Activity 4

Pattern thinking: Logic in Layers

Applied Python Programming with AI and Raspberry Pi Interfaces

Instructor: Dr. Vikas Thammanna Gowda Semester: ABCD 20YX

Assign: *TBD*

Name: ____

1. True or False

Statement	T/F
A function must always have a return statement.	
A variable defined within a function is limited to that function and is not accessible outside of it.	
A break statement and a return statement can be used interchangeably.	
Functions can have multiple return statements, but only one of them will be executed during the function's execution.	
A function definition must be followed by parentheses, even if it has no parameters.	
You can assign a function to a variable and then call it through that variable.	
You cannot call a function before it has been defined in the script.	

2. Answer the following

a.

```python
def Add():
    total = 10 + 20
    print(total)
```

Write statement to call the function:

b.

```python
def Add(X, Y):
    total = X + Y
    print(total)
```

Write statement to call the function with 5 and 10 as the arguments:

b.

```
1 def check(num):
2     if ((num % 2) == 0):
3         print("Hello")
4         return True
5         print("Hi")
6     else:
7         return False
```

Which line number code will never execute regardless of the value passed to the parameter num?

3. What is the output of the following code snippets?

a.

```
def cube(num):
        print(num * num * num)

print("Calling a function below")
cube(2)
```

Output:

b.

```
def cube(num):
        print(num * num * num)

result = cube(2)
print(result)
```

Output:

c.

```python
def sum_total(x, y =10, z = 20):
        sum = x + y + z
        print(sum)
```

Output:

Case 1: `sum_total(100)`

Case 2: `sum_total(10, 30)`

Case 2: `sum_total(1, 7, 99)`

d.

```python
def square(x):
    return x * x

def apply(func, value):
    return func(value)

print(apply(square, 5))
```

Output:

e.

```python
def countDracula():
     count = 0
     for num in range(10):
         count += 1

print(count)
```

Output:

f.

```
def Func_1(num_1):
         num_1 *= 2
    num_1 = Func_2(num_1)
    print(num_1)

def Func_2(value):
    return value + 10

Func_1(2)
add_10 = Func_2(2)
print(add_10)
```

Output:

g.

```
for i in range(4):
    for j in range(i):
        print(i + j, end=" ")
```

Output:

g.

```
for i in range(1, 4):
    for j in range(i):
        print("*", end="")
    print()
```

Output: