[redacted]

Test Information

Description	This lab consists of three parts and contain questions related to Python. This is the second part of Lab 2 and will be automatically graded by LMS. Your highest grade for this part by the submission deadline will be considered.
Instructions	
Multiple Attempts	This Test allows multiple attempts.
Force Completion	This Test can be saved and resumed later.

QUESTION 1

0.25 points Saved

Before we set off into the world of loops, here are a few questions to get warmed up.
Follow the multiple-choice prompt.
Which one of the following is a type of loop in Python.

- Do...Until
- For Each
- For
- While ... end

QUESTION 2

0.5 points Saved

Now let us have a look at a few samples of code.

What is the value of variable 'a' after running the following code ?

a = 0

for ii in range(0,10):

a+= 1

10

QUESTION 3

0.5 points Saved

What is the value of variable 'a' after running the following code?

a = 10;

for ii in range(0,11):

a+= ii

65

QUESTION 4

0.5 points Saved

What is the value of variable 'a' after running the following code ?

a = 10

for ii in range(0,10):

a+= 1

20

QUESTION 5

0.5 points Saved

a=[1]

for ii in range(1,7,2):

a.append(ii)

What is value of variable 'a' after running the above code ? Write your answer using square brackets, comma (,) separators and NO spaces. For example: [2,4,6,7,8,9]

[1,1,3,5]

QUESTION 6

0.5 points Saved

a = [0]*10

for ii in range(1,11):

if (ii%2)==0:

a[ii-1]=ii**2)

What will be the value of 'a' after running the above code? Enter your answer using square brackets, comma (,) separators and NO spaces. For example: [1,2,3]

[0,4,0,16,0,36,0,64,0,100]

QUESTION 7

1 points Save Answer

Okay, so now let's move onto some coding.

Write a function called 'simpleLoop' which accepts a single argument (a list of numbers). It should iterate through all the elements in the list and return a list containing the odd numbers only.
E.g. simpleLoop($[1, 2, 3, 4, 5]$) = $(1, 3, 5)$

What would be the output of this function for the following inputs. Enter your answer without any blank spaces just as the example above.

A. simpleLoop($(0, 1, 33, 7, 2)$) = _____.

B. simpleLoop($(2, 2, 2, 2, 2)$) = _____.

C. simpleLoop($[1, 1, 1, 1, 1]$) = _____.

QUESTION 8

1.25 points Save Answer

Write function **sumList** which takes a list of numbers as an input and returns the sum of the numbers in the list.
For instance, sumList($[1, 1, 1]$) = 3

What is the output of your sumList function for the inputs below:

A. sumList($[1, 2, 3, 4, 5]$) = _____

B. sumList($(0, -1, -3, 7, 2)$) = _____

C. sumList($(2, 2, 2, 2, 2)$) = _____

D. sumList($[1, 1, 1, 1, 1]$) = _____

QUESTION 9

1.25 points Save Answer

Now let's try using a **for** loop with a slightly bizarre iteration range. Write a function called **backwardsLoop** which accepts a single list which could contain anything. It should return a list which starts with the last element and contains every 'other' element backwards. This is, every second element moving backwards through the list.
E.g. backwardsLoop($[5, 4, 3, 2, 1]$) = $(1, 3, 5)$

Note: they are square brackets around the lists. They may look slightly different due to formatting issue

What would be the output of your function backwardsLoop for the following inputs?
please use double quotes for answers to show string if needed

A. backwardsLoop($[34, \text{"}Ace\text{"}, -3, True]$) = _____.

B. backwardsLoop($[1, 2, 3, 4, 5]$) = _____.

C. backwardsLoop([ ]) = _____.

QUESTION 10

1.25 points Save Answer

Now let's have one more go at a **for** loop.
Write a function called **oddThree** which accepts a single list of integers as an argument. It should return the first number in the list which is divisible by 3. If such a number does not exist, return 0
For instance, oddThree($[1, 2, 4, 5, 6, 8, 3]$) = 6
Hint: Once you have found the first number with these properties, you don't need to continue through the list. Simply return the value immediately.
What will be the output if the list provided as an input is $[1, 2, 5, 13, 120, 3, 16, 21]$

QUESTION 11

0.25 points Save Answer

The **for** loop is used when we know in advance how many times we need to loop. The **while** loop is used when we don't know in advance how many times we need to loop. Let us see first how **while** loop can replace a **for** loop (even when we know the number of iterations in advance).

```
# to print the text 'Hello, how are you?' 10 times using the for loop
for ii in range(0,10):
    print('Hello, how are you?');

# to do the same using a while loop
count = 0
while count < 10:
    print('Hello, how are you?')
    count+=1   # do not forget to do this, otherwise the loop will run forever
```

From the example above, we see that it is better to reserve the while loop to the cases where the number of times to loop is unknown.
Exercise: Write a Python script **tempConvert** that would ask the user to enter a non-negative temperature value in Celsius and print the equivalent value in Fahrenheit. It should then ask for another temperature value in Celsius and repeat this process until the user enters the value -1.

F = C × 9/5 + 32

where **F** is the temperature in Fahrenheit and **C** is the temperature in Celsius. To get input entered on the keyboard, use the Python **input** function, e.g.,
temperature = input('Please enter the temperature in Celsius: ')

The variable temperature is a string. You will have to convert it to an integer?

Which is better for this situation, *for* or *while* loop and why?

- A **for** loop, because the user can decide how many loop iterations.
- A **for** loop, because the number of loop iterations is known in advance.
- A **while** loop, because the number of loop iterations is unknown in advance.
- A **while** loop, because we need to complete a calculation inside the loop.

QUESTION 12

2 points Save Answer

Take any natural number n. If n is even, divide it by 2 to get n/2. If n is odd, multiply it by 3 and add 1 to obtain 3n+1 (This is called the Hailstone sequence). Set the new number as n and repeat this process until n = 1.
Write a function called 'hailstone' which takes a single argument n and returns the number of steps required until we reach 1.
What is the output of your function when n=81

_____

QUESTION 13

2 points Save Answer

The exponential function e to the power of x can be defined as the equation below.

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots$$

The series is infinite but is is usually approximated to the first few terms. Write your own function expoApprox w hich takes a single argument x and returns the exponential function e to the power of x. The calculation should be approximated to the first few terms till $\frac{x^n}{n!}$ becomes smaller than 0.01. What is the output of your function for the following values of x. Round your answer to 3 decimal places of accuracy. Remember to include the last term when the term becomes smaller than 0.01.
Note: You have to be careful with the last calculated term. Depending on the formulation of your loop, you may need to add or remove it from the total sum to match your result. This will help you to understand the loop in better manner.

A. expoApprox (1) = _____.

B. expoApprox (4) = _____.

QUESTION 14

8.25 points Save Answer

Write code for an interactive Guessing Game. The code should do the following:-
1. Predefine a random integer 'N' between 1 and 100.
2. Ask the user/ player to guess the number in 5 tries.
3. If the player guesses a number smaller than 'N' then ask the player to guess a bigger number.
4. If the player guesses a number bigger than 'N' then ask the player to guess a smaller number.
5. If the player guesses the correct number then tell the player that s/he has won in 'x' tries (The actual number of tries in which the player guessed the number) and terminate the game.
6. Do not use 'break' to terminate the game.
Hint: Your code should terminate when either the player has guessed the number or the number of tries are over.

Question: Which type of loop did you prefer for this excercise?

- A **while** loop because it allows me to put two terminating conditions simultaneously.
- A **for** loop because I know the total number of tries (five).
- A **for** loop because I can program the number of tries in the loop condition use the break command to terminate when the player has guessed the number.
- A **while** loop because it is easy to terminate.

QUESTION 15

1 points Save Answer

In your guessing game the player is allowed to guess the random number in 5 tries. What are the minimum number of tries you can allow a player if you want him/ her to win every time with a 100% probability.

_____

v Question Completion Status:

QUESTION 16

2 points Save Answer

Warning, this is easily the hardest question provided thus far.
Let's say that we have a contest the details of which we don't care about apart from one simple rule. "If a contestant earns a score equal to or greater than the k-th place finisher, they will advance to the next round if their own score is greater than 0"
So write a function **nextRound(k,scores)** which will count how many contestants will progress to the next round. There will be no more than 20 contestants, and $1 <= k <= 20$. The scores will be given in an order, but contestants can have the same score.
E.g. nextRound($5, [10, 9, 8, 7, 6, 6, 5, 4]$) = 7

E.g. nextRound($2, [0, 0, 0, 0]$) = 0

This one has a lot of edge-cases to think about. Good luck.
What is the output of your function for the following inputs:

A. nextRound($2, [1, 1, 1, 1]$) = _____.

B. nextRound($20, [20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]$) = _____.

C. nextRound($1, [5, 4, 3, 2]$) = _____.

D. nextRound($1, [10]$) = _____.