " john smith"
print(name.title())
```

```
 John Smith
```

String Exercise

Strip all white space and capitalize every name in the list given

In [7]:
```
names = ['    coNNor', 'max', ' EVan ', 'JORDAN']
for i in range (len(names)):
    names[i] = names[i].strip().title()
print(names)
```

```
['Connor', 'Max', 'Evan', 'Jordan']
```

Working With Lists

min()

In [8]:
```
# min(list)
numbers = [4,2,97,54,32,5,0]

print(min(numbers))
```

```
0
```

max()

In [10]:
```
# max(list)
numbers = [4,2,97,54,32,5,0]
print(max(numbers))
```

```
97
```

sum()

In [11]:
```
# sum(list)

numbers = [4,2,97,54,32,5,0]
print(sum(numbers))
```

```
194
```

sorted()

In [21]:
```
# sorted(list)
numbers = [4,2,97,54,32,5,0]
sorted_list = sorted(numbers, reverse = True)
print(sorted_list)
print(numbers)
```

```
[97, 54, 32, 5, 4, 2, 0]
[4, 2, 97, 54, 32, 5, 0]
```

.sort()

Difference between sort and sorted, is that sorted doesn't change original list it returns a copy, while .sort changes the original list

In [17]:
```
# list.sort()

numbers = [4,2,97,54,32,5,0]
numbers.sort()
print(numbers)
# use sorted when you don't want to alter original list, use .sort() when you
```

```
[0, 2, 4, 5, 32, 54, 97]
```

Copying a List

In [29]:
```
# [:] copies a list, doesn't alter original
numbers = [4,2,97,54,32,5,0]
#copy_list = numbers.copy()
#print(copy_list)
copy_list = numbers[:]
print(copy_list)
```

```
[4, 2, 97, 54, 32, 5, 0]
```

'in' keyword

In [31]:
```
l_3 = ["connor","Joel","Max","Derek"]

if "Derek" in l_3:
    print('Derek in the list')
```

```
derek in list
```

'not in' keyword

In [33]:
```
if "John" not in l_3:
        print('John Not in the list')
```

```
John Not in the list
```

Checking an Empty List

In [ ]:
```
# if l_1: or if l_1 = []
l_2 = []
if l_2 == []
    print("list empty")
```

Removing Instances with a Loop

In [35]:
```
# while, remove
names = ["Conor","Max","Evan","Rob","Evan"]
while "Evan" in names:
        names.remove("Evan")
print(names)
```

```
['Conor', 'Max', 'Rob']
```

List Exercise

Remove all duplicates

Extra: Create a program that will remove any duplicates from a given list

In [44]:
```
names = ['connor', 'connor', 'bob', 'connor', 'evan', 'max', 'evan', 2, 2, 2,
dub_names = []
for name in names:
    if name not in dub_names:
        dub_names.append(name)
print(dub_names)
```

```
['connor', 'bob', 'evan', 'max', 2, 3, 4, 'kevin']
```

List Comprehensions

Creating a quickly generated list to work with
result = [*transform* *iteration* *filter*]

In [52]:
```
# number comprehension


nums = []
for i in range(100):
    nums.append(i)

print(nums)

nums_comp = [x*200/7 for x in range(100)]

print(nums_comp)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 2
1, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 4
0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 5
9, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 7
8, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 9
7, 98, 99]
[0.0, 28.571428571428573, 57.142857142857146, 85.71428571428571, 114.28571428
571429, 142.85714285714286, 171.42857142857142, 200.0, 228.57142857142858, 25
7.14285714285717, 285.7142857142857, 314.2857142857143, 342.85714285714283, 3
71.42857142857144, 400.0, 428.57142857142856, 457.14285714285717, 485.7142857
142857, 514.2857142857143, 542.8571428571429, 571.4285714285714, 600.0, 628.5
714285714286, 657.1428571428571, 685.7142857142857, 714.2857142857143, 742.85
71428571429, 771.4285714285714, 800.0, 828.5714285714286, 857.1428571428571,
885.7142857142857, 914.2857142857143, 942.8571428571429, 971.4285714285714, 1
000.0, 1028.5714285714287, 1057.142857142857, 1085.7142857142858, 1114.285714
2857142, 1142.857142857143, 1171.4285714285713, 1200.0, 1228.5714285714287, 1
257.142857142857, 1285.7142857142858, 1314.2857142857142, 1342.857142857143,
1371.4285714285713, 1400.0, 1428.5714285714287, 1457.142857142857, 1485.71428
57142858, 1514.2857142857142, 1542.857142857143, 1571.4285714285713, 1600.0,
1628.5714285714287, 1657.142857142857, 1685.7142857142858, 1714.285714285714
2, 1742.857142857143, 1771.4285714285713, 1800.0, 1828.5714285714287, 1857.14
2857142857, 1885.7142857142858, 1914.2857142857142, 1942.857142857143, 1971.4
285714285713, 2000.0, 2028.5714285714287, 2057.1428571428573, 2085.7142857142
86, 2114.285714285714, 2142.8571428571427, 2171.4285714285716, 2200.0, 2228.5
714285714284, 2257.1428571428573, 2285.714285714286, 2314.285714285714, 2342.
8571428571427, 2371.4285714285716, 2400.0, 2428.5714285714284, 2457.142857142
8573, 2485.714285714286, 2514.285714285714, 2542.8571428571427, 2571.42857142
85716, 2600.0, 2628.5714285714284, 2657.1428571428573, 2685.714285714286, 271
4.285714285714, 2742.8571428571427, 2771.4285714285716, 2800.0, 2828.57142857
14284]
```

In [55]:
```
# square number comprehension
squares = []
squares = [x*x for x in range(100)]
print(squares)
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289,
324, 361, 400, 441, 484, 529, 576, 625, 676, 729, 784, 841, 900, 961, 1024, 1
089, 1156, 1225, 1296, 1369, 1444, 1521, 1600, 1681, 1764, 1849, 1936, 2025,
2116, 2209, 2304, 2401, 2500, 2601, 2704, 2809, 2916, 3025, 3136, 3249, 3364,
3481, 3600, 3721, 3844, 3969, 4096, 4225, 4356, 4489, 4624, 4761, 4900, 5041,
5184, 5329, 5476, 5625, 5776, 5929, 6084, 6241, 6400, 6561, 6724, 6889, 7056,
7225, 7396, 7569, 7744, 7921, 8100, 8281, 8464, 8649, 8836, 9025, 9216, 9409,
9604, 9801]
```

In [177]:
```
# string comprehension
names = ['chonor', 'max','evan','rob']
first_char = []
for name in names:
    first_char.append(name[0])
print(first_char)

first_char_comp = [name[0] for name in names]
print(first_char_comp)
```

```
['c', 'm', 'e', 'r']
['c', 'm', 'e', 'r']
```

In [176]:
```
# using the 'if' statement ... if statement after forc
names = ['chonor', 'max','evan','rob']
c_name = [name for name in names if name[0] == "C"]
print(c_names)

#c_names_long = []
 #   for name in names:
 #       if name[0] == "C"
#            c_names_long.append(name)

#print(c_names_long)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-176-169cd11dc0e8> in <module>
      2 names = ['chonor', 'max','evan','rob']
      3 c_name = [name for name in names if name[0] == "C"]
----> 4 print(c_names)
      5
      6 #c_names_long = []

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

In [ ]:
```
# using multiplication outside the list, list comprehension within list compre
```

List Comprehension Exercise

Create a grid of alternating 0's and 1's (5x5 grid)

In [119]:
```
# hint use %
grid = [[row % 2 for col in range(5)] for row in range(5)]

grid


for row in range(5):
    grid = []
    for col in range(5):
        grid.append(row%2)
    print(grid)
```

```
[0, 0, 0, 0, 0]
[1, 1, 1, 1, 1]
[0, 0, 0, 0, 0]
[1, 1, 1, 1, 1]
[0, 0, 0, 0, 0]
```

Tuples

Defined as an immutable list

Seperated by commas using parenthesis

In [ ]:
```
# can be defined with or without parens (try is instance)
# type()
```

sorted()

