**CIS-298 Intro to Python**

**With Professor Robert Mann**

**HW #3**

**Student: Demetrius Johnson**

**29 January 2023**

**Due: 31 January 2023 at 4pm**

#Submit your homework in a report named using the format lastname\_firstname\_HWnumber.

#For each of the 20 questions, enter question number, copy/paste code that solves the question, followed by a snippet of output demonstrating your code works

# #p58, 1-10

## #1. Assume the days of the week are numbered 0,1,2,3,4,5,6 from Sunday to Saturday. Write a program that asks a

day number, and prints the day name (a string).

day\_of\_week = int(input("Enter a day of the week (0 = Sunday, 6 = Saturday): "))

DAYS\_list = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"]

print(DAYS\_list[day\_of\_week])

## #2. You go on a wonderful holiday (perhaps to jail, if you don’t like happy exercises) leaving on day number 3

#(a Wednesday). You return home after 137 sleeps. Write a general version of the program which asks for the

#starting day number, and the length of your stay, and it will tell you the name of day of the week you will return on

day\_of\_week = int(input("Enter a day of the week that you are leaving on (0 = Sunday, 6 = Saturday): "))

length\_of\_stay = int(input("Enter the number of days you will be away: "))

DAYS\_list = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"]

print("You left on: ", DAYS\_list[day\_of\_week])

day\_of\_week = (day\_of\_week + (length\_of\_stay % 7)) % 7

print("You return on: ", DAYS\_list[day\_of\_week])

## #3. Give the logical opposites of these conditions

### #(a) a > b

print("logical opposite of a > b --> a >= b")

### #(b) a >= b

print("logical opposite of a >= b --> a < b")

### #(c) a >= 18 and day == 3

print("logical opposite of a >= 18 and day == 3 --> a < 18 or day != 3")

### #(d) a >= 18 and day != 3

print("logical opposite of a >= 18 and day != 3 --> a < 18 or day == 3")

## #4. What do these expressions evaluate to?

### #(a) 3 == 3

print("#(a) 3 == 3 --> TRUE")

### #(b) 3 != 3

print("#(b) 3 != 3 --> FALSE")

### #(c) 3 >= 4

print("#(c) 3 >= 4 --> FALSE")

### #(d) not (3 < 4)

print("#(d) not (3 < 4) --> not (TRUE) --> FALSE")

## #5. Complete the truth table:

print("\n#5 Complete this truth table:")

print("p q r (not (p and q)) or r")

print("F F F ? --> T")

print("F F T ? --> T")

print("F T F ? --> T")

print("F T T ? --> T")

print("T F F ? --> T")

print("T F T ? --> T")

print("T T F ? --> F")

print("T T T ? --> T")

## SCREENSHOTS FOR #’S 1-5

Text

Description automatically generated

## #6. Write a program which is given an exam mark, and it returns a string — the grade for that mark — according to

#this scheme:

#Mark Grade

#>= 75 First

#[70-75) Upper Second

#[60-70) Second

#[50-60) Third

#[45-50) F1 Supp

#[40-45) F2

#< 40 F3

#Test your code by printing the mark and the grade for all the elements in this list:

numbers = [83, 75, 74.9, 70, 69.9, 65, 60, 59.9, 55, 50, 49.9, 45, 44.9, 40, 39.9, 2, 0]

print("\n#6. Exam Letter Grade Calculator:")

for grade in numbers:

if grade >= 75:

print(grade, " = First")

elif grade >= 70:

print(grade, " = Upper Second")

elif grade >= 60:

print(grade, " = Second")

elif grade >= 50:

print(grade, " = Third")

elif grade >= 45:

print(grade, " = F1 Supp")

elif grade >= 40:

print(grade, " = F2")

else:

print(grade, " = F3")

## #7. Write a program which, given the length of two sides of a right-angled triangle, returns the length of the hypotenuse.

#(Hint: x \*\* 0.5 will return the square root.)

L1 = float(input("Enter the length of one leg of a right triangle: "))

L2 = float(input("Enter the length of the other leg of the same right triangle from above: "))

#Use Pythagorium Theorem L1^2 + L2^2 = H^2 --> H = sqrt(L1^2 + L2^2):

print("The length of the Hypotenuse of the right triangle is: ", "\b", (L1\*\*2 + L2\*\*2)\*\*0.5 )

## #8. Write a program which, given the length of three sides of a triangle, will determine whether the triangle is right-angled.

#Assume that the third argument to the function is always the longest side. It will return True if the triangle is right-angled, or False otherwise

print("Input the length of the 3 sides of a triangle, inputting the longest side last: ")

S1 = float(input("Side 1: "))

S2 = float(input("Side 2: "))

S3 = float(input("Side 3 (longest side --> possible hypotenuse): "))

#S1 and S2 == the possible legs of the possibly right triangle

#S3 == the possible hypotenus of a possibly right triangle, also the longest side of any right triangle

#S3^2 = S1^2 + S2^2 (Pythagorean Theorem)

#S3^2 - (S1^2 + S2^2) = 0

#use a precision variable to compare (with some set precision) how close the subtraction is to 0 (up to some number of decimal places)

precision = float(1e-7)

print("Is it a right triangle?: " , abs(S3\*\*2 - (S1\*\*2 + S2\*\*2)) < precision )

## SCREENSHOTS FOR #’S 6-8

Text

Description automatically generated

## #9. Extend the above program so that the sides can be given to the function in any order.

print("Input the length of the 3 sides of a triangle, in any order: ")

S1 = float(input("Side 1: "))

S2 = float(input("Side 2: "))

S3 = float(input("Side 3: "))

sorted\_sides = sorted([S1, S2, S3]) #sort the sides of triangle in ascending order

#S1 and S2 == the possible legs of the possibly right triangle

#S3 == the possible hypotenus of a possibly right triangle, also the longest side of any right triangle

#S3^2 = S1^2 + S2^2 (Pythagorean Theorem)

#S3^2 - (S1^2 + S2^2) = 0

#use a precision variable to compare (with some set precision) how close the subtraction is to 0 (up to some number of decimal places)

precision = 1e-7

print("Is it a right triangle?: " , abs(sorted\_sides[2]\*\*2 - (sorted\_sides[0]\*\*2 + sorted\_sides[1]\*\*2)) < precision )

## #10 [why floating point arithmetic is sometimes innacuarte]...

import math

a = math.sqrt(2.0)

print(a, a\*a)

print(a\*a == 2.0)

## SCREENSHOTS FOR #’S 9 & 10

Text

Description automatically generated

# #p60. 1 and 4

## #1. Write a program that prints We like Python's turtles! 1000 times.

for \_ in range(1000):

print("We like Python's turtles!")

Text

Description automatically generated

## #4. Assume you have the assignment numbers = [12, 10, 32, 3, 66, 17, 42, 99, 20]

numbers = [12, 10, 32, 3, 66, 17, 42, 99, 20]

### #(a) Write a loop that prints each of the numbers on a new line.

for element in numbers:

print(element)

### #(b) Write a loop that prints each number and its square on a new line.

for element in numbers:

print(element, "; element^2--> ", element\*\*2)

#(c) Write a loop that adds all the numbers from the list into a variable called total. You should set the total

#variable to have the value 0 before you start adding them up, and print the value in total after the loop

#has completed.

total = 0

for element in numbers:

total += element

print("sum of elements = ",total)

### #(d) Print the product of all the numbers in the list. (product means all multiplied together)

total = 1 #initialize as 1 since we are doing product of all numbers in the list

for element in numbers:

total \*= element

print("product of elements = ",total)

## Screenshots for #4 (a – d)

Text

Description automatically generated

# #p61. 1-7 (see page 52 for Newton’s method), 13

numbers = [12, 10, -32, 3, 66, -17, 42, 99, -20] #I will use this list to test the questions 1-7 from page 61

## #1. Write a program to count how many odd numbers are in a list.

odd\_elements = 0

for element in numbers:

if (element % 2) == 1:

odd\_elements += 1 #track number of odd elements

print("Number of odd elements in the list: ", odd\_elements)

## #2. Sum up all the even numbers in a list.

even\_elements = 0

sum\_even\_elements = 0

for element in numbers:

if (element % 2) == 0:

even\_elements += 1 # track number of even elements

sum\_even\_elements += element #sum even elements

print("Sum of even elements in the list: ", sum\_even\_elements)

## #3. Sum up all the negative numbers in a list.

sum\_negative\_elements = 0

for element in numbers:

if element < 0:

sum\_negative\_elements += element

print("Sum of negative elements in the list: ", sum\_negative\_elements)

## #4. Count how many words in a list have length 5.

word\_list = ["TEST1", "TEST2", "TESTthree", "TESTfour"]

num\_words\_len\_5 = 0 #use this to track number of words with length 5

for word in word\_list:

if len(word) == 5:

num\_words\_len\_5 += 1

## #5. Sum all the elements in a list up to but not including the first even number. (What if there is no even number?)

#if no even numbers, simply add all elements

numbers = [0, 1, 3, 5, 7, 9, 10, 11, 13, 15]

sum\_elements\_notFirstEven = 0 #use to track sum

for element in numbers:

if (element % 2 == 1) or (element == 0): #check if element is even

sum\_elements\_notFirstEven += element

else:

break #break if we strike an even element, which will also be the first even

element

print("sum of elements up to and not including first even element: ", sum\_elements\_notFirstEven)

## #6. Count how many words occur in a list up to and including the first occurrence of the word "sam". (What if

#"sam" does not occur?)

#if sam does not occur, all of the words will simply be counted

word\_list = ["TEST1", "TEST2", "sam", "TESTfour"]

num\_words\_uptoand\_sam = 0 #use to track sum

for word in word\_list:

if word != "sam":

num\_words\_uptoand\_sam += 1

else:

num\_words\_uptoand\_sam += 1

break

print("number of words in list up to and including sam: ", num\_words\_uptoand\_sam)

## Screenshots for #’S 1-6

Text

Description automatically generated

## #7. Add a print function to Newton’s sqrt algorithm that prints out better each time it is calculated. Call your

#modified program with 25 as an argument and record the results.

print("Newton's algorithm: testing approximating sqrt(25) (threshold is 0.001)")

n = 25

threshold = 0.001

approximation = n/2 # Start with some or other guess at the answer

iteration = 0

while True:

better = (approximation + n/approximation)/2

if abs(approximation - better) < threshold:

print("Iteration",iteration,": ",better)

break

approximation = better

print("Iteration",iteration,": ",approximation)

iteration += 1

Text

Description automatically generated

## #13. Write a program that counts the number of even digits in n.

user\_num\_string = str(input("Input a number, and the number of even digits in the number will be calculated: "))

num\_even\_digits = 0

for digit in user\_num\_string:

if int(digit) % 2 == 0:

num\_even\_digits += 1

print("Number of even digits in your number is: ", num\_even\_digits)

Text

Description automatically generated