Econ 425 - HomeAssignment-1

Anshika Sharma, UCLA ID(305488635)

In [1]:
```
#importing the packages
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import scipy.stats as st
import os
```

Q1. Write a Python function that takes a positive integer N and returns the factorial of N, i.e., N! The factorial of N, denoted N!, is the product of the integers from 1 to N. (1 Point)

In [2]:
```
# Function to find factorial of given number
def factorial(n):
    if n < 0:
        return ("Not Defined")
    elif n ==0 :
        return 1
    else:
        x = 1
        for i in range(2, n+1):
            x *= i
        return x

# Checking the results
num = int(input("Input a number to compute the factiorial : "))
print("Factorial of", num, "is", factorial(num))
```

```
Input a number to compute the factiorial : 5
Factorial of 5 is 120
```

Q2. Write a short Python function that takes a sequence of integer values and determines if there is a distinct pair of numbers in the sequence whose product is odd.

In [3]:
```
# Defining the python function, "prod"
def prod(lst):
    for i in range(len(lst)):
        for j in range(len(lst)):
            if i != j:
                prod = lst[i] * lst[j]
                if prod & 1: #product is odd
                    return True
                else:
                    return False

# Checking the results
print(prod([2,4,5]))
print(prod([1,3,4]))
print(prod([2,4]))
```

```
False
True
False
```

Q3. Write a python function that takes an integer (e.g., 342, -123) and returns its reverse digit (i.e., 243, -321).

In [4]:
```
# Function that takes an integer and returns its reverse digit
def reverse(x):
    sign = -1 if x < 0 else 1 #convert negative value to postive one and use t
he following logic to make a reverse
    x *= sign
    rev = 0
    while (x>0) :
        remainder = x % 10  #find the remainder from dividing x by 10
        rev = (rev * 10) + remainder  #add the remainder to rev*10
        x = x // 10
    result = rev * sign  #backtranform to the positive or negative value
    print("The Reverse digit is", result)


# Print to determine if function worked
reverse(342)
reverse(-123)
```

```
The Reverse digit is 243
The Reverse digit is -321
```

Q4. Write a Python function that takes a string s, representing a sentence,and returns a copy of the string with all comma removed.

In [5]:
```
######
def removeCommas(s):
   outStr = ""
   for char in s:       #for each char in s
       if char != ",": # if char is not comma add to outStr
           outStr = outStr + char
   return outStr


# Test
s = "Sit down, Please"
print("With commas:", s)
print("Without commas:", removeCommas(s))

s = "Hello Python, I don't really know you well"
print(s)
print(removeCommas(s))
```

```
With commas: Sit down, Please
Without commas: Sit down Please
Hello Python, I don't really know you well
Hello Python I don't really know you well
```

Q5. Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', write a Python function determine if the input string is valid.

In [6]:
```
#####
def checkBrackets(s):
   print("s = ", s)
   brackets = []
   for char in s:
       if char in ['[', '{', '(']:       #if char is opening bracket, add to l
ist brackets
           brackets.append(char)

       if len(brackets) == 0:           #if brackets is empty and char is clos
ed bracket
           return False

       if char == ']' and brackets.pop() != '[': #opening and closing bracker
 should match
           return False

       if char == ')' and brackets.pop() != '(':
           return False

       if char == '}' and brackets.pop() != '{':
           return False

   if len(brackets) == 0:
       return True
   else:
       return False

#Test
s = "()"
print(checkBrackets(s))

s = "({" + "})"
print(checkBrackets(s))

s = "(}"
print(checkBrackets(s))

s = "([{" + "})]"
print(checkBrackets(s))

s = ")"
print(checkBrackets(s))
```

```
s =  ()
True
s =  ({})
True
s =  (}
False
s =  ([{})]
False
s =  )
False
```

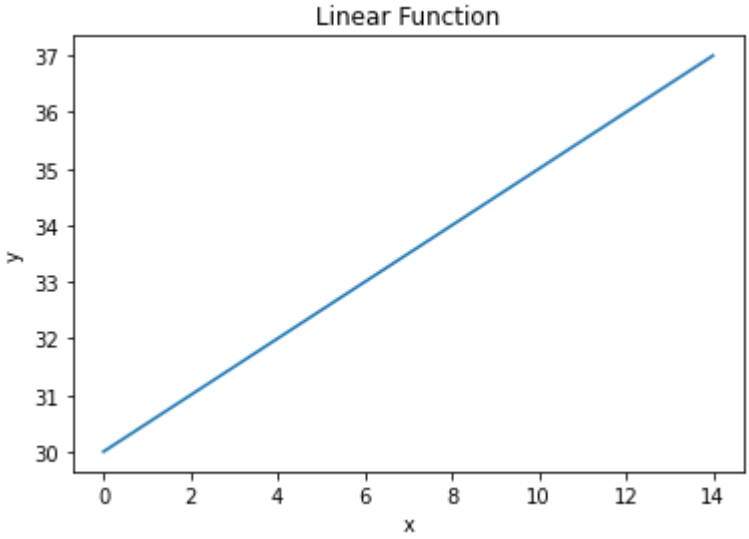
Q6. Write a Python function that merges two sorted lists and return a new sorted list. Both the input lists and output lists should be sorted. You might use the Python list as the data structure

In [7]:
```
#####
def merge(lst1, lst2):
   ind1 = 0   #index to list1
   ind2 = 0   #index to list2
   outList = []   #output list

   while ind1 < len(lst1) and ind2 < len(lst2): #loop until both lists have el
ements
       if lst1[ind1] < lst2[ind2]:    #if element in list1 is smaller
           outList.append(lst1[ind1])   #add it to output list
           ind1 = ind1 + 1

       else:
           outList.append(lst2[ind2])   #if element in list2 is smaller
           ind2 = ind2 + 1

   outList = outList + lst1[ind1:] + lst2[ind2:]
   return outList


#Test
print(merge([1,3,4], [1,2,6,8]))
```

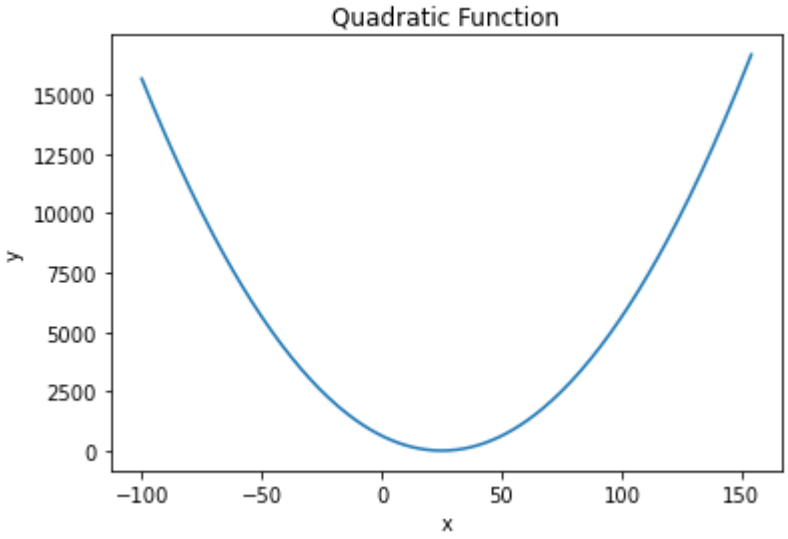
```
[1, 1, 2, 3, 4, 6, 8]
```

Q7. Please select a proper range for the variable x, and write a python function to visualize the following functions:
-Straight line y=θ_0+θ_1 x where θ_0=30,θ_1=0.5
-Quadratic function: y=(x-θ_1)^2+θ_0,where θ_1=25,θ_0=20
-Log function, y=-log(x) and y=-log(1-x)
-Sigmoid function, y=1/(1+e^(-x))
Note that you should select a set of proper values for x, calculate the corresponding y values and plot (x,y) pairs.

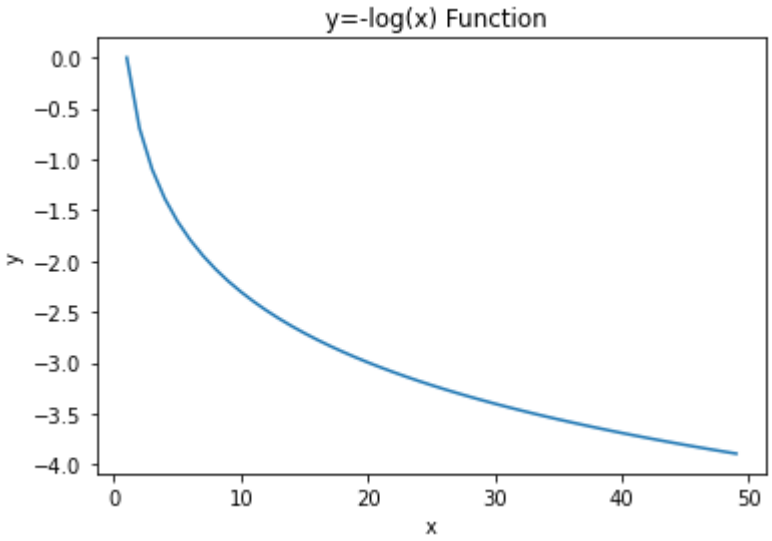
In [8]:
```
import numpy as np
import matplotlib.pyplot as plt

# Straight line y=ϑ_0+ϑ_1 x where ϑ_0=30,ϑ_1=0.5
ax = plt.subplot(111)
x = np.arange(0,15)
theta0 = 30
theta1 = 0.5
plt.plot(x,theta0+theta1*x,linestyle='-')
plt.xlabel("x")
plt.ylabel("y")
plt.title("Linear Function")
plt.show()
```



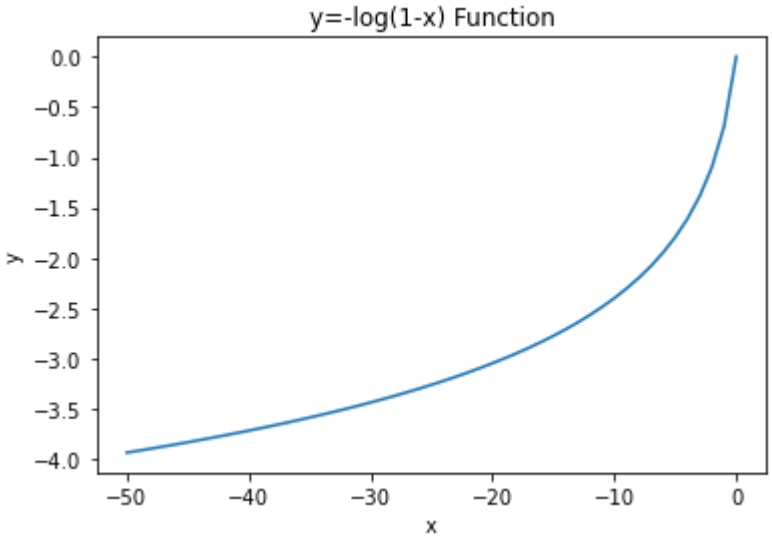
In [9]:
```
# Quadratic function: y=(x-ϑ_1 )^2+ϑ_0,where ϑ_1=25,ϑ_0=20
ax = plt.subplot(111)
x = np.arange(-100,155)
theta0 = 20
theta1 = 25
y = ((x-theta1)**2)+theta0
plt.plot(x,y,linestyle='-')
plt.xlabel("x")
plt.ylabel("y")
plt.title("Quadratic Function")
plt.show()
```



In [10]:
```
# Log function, y=-log(x)
ax = plt.subplot(111)
x = np.arange(1,50)
y = -np.log(x)
plt.plot(x,y,linestyle='-')
plt.xlabel("x")
plt.ylabel("y")
plt.title(" y=-log(x) Function")
plt.show()
```



In [11]:
```
# Log function, y=-log(1-x)
ax = plt.subplot(111)
x = np.arange(-50,1)
y = -np.log(1-x)
plt.plot(x,y,linestyle='-')
plt.xlabel("x")
plt.ylabel("y")
plt.title("y=-log(1-x) Function")
plt.show()
```



In [12]:
```
# Sigmoid function, y=1/(1+e^(-x))
ax = plt.subplot(111)
x = np.arange(-50,50)
y = 1/(1+np.exp(-x))
plt.plot(x,y,linestyle='-')
plt.xlabel("x")
plt.ylabel("y")
plt.title("Sigmod Function")
plt.show()
```