Python Project 1

All questions are weighted equally. Total number of points is 50 with an extra credit (Part 4) worth 10 points.

Part 1 - Conditionals

We use conditionals to make decisions in our code. Recall the syntax for the 3 different conditional constructs.

- Construct 1 - to execute a block of code if the condition is true and do nothing if it is false.

```
if (test condition) :
    statements to execute
```

- Construct 2 - to execute a block of code if the condition is true and another block if it is false.

```
if (condition) :
    execute some code
else :
    execute some other code
```

- Construct 3 - to have multiple `elif` statements but the last conditional may be an `else` statement. Here is an example with 4 branches.

```
if (condition 1) :
    first group of statements to execute
elif (condition 2) :
    second group of statements to execute
elif (condition 3) :
    third group of statements to execute
else:
    we do not need a condition here because it is only executed if all previous
conditions are not satisfied
```

Question 1.

First input the lists x and y given below which are the values defining the sides of two different rectangles. So the first rectangle has sides given by the first element of list x and the first element of list y; similarly for the second rectangle. Calculate the area of the first rectangle and print out. Then write a conditional which tests to see if the area is greater than 25. If the area is greater than 25, use a formatted print statement to print out the

fact that the area is >25 and the values of the sides. If the area is <= 25, print out an appropriate statement along with values of x, y, and area. There are two rectangles so you could do a loop but here I just want you to access the correct element from the list (recall that this is done using square brackets). The easiest way is to just cut, paste and modify your code for the first rectangle. I have provided a skeleton code.

```
x = [2, 4]
y = [5, 8]


# first rectangle
print (f"Output for first rectangle with sides x = {x[0]} and y = {y[0]}")
area =   y[0] * x[0]


# Add conditional block with print statements

if (area > 25):
    print(f"The area is greater than 25. The values for the sides are {x[0]} and {y[0]} ")
elif (area <= 25):
    print(f"The area is less than or equal too 25. The values for the sides are {x[0]} and {y[0]}
```

```
Output for first rectangle with sides x = 2 and y = 5
The area is less than or equal too 25. The values for the sides are 2 and 5
```

```
# Repeat results for second rectangle


print (f"Output for first rectangle with sides x = {x[1]} and y = {y[1]}")
area =   y[1] * x[1]


# Add conditional block with print statements

if (area > 25):
    print(f"The area is greater than 25. The values for the sides are {x[1]} and {y[1]} ")
elif (area <= 25):
    print(f"The area is less than or equal too 25. The values for the sides are {x[1]} and {y[1]}
```

```
Output for first rectangle with sides x = 4 and y = 8
The area is greater than 25. The values for the sides are 4 and 8
```

Question 2.

Write a conditional to test if a number x is

- negative
- nonnegative and less than 10
- 10 or greater but less than or equal to 25

- greater than 25
- if x is greater than 25 print this out and then check to see if it is even or odd (Hint: put a conditional inside another conditional and use the modulo operator which is accessed using percent sign; for example 4 % 2 gives 0 and so 4 is even)

Add formatted print statements for each condition giving the value of x and its range (for example, x is greater than 7 but less than 20) and run your code for x=-3, 7, 21, 45 and 88. For simplicity, when you use a different value for x just paste your code in the next cell.

```
x = -3
# add conditional block
if (x < 0) :
    print (f"x = {x}")
    print ("x is negative")
# add remaining branches of conditional
elif (0 < x < 10):
    print (f"x = {x}")
    print ("x is nonnegative but less than 10")
elif (10 < x < 25):
    print (f"x = {x}")
    print ("x is greater than 10 but less than 25")
elif (x > 25):
    print (f"x = {x}")
    print ("x is greater than 25")

    if (x % 2):
        print("x is odd")
    else:
        print("x is even")
```

```
x = 100
x is greater than 25
x is even
```

```
x = 7
if (x < 0) :
    print (f"x = {x}")
    print ("x is negative")
# add remaining branches of conditional
elif (0 < x < 10):
    print (f"x = {x}")
    print ("x is nonnegative but less than 10")
elif (10 < x < 25):
    print (f"x = {x}")
    print ("x is greater than 10 but less than 25")
elif (x > 25):
    print (f"x = {x}")
    print ("x is greater than 25")

    if (x % 2):
```

```
        print("x is odd")
    else:
        print("x is even")
```

```
x = 7
x is nonnegative but less than 10
```

```
x = 21
if (x < 0) :
    print (f"x = {x}")
    print ("x is negative")
# add remaining branches of conditional
elif (0 < x < 10):
    print (f"x = {x}")
    print ("x is nonnegative but less than 10")
elif (10 < x < 25):
    print (f"x = {x}")
    print ("x is greater than 10 but less than 25")
elif (x > 25):
    print (f"x = {x}")
    print ("x is greater than 25")

    if (x % 2):
        print("x is odd")
    else:
        print("x is even")
```

```
x = 21
x is greater than 10 but less than 25
```

```
x = 45
if (x < 0) :
    print (f"x = {x}")
    print ("x is negative")
# add remaining branches of conditional
elif (0 < x < 10):
    print (f"x = {x}")
    print ("x is nonnegative but less than 10")
elif (10 < x < 25):
    print (f"x = {x}")
    print ("x is greater than 10 but less than 25")
elif (x > 25):
    print (f"x = {x}")
    print ("x is greater than 25")

    if (x % 2):
        print("x is odd")
    else:
        print("x is even")
```

```
x = 45
x is greater than 25
x is odd
```

```
x = 88
if (x < 0) :
    print (f"x = {x}")
    print ("x is negative")
# add remaining branches of conditional
elif (0 < x < 10):
    print (f"x = {x}")
    print ("x is nonnegative but less than 10")
elif (10 < x < 25):
    print (f"x = {x}")
    print ("x is greater than 10 but less than 25")
elif (x > 25):
    print (f"x = {x}")
    print ("x is greater than 25")

    if (x % 2):
        print("x is odd")
    else:
        print("x is even")
```

```
x = 88
x is greater than 25
x is even
```

Part 2 - Functions & Loops

We want to break our codes into blocks of code which can be used in different parts of the program. To do this we use Python functions. The syntax is

```
def function_name(optional arguments separated by commas) :
    some statements
    return items to return to main program, if any
```

Remember the colon (:) after the first line; all remaining statements must be indented.

To call the function you just need the function name with the values for your inputs in the same order than they are listed in the definition of the function.

We use `for` loops to perform the same block of code multiple times with different variable values. Typically we either loop over elements in a list or a range of values. The syntax is review below.

- for loop over a list

```
my_list = []  # define an empty list
for a in my_list:
    statements to execute
```

- for loop over a range of values

```
for i in range(m,n,d):
    statements to execute
```

Here i takes on the starting value i=m and ends with i = n-1 with an increment of d. If the increment is omitted then the default value is 1.

Question 3.

Write a function to calculate the sum of the first n positive integers starting with 1; the only input should be n and it should return the sum. Test your function with n=1000 and print out result using a formatted print statement. There is a formula for calculating this sum so check your answer with the formula: n(n+1) / 2.
Hints:

- don't name your variable for the sum as `sum` because Python uses this name.
- don't forget to initialize your sum to zero before the loop.
- remember that in the loop `for i in range(1,5)` `i` takes on the values 1,2,3, 4, but NOT 5

```
def first_sum(n):
    z = n * (n + 1) // 2
    return (z)


n = 1000
z=first_sum(n)
print(f"my result is {z}")

print(f"result using formula is {n * (n+1) / 2}")
```

```
my result is 500500
result using formula is 500500.0
```

Question 4.

We are given the two lists

```
horse_breed = ['arabian', 'thoroughbred', 'quarter horse']
color = ['gray', 'chestnut', 'bay']
```

Use a nested for loop to print out each combination of horse color & breed. For example, print out "gray arabian", etc. There should be 9 combinations of color and breed.

```
horse_breed = ['arabian', 'thoroughbred', 'quarter horse']
color = ['gray', 'chestnut', 'bay']
for i in  horse_breed:
    for j in  color:
        print(i,j)
```

```
arabian gray
arabian chestnut
arabian bay
thoroughbred gray
thoroughbred chestnut
thoroughbred bay
quarter horse gray
quarter horse chestnut
quarter horse bay
```

Part 3 - Plotting with matplotlib & using numpy

Recall that to import a library using an alias we use the syntax

```
import library_name as alias
```

Typically we import numpy with alias np and matplotlib.pyplot with alias plt. Remember that to access a function in, for example, the NumPy library you need to use the syntax `np.function_name` .

For plotting syntax using matplotlib.plt check out the notebook entitled Plotting with the Python Library matplotlib.

Question 5.

The natural logarithm and the exponential (e^x) are inverse functions. This means that y=e^x if and only if ln y =x. For example, ln 1 = 0 if and only if 1=e^0. These functions are available in NumPy using `np.exp(x)`, `np.log(x)` . Plot both functions on the same plot on the interval [0.01, 2]; add a title and a legend. Use red for the exponential and green for the logarithm. Add a title and legend.

Hints:

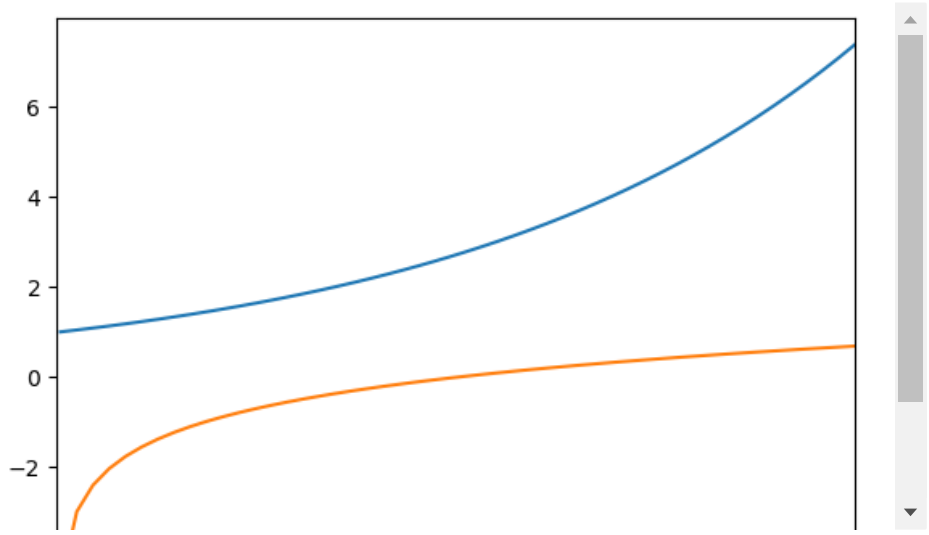
-Check out the notebook entitled Plotting with the Python Library matplotlib where we plot the sin and cos on same plot.

-You will need to use the NumPy function `linspace` to set up the points on the x-axis.

```
# import libraries  Numpy  and matplotlib.pyplot using standard aliases
import numpy as np
import matplotlib.pyplot as plt
```

```
# plot functions on same plot
x = np.linspace (0.01,2)
y = np.exp(x)
z = np.log(x)
plt.plot(x,y)
plt.plot(x,z)
plt.xlim([0,2])
```

```
(0.0, 2.0)
```



Part 4 - Putting it all together (Extra credit - 10 pts)

Question 6.

There are many famous sequences in mathematics. One of these is a sequence called the Catalan numbers (named after a Belgian mathematician) which occur in some counting problems. The first few terms in the sequence are 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862

There is a formula for calculating the terms in the sequence given by

$\frac{(2n)!}{(n!)(n+1)!}$ for n=0,1,2,3,...

Recall that, for example $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$ and 0! is defined as 1.

Your objective is to create a Python list containing the first 8 terms in the sequence (starting at n=0 and ending at n=7) and then plot them where on the x-axis we plot n and on the y-axis we plot the corresponding Catalan number.

- First write a function that computes n!; remember that when n=0, n!=1 so you will need a conditional to test if n = 0. Test the function before proceeding to verify it is working.
- Next write a loop to create a list containing the desired Catalan numbers; for clarity, assign a variable name to each of the three terms in the formula (n!, (2n)! and (n+1)!). For plotting purposes you may want to also create a list of the values of n used to generate the Catalan numbers.
- Finally plot the numbers using a red triangle to indicate each point; on the x-axis is n and the y-axis is the Catalan number. Add a title for your plot and labels for the x and y axes.

```
# Define function to calculate n!
def factorial_func (n):
    if (n < 2):
        return (1)
    else:
        x = 1
        for i in range (2, n+1):
            x = x * i
        return (x)

n = 10
print(factorial_func(n))
```

```
3628800
```

```
# Create a list of Catalan numbers using a for loop and the function you created to compute the f
n = 8
def catalan(n):
    z = factorial_func(2 * n) // (factorial_func(n + 1) * factorial_func(n))
    return (z)

n = 8
print(catalan(n))
```

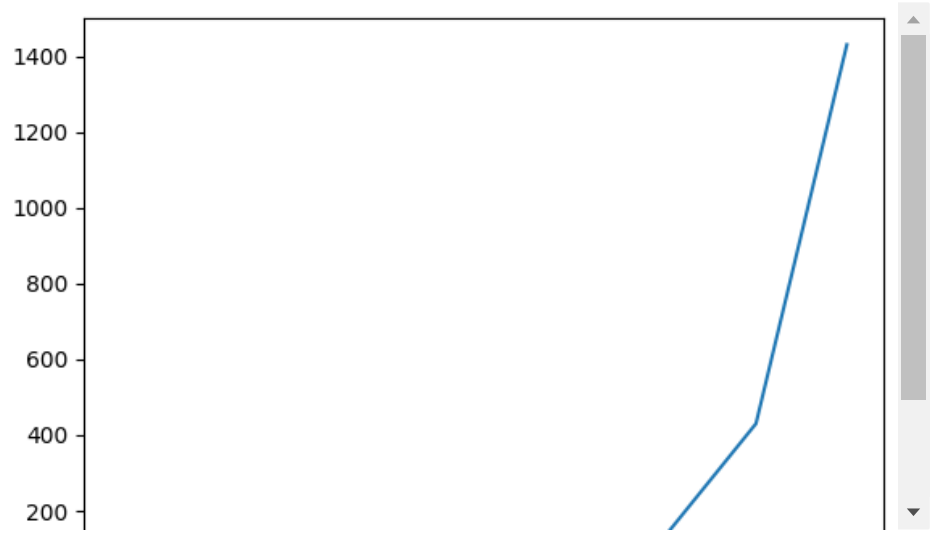
```
1430
```

```
# Plot
import matplotlib.pyplot as plt

catalan_numbers = [catalan(i) for i in range(n + 1)]
```

```
plt.plot(list(range(n + 1)), catalan_numbers)
```

```
[<matplotlib.lines.Line2D at 0x7f733cf02dd0>]
```




```
.--. .- ..- .-..   .. ...   --. .- -.--
```