**Andrew Wang**

**Homework 4**

1. **Code:**

################

# Author: Andrew Wang

# Date: 10/3/2019

# This program uses user define function that accepts an object’s falling time (in seconds)

# as an argument and calls the function in a loop from 1 sec to 10 sec.

#################

#Main function

def main():

#Time (s)

time = 1

print('Time(s) Falling Distance(m)')

print('-----------------------------')

#While loop that increments time by 1 until 10

while time <= 10:

#Inputs time and returns falling distancce

d = falling\_distance(time)

print ('%d\t%.2f' %(time, d))

time = time + 1

#User defined function for calculating falling distance

def falling\_distance(time):

#Graititional constant

g = 9.8

#Calculation of falling distance

d = 0.5 \* g \* (time \*\* 2)

#Return calculation of distance

return d

main()

**Output:**

**A screenshot of a cell phone

Description automatically generated**

1. **Code**

################

# Author: Andrew Wang

# Date: 10/3/2019

# This program uses user define function determines input of two integers and determining which

# one is larger of the two.

#################

#Main function

def main():

#Execute user defined function

max\_udf()

#User defined function for determining two intergers and which one is larger

def max\_udf():

#Allows user input for first and second intergers

first\_int = float(input('Enter the fist interger: '))

second\_int = float(input('Enter the second interger: '))

#If statement that determines if two values are intergers and which one is larger

if (first\_int % 1) == 0 and (second\_int % 1) == 0:

if first\_int > second\_int:

print('%d is greater.' %first\_int)

elif second\_int > first\_int:

print('%d is greater.' %second\_int)

else:

if (first\_int % 1) != 0:

print('%.1f is not an interger.' %first\_int)

elif (second\_int % 1) != 0:

print('%.1f is not an interger.' %second\_int)

main()

**Output:**

**A screenshot of a cell phone

Description automatically generated**

1. **Code:**

################

# Author: Andrew Wang

# Date: 10/3/2019

# This program uses user define function allows user input 5 test scores and the program will determine

# the letter grade of the score and the average.

#################

#Main function

def main():

#Initialize variables

total\_score = []

grade = []

#For loop that only allows 5 input

for i in range(5):

total\_score.append(get\_valid\_score())

average = calc\_average(sum(total\_score))

grade.append(determine\_grade(total\_score[i]))

for i in range(5):

print('The letter grade for %d is %c' %(total\_score[i], grade[i]))

print('The average test score is %.2f' %(average))

#User defined function which allows user to input test scores

def get\_valid\_score():

count = 5

for i in range(count):

score = float(input('Enter a score: '))

if score >= 0 and score <= 100:

return score

else:

print('Invalid Input. Please try again')

#User defined function which calculates average of five test scores

def calc\_average(total\_score):

average = total\_score / 5

return average

#User defined function which assigns letter grade to a score

def determine\_grade(score):

if score >= 90 and score <= 100:

letter\_grade = 'A'

elif score >= 80 and score <= 89:

letter\_grade = 'B'

elif score >= 70 and score <= 79:

letter\_grade = 'C'

elif score >= 60 and score <= 69:

letter\_grade = 'D'

elif score < 60:

letter\_grade = 'F'

return letter\_grade

main()

**Output:**

**A screenshot of a cell phone

Description automatically generated**

1. **Code:**

################

# Author: Andrew Wang

# Date: 10/3/2019

# This program finds prime numbers between 1 to 100f

#################

#Main function

def main():

#Initialize variables

index = 0

prime = 1

#While loop that calculates prime number from 1 to 100

while index < 100:

prime\_stat = is\_prime(prime)

#If statement that shows if inputted number is a prime or not

if prime\_stat == 'True':

print('%d is a prime number.' %prime)

elif prime\_stat == 'False':

#print('%d is not a prime number.' %prime)

nothing = 0

prime += 1

index += 1

#User defined function that determines if a number is prime or not

def is\_prime(prime):

#If statement that determins a number is 1 or 2, else find if prime or not

if prime == 1 or prime == 2:

prime\_stat = 'True'

return prime\_stat

elif prime != 1 or prime != 2:

#For loop that checks number from 2 to itself - 1

for check\_prime in range(2, prime, 1):

#If statement that determines a number is prime or not

if prime % check\_prime == 0:

prime\_stat = 'False'

return prime\_stat

else:

prime\_stat = 'True'

return prime\_stat

main()

**Output:**

**A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated**

1. **Code:**

################

# Author: Andrew Wang

# Date: 10/3/2019

# This program uses durtle graphics to draw a Hypnotic Pattern with width between two rings as 10.

#################

import turtle

#Initialize variable

index = 1

#Setting up turtle

turtle.pencolor('black')

turtle.forward(1)

turtle.setheading(90)

turtle.forward(1)

#While loop that increases length by 10 for each side

while index <= 48:

turtle.setheading(90)

turtle.forward(10\*index)

index += 1

turtle.setheading(180)

turtle.forward(10\*index)

index += 1

turtle.setheading(270)

turtle.forward(10\*index)

index += 1

turtle.setheading(0)

turtle.forward(10\*index)

index += 1

turtle.setheading(90)

turtle.forward(10\*49)

turtle.done()

**Output:**

**A picture containing building

Description automatically generated**

1. **Code:**

################

# Author: Andrew Wang

# Date: 10/3/2019

# This program uses turtle graphics with a loop and drasws a Start Pattern with a user input number of corners

#################

import turtle

#Allows user to input amount of corners

corners = int(input('Enter the number of corners: '))

#Calculation of angle

angle = 360 / corners

#Setting up turtle

turtle.setup(2000,2000)

turtle.pencolor('red')

turtle.fillcolor('green')

turtle.goto(0,0)

turtle.begin\_fill()

#For loop that draws the shape from 0 to corner - 1

for i in range(0, corners):

turtle.setheading(i \* (180 - angle))

turtle.forward(200)

turtle.setheading(0)

turtle.end\_fill()

turtle.done()

**Output:**

A close up of a logo

Description automatically generated