In [121]:
```
# sorted(tuple) ... returns a list

t_2 = (20,34,52,23,4,6,77,54,)
print(sorted(t_2))
```

```
[4, 6, 20, 23, 34, 52, 54, 77]
```

Adding values to a Tuple

In [180]:
```
t = (1,)
t = t + (5,)
print(t)
```

```
(1, 5)
```

Functions

User-Defined vs. Built-In Functions

In [134]:
```
# BUit in funktion = range()
def SayHello():
        print("Hello wolrd")
    #To call functio
print(SayHello)
SayHello()
```

```
<function SayHello at 0x7ff4082dad40>
Hello wolrd
```

Accepting Parameters

In [ ]:
```
# order matters
# can take in any type of variable

# def sayHello(name):
#     print('Hello {}'.format(name))

# sayHello('Connor')
```

Default Parameters

In [136]:
```
# default paramaters need to be after non-default parameters at all times
def printAgentName(first, middle = "007", last="Bond"):
    print("hello {} {} {}".format(first,middle,last))
name = " JAmes"

printAgentName(name)
```

```
hello  JAmes 007 Bond
```

Making an Argument Optional

In [ ]:
```
# name=''
```

Keyword Arguments

In [ ]:
```
# last_name='Max', first_name='Smith' in the function call

# see above
```

Creating a start, stop, step function

In [150]:
```
# def my_range(stop, start=0, step=1):

def my_range(stop, start = 0, step = 1):
    for i in range(start,stop,step):
        print(i)
my_range(100,50,2)
```

```
50
52
54
56
58
60
62
64
66
68
70
72
74
76
78
80
82
84
86
88
90
92
94
96
98
```

Returning Values

In [ ]:
```
# using the return keyword, returns a certain value back to where the function
```

****args***

In [ ]:
```
# stands for arguments, takes any number of arguments as parameters
# must be last if multiple parameters

def prtintArgs(num1, num2 , *args)
    print(num1)
    print(num2)
    print(args)
        for arg in args:
            print(arg)
```

Docstring

In [ ]:
```
# description of function, used with ''' ''' and a

"""


"""
```

Using a User Function in a Loop

In [ ]:
```
# add two lists together via index

def ask(answer)
    if answer.lower == "quit":
        break
```

Function Exercise

Write a function that loops through a list of first_names and a list of last_names, combines the two and return a list of full_names

In [ ]:
```
first_name = ['John', 'Evan', 'Jordan', 'Max']
last_name = ['Smith', 'Smith', 'Williams', 'Bell']
```

Scope

Scope refers to the ability to access variables, different types of scope include:
a) Global
b) Function (local)
c) Class (local)

In [170]:
```
# placement of variable declaration matters


num = 3 # global variable
def my_func():
    num3 = 6 #
    return num3
    #print(num3)# this is will be give a Name Error - not defined
num2 = my_func()
print(num2)
```

```
6
```

Exercises

Exercise 1

Given a list as a parameter,write a function that returns a list of numbers that are less than ten

For example: Say your input parameter to the function is [1,11,14,5,8,9]...Your output should [1,5,8,9]

In [211]:
```
def list_less_then_ten(a_list):
    b_list = [a_list[i] for i in range(len(a_list)) if a_list[i] < 10]
    return b_list
```

```
[1, 5, 8, 9]
```

In [212]:
```
c_list =[1,11,14,5,8,9]
print(list_less_then_ten(c_list))
```

```
[1, 5, 8, 9]
```

Exercise 2

Write a function that takes in two lists and returns the two lists merged together and sorted
Hint: You can use the .sort() method

In [234]:
```
def list_merge_by_index(list_a,list_b):
    def merge_list(list_big, list_small):
        list_c = []
        index_sum = []
        for i in range(len(list_big)):
            if i < len(list_small):
                list_c.append(list_small[i])
                list_c.append(list_big[i])
                index_sum.append(i*2)
            else:
                list_c.append(list_big[i])
                index_sum.append(i)
        return sorted(list_c, reverse = True), sorted(index_sum, reverse = Tru
    if len(list_a) >= len(list_b):
        return merge_list(list_a,list_b)
    else:
        return merge_list(list_b,list_a)
list_a = [1,2,3,4,5,6]
list_b = [8,9,10]
print(list_merge_by_index(list_a,list_b))



# adds up lists together by index and returns sum of each index in sorted orde
```

```
([10, 9, 8, 6, 5, 4, 3, 2, 1], [5, 4, 4, 3, 2, 0])
```

Exercise 3

Ask the user to input a number and check if that number is prime using a function, loop (using a WHILE Loop) until they no longer want to check a number.
Hint: Use yesterday's prime number code

In [*]:
```
def check_Prime(x):
    check = True
    for j in range (2,x,1):
        if x % j == 0:
            check = False
    if check == True:
        print ("Number {} is Prime ".format(x))
    else:
        print ("Number {} is not Prime ".format(x))
def user_ask():
    answer = ""
    while answer.lower() != "no":
        try:
            check_Prime(int(input("Enter an number for Prime check: ")))
        except:
            print("You have to use integer numbers!!! ")
        answer = input("Do you want to continue? ")
user_ask()
```

```
Enter an number for Prime check: f
U have to use integer numbers
Do you want to continue?
Enter an number for Prime check: 1001
Number 1001 is not Prime
Do you want to continue?
Enter an number for Prime check: 1031
Number 1031 is Prime
Do you want to continue? 13701
Enter an number for Prime check: 13701
Number 13701 is not Prime
Do you want to continue? 13707
Enter an number for Prime check: 13707
Number 13707 is not Prime
Do you want to continue?
Enter an number for Prime check: 400217
Number 400217 is Prime
Do you want to continue? 909773
Enter an number for Prime check: 909773
Number 909773 is Prime
Do you want to continue?
Enter an number for Prime check: 999953
Number 999953 is Prime

Do you want to continue?
```

Exercise 4

Create a calculator using functions that asks for two numbers and performs a calculation that the user inputs...
Loop until the user chooses not to perform any more calculations.
Hint: Take yesterday's code from the extra exercise...

In [268]:

```
def calculation_1():
    answer = ""
    while answer.lower() != "no":
        x = int(input("Enter a first number: "))
        y = int(input("Enter a second number "))
        z = input(" + - / * ")
        if z == "+":
            print(x+y)
        elif z == "-":
            print(x-y)
        elif z == "*":
            print(x*y)
        elif z == "/":
            print(x/y)
        answer = input("Do you want to continue? ")
calculation_1()
```

```
Enter a first number: 12
Enter a second number 44
 + - / * 8
Do you want to continue?
Enter a first number: 13
Enter a second number 44
 + - / * *
572


---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
/usr/lib/python3.7/site-packages/ipykernel/kernelbase.py in _input_request(se
lf, prompt, ident, parent, password)
    884             try:
--> 885                 ident, reply = self.session.recv(self.stdin_socket,
0)
    886             except Exception:

/usr/lib/python3.7/site-packages/jupyter_client/session.py in recv(self, sock
et, mode, content, copy)
    802         try:
--> 803             msg_list = socket.recv_multipart(mode, copy=copy)
    804         except zmq.ZMQError as e:

/usr/lib/python3.7/site-packages/zmq/sugar/socket.py in recv_multipart(self,
flags, copy, track)
    469         """
--> 470         parts = [self.recv(flags, copy=copy, track=track)]
    471         # have first part already, only loop while more to receive

zmq/backend/cython/socket.pyx in zmq.backend.cython.socket.Socket.recv()

zmq/backend/cython/socket.pyx in zmq.backend.cython.socket.Socket.recv()

zmq/backend/cython/socket.pyx in zmq.backend.cython.socket._recv_copy()

/usr/lib/python3.7/site-packages/zmq/backend/cython/checkrc.pxd in zmq.backen
d.cython.checkrc._check_rc()

KeyboardInterrupt:

During handling of the above exception, another exception occurred:

KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-268-81aef8e4ff05> in <module>
     14             print(x/y)
```

In [ ]: