USN | | | | | | | | | | |

Internal Assessment Test 1 – Nov 2021

Sub:	Application Development Using Python	Sub Code:	18CS55	Branch:	ISE

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1a)	Explain the use of following functions/methods with code snippets **(4*1=4 Marks)** **Use-0.5 Marks** **Code-0.5 Marks** **i) input()** **input() is used to read the data from the console and provide the data in the same type.** **Example** **X =input('Enter x') #10** **Y =input('Enter y') #20** **print('sum = ',X+Y)** **ii) type()** **To know the type of the variable. Ex: type(x) <class 'int'>** **iii) format()** **The format () method formats the specified value(s) and insert them inside the string's placeholder.** **txt1 = "My name is {fname}, I'm {age}".format(fname = "John", age = 36** **iv) remove()** **The remove() method is passed the value to be removed from the list it is called on.Ex:** **>>> spam = ['cat', 'bat', 'rat', 'elephant']** **>>> spam.remove('bat')** **>>> spam** **['cat', 'rat', 'elephant']**	[4]	CO1	L3
1b)	Identify the blocks in this code**:** **(4*0.5=2 Marks)** **if (i == 10):** block 1 **# First if statement** if (i < 15): block 2 print("i is smaller than 15") # Nested - if statement # Will only be executed if statement above # it is true if (i < 12): block 3 print("i is smaller than 12 too") else: block 4 print("i is greater than 15") **4 blocks:** **If 3** **Else** 1	[2]	CO1	L2
1c)	Explain the purpose of following keyword with code snippets: **(4*1= 4 marks)** 1. def We can also define our own functions that accept arguments	[4]	CO1 &CO2	L3

```
❶ def hello(name):
❷     print('Hello ' + name)

❸ hello('Alice')
  hello('Bob')
```

2. None

```
>>> spam = print('Hello!')
Hello!
>>> None == spam
True
```

In Python there is a value called None, which represents the absence of a value. None is the only value of the None Type data type.

3. not in

We can determine whether a value is or isn’t in a list with the in and not in operators.

in and **not in** are used in expressions and connect two values: a value to look for in a list and the listwhere it may be found and these expressions will evaluate to a Boolean value

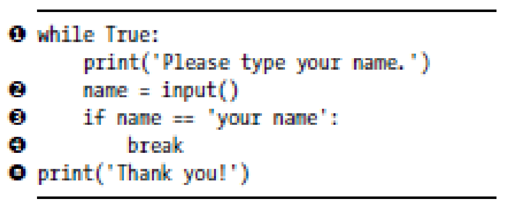
```
>>> 'howdy' in ['hello', 'hi', 'howdy', 'heyas']
True
>>> spam = ['hello', 'hi', 'howdy', 'heyas']
>>> 'cat' in spam
False
>>> 'howdy' not in spam
False
>>> 'cat' not in spam
True
```

4. del The del statement will delete values at an **index** in a list.
All of the values in the list after the deleted value will be moved up one index.Ex:

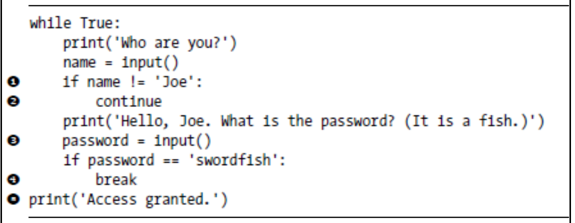
```
>>> spam = ['cat', 'bat', 'rat', 'elephant']
>>> del spam[2]
>>> spam ['cat', 'bat', 'elephant']
>>> del spam[2]
```

2a) What is the difference between break and continue statement? Explain with programs **(2 Marks +2 Marks)** [4] CO1 L2

break	continue
1.There is a shortcut to getting the program execution to break out of a while loop’s clause early 2.If the execution reaches a break statement, it immediately exits the while loop’s clause.	1.Like break statements, continue statements are used inside loops 2.When the program execution reaches a continue statement, the program execution immediately jumps back to the start of the loop and reevaluates the loop’s condition.



```
❶ while True:
      print('Please type your name.')
❷     name = input()
❸     if name == 'your name':
❹         break
❺ print('Thank you!')
```



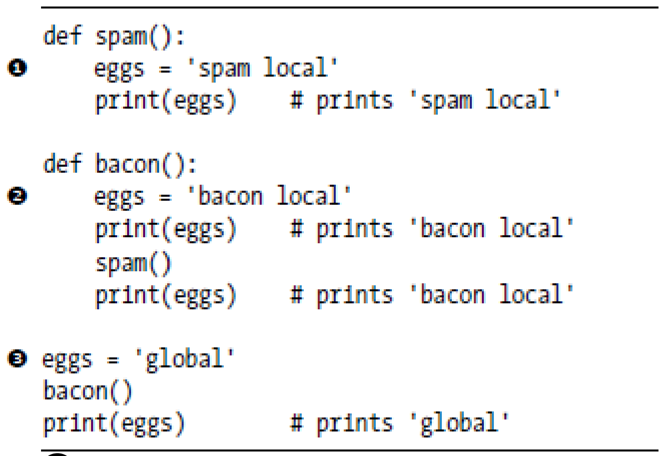
```
  while True:
      print('Who are you?')
      name = input()
❶     if name != 'Joe':
❷         continue
      print('Hello, Joe. What is the password? (It is a fish.)')
❸     password = input()
      if password == 'swordfish':
❹         break
❺ print('Access granted.')
```

2b) Explain the concept of Local scope and global scope of the variables in Python with example code snippets. **(3*2=6 Marks)** [6] CO1 L2

Explanation -3

Example-3

- ➢ Parameters and variables that are assigned in a called function are said to exist in that function's *localscope.*
- ➢ Variables that are assigned outside all functions are said to exist in the *global scope*.
- ➢ A variable that exists in a local scope is called a *local variable*, while a variable that exists in the globalscope is called a *global variable.*
- ➢ A variable must be one or the other; it cannot be both local and global.
- ➢ When a scope is destroyed, all the values stored in the scope's variables are forgotten.
- ➢ There is only one global scope, and it is created when your program begins. When your programterminates, the global scope is destroyed, and all its variables are forgotten.
- ➢ A local scope is created whenever a function is called. Any variables assigned in this function exist withinthe local scope. When the function returns, the local scope is destroyed, and these variables are forgotten.
- ➢ Scopes matter for several reasons:
  1. Code in the global scope cannot use any local variables.
  2. However, a local scope can access global variables.
  3. Code in a function's local scope cannot use variables in any other local scope.
  4. We can use the same name for different variables if they are in different scopes. That is, there can bea local variable named spam and a global variable also named spam.



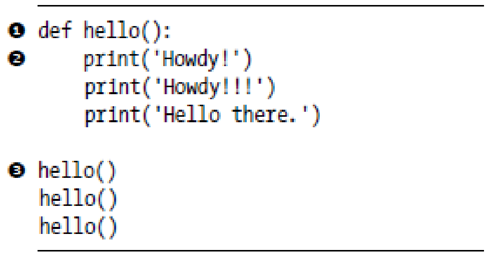
```
  def spam():
❶     eggs = 'spam local'
      print(eggs)    # prints 'spam local'

  def bacon():
❷     eggs = 'bacon local'
      print(eggs)    # prints 'bacon local'
      spam()
      print(eggs)    # prints 'bacon local'

❸ eggs = 'global'
  bacon()
  print(eggs)        # prints 'global'
```

❶ A variable named eggs that exists in a local scope when spam() is called.
❷ A variable named eggs that exists in a local scope when bacon() is called.
❸ A variable named eggs that exists in the global scope.

3(a) Explain the difference between function and methods with examples(2+2) [4] CO1 L2

Explanation and Example (2+2)

- ➢ A function is like a mini-program within a program.



```
❶ def hello():
❷     print('Howdy!')
      print('Howdy!!!')
      print('Hello there.')

❸ hello()
  hello()
  hello()
```

- ➢ Example:
- ➢ The first line is a def statement ❶, which defines a function named hello().
- ➢ The code in the block that follows the def statement ❷ is the body of the function. This code is executedwhen the function is called, not when the function is first defined.
- ➢ The hello() lines after the function ❸ are function calls.
- ➢ In code, a function call is just the function's name followed by parentheses, possibly with some numberof arguments in between the parentheses.

Methods

- A method is the same thing as a function, except it is **"called on" a value.**
- Each data type has its own set of methods.
- The list data type, has several useful methods for finding, adding, removing, andotherwise manipulating values in a list.
- List.append()

3(b) Write a Python Program to check if a given number is Fibonacci number or not? (4+2=6) [6] CO1 L3

Program code=4 marks
Output=2 Marks
Program:

```
def fib(n):
   c=0
   a=1
   b=1
   if n==0 or n==1:
      print("True")
   else:
      while c<n:
         c=a+b
         b=a
         a=c
      if c==n:
         print("True")
      else:
         print("False")
n=int(input())
fib(n)
Input: 13
Output:
True
Input:25
Output:
False
```

5 (a)	Predict the output and justify the answer **(8*0.5= 4 marks)** 1. 22/8 = 2 2. 7.7//7=1 3. (200-70)*10/5 =260 4. not not not False=True 5. -5%-6 =-1 6. (10<=16) and (not True) and (2==2)=False 7. 3**5+(2*10)/5-6 =241 8. (False and True) or True =True	[04]	CO1	L2

(b)	Write a program to generate the random numbers between 1 to 100. if the generated number is odd, add them in the List named ODD_LIST and if it is even number, add them in the list named EVEN_LIST. If the size of the both the lists become 15, stop the generation of numbers and display both the lists.
	Program code:5 Marks
	Output:1 Marks
	even_count, odd_count = 0, 0
	even_list = []
	odd_list = []
	n = int(input("Enter the lower bound of range: "))
	m = int(input("Enter the upper bound of range: "))
	for i in range(n,m+1):
	    if i % 2 == 0:
	        even_count += 1
	        even_list.append(i)
	    else:
	        odd_count += 1
	         odd_list.append(i)
	print("Total even numbers in the range {0} to {1} is {2}".format(n, m,even_count), "and numbers are", even_list)
	print("Total odd numbers in the range {0} to {1} is {2}".format(n,m,odd_count), "and numbers are", odd_list)
		[06]	CO2	L3

6 (a)	Explain with examples the way how indexing and slicing can be done in Lists **(2+2)** **Explanation:2 Marks** **Example:2 Marks** **Indexing:** The integer inside the square brackets that follows the list is called an index. The first value in the list is at index 0, the second value is at index 1, the third value is at index 2, and so on. spam=['cat', 'bat', 'rat', 'elephant'] >>>spam[0] cat **Slicing:** Index is used to get a single value from a list, but a slice can get several values from alist, in the form of a new list.A slice is typed between square brackets, like an index, but it has two integers separatedby a colon. Ex l: >>> spam = ['cat', 'bat', 'rat', 'elephant']	[4]	CO2	L2

```
>>> spam[0:4] ['cat', 'bat', 'rat', 'elephant']
>>> spam[1:3] ['bat', 'rat']
>>> spam[0:-1] ['cat', 'bat', 'rat']
```

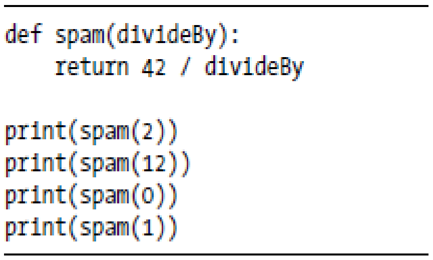
6(b) **What are exceptions in Python? How it can be handled in Python? Exception Handling** **(4+2)** [6] CO1 L2

Explanation:4 Marks
Example:2 Marks

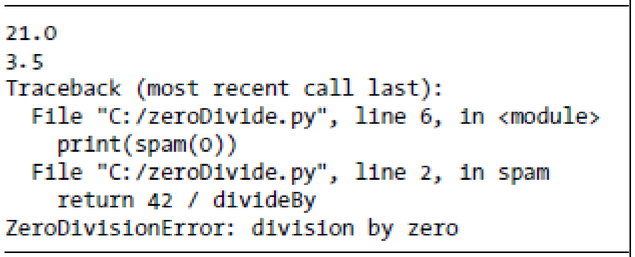
Exception Handling

- If we don't want to crash the program due to errors instead we want the program to detect errors, handlethem, and then continue to run.
- For example,

Program **Output**



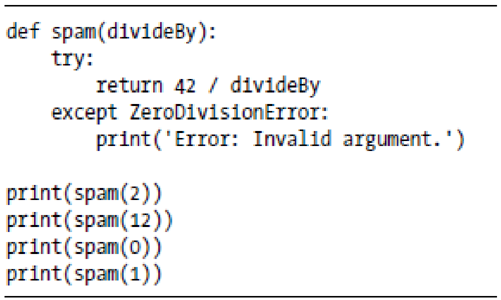
```
def spam(divideBy):
    return 42 / divideBy

print(spam(2))
print(spam(12))
print(spam(0))
print(spam(1))
```



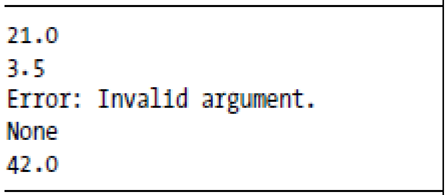
```
21.0
3.5
Traceback (most recent call last):
  File "C:/zeroDivide.py", line 6, in <module>
    print(spam(0))
  File "C:/zeroDivide.py", line 2, in spam
    return 42 / divideBy
ZeroDivisionError: division by zero
```

- A ZeroDivisionError happens whenever we try to divide a number by zero. From the line number givenin the error message, we know that the return statement in spam() is causing an error.
- Errors can be handled with try and except statements.
- The code that could potentially have an error is put in a try clause. The program execution moves to thestart of a following except clause if an error happens.
- We can put the previous divide-by-zero code in a try clause and have an except clause contain code tohandle what happens when this error occurs.

Program **Output**



```
def spam(divideBy):
    try:
        return 42 / divideBy
    except ZeroDivisionError:
        print('Error: Invalid argument.')

print(spam(2))
print(spam(12))
print(spam(0))
print(spam(1))
```



```
21.0
3.5
Error: Invalid argument.
None
42.0
```

any errors that occur in function calls in a try block will also be caught.

HoD Signature CCI signature Course Instructor