```
#
---------------
-
# Spencer Neveux
# EE 381
# 4/17/18
# Project 4
# --------------


# ---------------------------------------
# Title - Hypothesis Testing
# ---------------------------------------


# Importing and setting up variables
import matplotlib.pyplot as plt
import numpy as np
import math


number_trials = 18
p = 0.5


# ---------------------------------------
# Menu
# ---------------------------------------
def PrintMenu():
  print("\nMain Menu\n1. Hypothesis Test Statement\n2. Binomial
Distribution\n3. Critical Value\
  \n4. Binomial Probability\n5. Power Test\n6. Quit")
```

```
# ---------------------------------------
# Get User Menu Choice
# ---------------------------------------
def GetMenuChoice():
    user_input = int(input("\nChoose a function by its appropriate number: "))
    while not(1 <= user_input <=6):
        user_input = int(input("\nThat isn't valid. Choose a function by its appropriate number: "))

    return user_input


# ---------------------------------------
# Combinations Calculation
# ---------------------------------------
def nCx(n, x):
    factorial = math.factorial
    return (factorial(n) // (factorial(x) * factorial(n - x)))


# ---------------------------------------
# Generate Graph
# ---------------------------------------
def Graph(x_value_list, probability_list):
    # Setting Up Figure
    fig = plt.figure()
    fig.suptitle("Lab 4: Hypothesis Testing")


    # Set up labels
```

```
  ax = fig.add_subplot(111)
  fig.subplots_adjust(left=.125, top=0.85)

  ax.set_title("Probability Doohickey")
  ax.set_xlabel("R.V. values(x)")
  ax.set_ylabel("Probability")

  plt.bar(x_value_list, probability_list, color="green")
  plt.show()

# ---------------------------------------
# Hypothesis Statement
# ---------------------------------------
def HypothesisStatement():
  print("\nThe Hypothesis Statement is: H0: p = 50% Ha: p > 50%")

# ---------------------------------------
# Binomial Distribution
# ---------------------------------------
def BinomialDist():
  probability_list = []
  x_value_list = list(range(0, 19))

  for x_values in x_value_list:
     probability_x = nCx(18, x_values) * (p ** x_values) * ((1-p) **
(number_trials - x_values))
     probability_list.append(probability_x)
```

```
    return x_value_list, probability_list


# ---------------------------------------
# Critical Value
# ---------------------------------------
def CriticalValue():
    critical_value_list = []


    user_input = int(input("Enter the C.V.\n"))


    for y in range(user_input, 18):


        for x in range(y, 18):
            critical_value = nCx(18, x) * (p ** x) * ((1-p) ** (18 - x))
            critical_value_list.append(critical_value)

        ans = sum(critical_value_list)
        print("\nCritical Value: {0} ; Probability: {1:0.3f}".format(y, ans))
        critical_value_list.clear()


# ---------------------------------------
# List of p values
# ---------------------------------------
def PValueGenerator():
    p_value_list = []
    for x in range(55, 100, 5):
```

```
    p_value_list.append(x/100)


  return p_value_list


# ----------------------------------------
# Beta Values
# ----------------------------------------
def BetaValues(p_value_list):
  P = []
  beta_value_list = []


  for value in p_value_list:


    for x in range(13):
      probability = nCx(18, x) * (value ** x) * ((1 - value) ** (18 - x))
      P.append(probability)


    answer = sum(P)
    beta_value_list.append(answer)
    P.clear()
  return beta_value_list


# ----------------------------------------
# Binomial Probabilities for n = 18
# ----------------------------------------
def BinomialProbabilities():
  P = []
```

```
    X = []

    value = float(input("Please enter a p value between 0.5 to 1: "))

    for x in range(19):
        probability = nCx(18, x) * (value ** x) * ((1 - value) ** (18 - x))
        P.append(probability)
        X.append(x)

    Graph(X, P)

# ---------------------------------------
# The Power of the Test
# ---------------------------------------
def TestOfPower(beta_values, p_value_list):
    power_list = []

    for beta in beta_values:
        power = 1 - beta
        power_list.append(power)
        power = 0

    # Create a curved plot of power vs. p_value_list
    plt.plot(p_value_list, power_list, 'bs')
    plt.axis([0.5, 1, 0, 1.5])
    plt.xlabel("P values")
    plt.ylabel("Power")
    plt.title("Power vs p")
    plt.grid(True)
    plt.show()


# ---------------------------------------
# The One to Rule Them All!! - Main Function
# ---------------------------------------
def main():
    while True:
        PrintMenu()
        user_input = GetMenuChoice()

        if user_input == 1:
            HypothesisStatement()
            continue
        elif user_input == 2:
            x_value_list, probability_list = BinomialDist()
            Graph(x_value_list, probability_list)
            continue
        elif user_input == 3:
            CriticalValue()
            continue
        elif user_input == 4:
            BinomialProbabilities()
            continue
        elif user_input == 5:
            p_value_list = PValueGenerator()
            beta_value_list = BetaValues(p_value_list)
            TestOfPower(beta_value_list, p_value_list)
            continue
        else:
            print("Quitting Program")
            break


main()
```