Quiz: Chapter 5

Question 1

1/1 point (graded)

What is wrong with this Python loop:

```
n = 5
while n > 0 :
    print(n)
print('All done')
```

- (●) This loop will run forever ✔
- ( ) The **print('All done')** statement should be indented four spaces
- ( ) There should be no colon on the **while** statement
- ( ) **while** is not a Python reserved word

Submit

Question 2

1/1 point (graded)

What does the **break** statement do?

- Resets the iteration variable to its initial value
- Exits the program
- Jumps to the "top" of the loop and starts the next iteration
- (selected) Exits the currently executing loop ✔

Submit

Question 3

1/1 point (graded)

What does the **continue** statement do?

- Resets the iteration variable to its initial value
- Exits the currently executing loop
- Exits the program
- (selected) Jumps to the "top" of the loop and starts the next iteration ✔

Submit

Question 4

1/1 point (graded)

What does the following Python program print out?

```
tot = 0
for i in [5, 4, 3, 2, 1] :
    tot = tot + 1
print(tot)
```

- ○ 10
- ○ 0
- ○ 15
- ◉ 5 ✔

Submit

Question 5

1/1 point (graded)

What is the *iteration* variable in the following Python code:

```
friends = ['Joseph', 'Glenn', 'Sally']
for friend in friends :
    print('Happy New Year:',  friend)
print('Done!')
```

- (●) friend ✔
- ( ) Sally
- ( ) Glenn
- ( ) Joseph

Submit

Question 6

1/1 point (graded)

What is a good description of the following bit of Python code?

```
zork = 0
for thing in [9, 41, 12, 3, 74, 15] :
    zork = zork + thing
print('After', zork)
```

- ○ Find the smallest item in a list
- ○ Count all of the elements in a list
- ○ Compute the average of the elements in a list
- ◉ Sum all the elements of a list ✔

Submit

Question 7

1/1 point (graded)

What will the following code print out?

```
smallest_so_far = -1
for the_num in [9, 41, 12, 3, 74, 15] :
    if the_num < smallest_so_far :
        smallest_so_far = the_num
print(smallest_so_far)
```

Hint: This is a trick question and most would say this code has a bug - so read carefully

- (●) -1 ✔
- ( ) 74
- ( ) 42
- ( ) 3

Answer
Correct:
Look closely at the if statement. If the variable starts out as "-1", will the if statement ever be true given the values that the loop will iterate through?

Submit

Question 8

1/1 point (graded)

What is a good statement to describe the **is** operator as used in the following if statement:

```
if smallest is None :
    smallest = value
```

- ○ The if statement is a syntax error
- ○ Looks up 'None' in the **smallest** variable if it is a string
- ○ Is true if the **smallest** variable has a value of -1
- ◉ matches both type and value ✔

Answer
Correct:
The **is** operator is stronger than the equality operator (==) as it insists on matching the two values exactly including type. This simple example shows the difference: **>>> 1.0 == 1 True >>> 1.0 is 1 False** While 1.0 is the same value after the integer 1 is converted to floating point, the **is** operator does no conversion and so the two values do not match. The **is** operator is best used on small constant values like small integers, True, False, and None. The **is** operator should not be used with large numeric values or strings - these values should be compared with the == operator.

Submit

Question 9

1/1 point (graded)

Which reserved word indicates the start of an "indefinite" loop in Python?

- indef
- break
- for
- def
- while ✔

Submit

Question 10

1/1 point (graded)

How many times will the body of the following loop be executed?

```
n = 0
while n > 0 :
    print('Lather')
    print('Rinse')
print('Dry off!')
```

You can add an optional tip or note related to the prompt like this.

- This is an infinite loop
- 5
- 1
- 0 ✔

Submit