SOLUTIONS&SCHEMEIAT1
Date 7th Nov 2022

CourseName: Application Development Using PythonCourseCode:18CS55

Q.No.	Solution Scheme	Marks
1a.	How can we pass parameters in user defined functions? Explain with suitable examples.	1+2+2=5 M
	A function is a mini-program within a program and A parameter is a variable that an argument is stored in when a function is called . The Purpose of user defined function is to group code that gets executed multiple times . [1Marks] How to Pass values to functions (parameters) **Example Code:[2Mark]** def hello(name): print("Hello " + name) hello("Alice") hello("Bob") Output: Hello Alice Hello Bob **Explanation [2Mark]** The definition of the hello() function in this program has a parameter called name. A *parameter* is a variable that an argument is stored in when a function is called. The first time the hello() function is called, it's with the argument 'Alice'. The program execution enters the function, and the variable name is automatically set to 'Alice', which is what gets printed by the print() statement.	
1b	Consider below expressions and identify type of errors occurs. Justify Your Answer (1*5=5marks). a. >>>5 + b. >>>42 + 5 + * 2 c. >>>'Hello, world!' d. >>>'Alice' + 42 e. >>>'Alice'*'Bob'	5M
	a. >>>5+ -Syntax error, is not Grammatically correct if you enter a bad Python instruction, Python won't be able to understand it and will display a Syntax Error error message . b. >>> 42 + 5 + * 2 - Syntax error, This is not correct syntax as two operators are not acceptable in python in order to evaluate expression. So will end up with an syntax error c. >>> 'Hello, world!' – NO Error d. >>> 'Alice' + 42 – Type error TypeError: can only concatenate str (not "int") to str	

	e. >>>'Alice'*'Bob' - TypeError: can't multiply sequence by non-int of type 'str'	
2a	Explain the Three rules to create Variables with Example. also Justify Why is eggs a valid variable name while 100 is invalid?	3+2+1
	Variable Names three rules: **(3Marks)** 1. It can be only one word. 2. It can use only letters, numbers, and the underscore (_) character. 3. It can't begin with a number. Examples of valid and invalid variable names **(2Marks)** Ex. Valid variable names balance currentBalance current_balance _spam SPAM account4 Ex. Invalid Variable Names current-balance (hyphens are not allowed) current balance (spaces are not allowed) 4account (can't begin with a number) 42 (can't begin with a number) total_$um (special characters like $ are not allowed) 'hello' (special characters like ' are not allowed) **(1Mark)** Eggs is Valid variable Name as its one word where as 100 is invalid as variable name can not begin a number	
2b	What do the following expressions evaluate to? **(each ans carries 1 Mark)** 1. (5 > 4) and (3 == 5) 2. not (5 > 4) 3. (5 > 4) or (3 == 5) ***4.*** not ((5 > 4) or (3 == 5))	1*4=4M
	1. (5 > 4) and (3 == 5) T and F=F 2. not (5 > 4) not(T)=F 3. (5 > 4) or (3 == 5) T or F=T 4. not ((5 > 4) or (3 == 5)) not(T or F) not(T)= F	
3a	Write a python program to ask user to enter a number and check if entered number is Prime number or not ?	3+2M
	Program+O/p (3+2Marks)	

```python
# Program to check if a number is prime or not

num = int(input("Enter a number: "))

flag = False

if num > 1:
    # check for factors
    for i in range(2, num):
        if (num % i) == 0:
            flag = True
            break

if flag:
    print(num, "is not a prime number")
else:
    print(num, "is a prime number")
```

```
Enter a number: 4
4 is not a prime number
```

3b Differentiate between the following:
a. del and remove() methods of list.
c. append() and insert() methods of list. 2.5*2=5 M

a. del() and remove() (2.5Marks)

The remove() method is passed the value to be removed from the list it is called on.

>>> spam = ['cat', 'bat', 'rat', 'elephant'] >>>spam.remove('bat')
>>> spam
['cat', 'rat', 'elephant']

The del() method:

This method is similar to the List object's pop() method with one important difference. The del operator removes the item or an element at the specified index location from the list, but the removed item is not returned,

Examples

myList = ["Bran",11,22,33,"Stark",22,33,11]

del myList[2]

myList

OUTPUT: ['Bran', 11, 33, 'Stark', 22, 33, 11]

del myList[1:4]

myList

OUTPUT: ['Bran', 22, 33, 11]

b. append() and insert() methods of list.(2.5Marks)

To add new values to a list, use the append() and insert() methods. Enter the following into the interactive shell to call the append() method on a list value stored in the variable spam:

>>> spam = ['cat', 'dog', 'bat']

>>>spam.append('moose')

	>>> spam
	['cat', 'dog', 'bat', 'moose']
	
	The previous append() method call adds the argument to the end of the list. The insert() method can insert a value at any index in the list. The first argument to insert() is the index for the new value, and the sec- ond argument is the new value to be inserted.
	
	>>> spam = ['cat', 'dog', 'bat']
	
	>>>spam.insert(1, 'chicken')
	
	>>>spam
	['cat', 'chicken', 'dog', 'bat']	

4.a	**Explain Keyword argument? What is use of 'sep' and 'end' argument in print() function**	1+1.5+1.5 =4M
	keyword arguments are identified by the keyword put before them in the function call. Keyword arguments are often used for optional parameters. For example, the print() function has the optional parameters end and sep to specify what should be printed at the end of its arguments and between its arguments (separating them), respectively. **[1 mark]** *1**. The end parameter [1.5 Marks]*** ***The** end **parameter is used to append any string at the end of the output of the print statement in python.*** By default, the print method ends with a **newline**. This means there is no need to explicitly specify the parameter **end** as '**\n**'. Programmers with a background in C/C++ may find this interesting. Ex: print('Hello', end='') print('World') **2. The sep parameter[1.5 Marks]** Sometimes, we may wish to print multiple values in a Python program in a readable manner. This is when the argument **sep** comes to play. The arguments passed to the program can be separated by different values. The default value for **sep** is whitespace. The **sep** parameter is primarily used to format the strings that need to be printed on the console and add a separator between strings to be printed. This feature was newly introduced in Python 3.x version. The below example shows that passing **sep** parameter as whitespace or not passing the **sep** at all doesn't make a difference.	

	Ex: >>> **print('cats', 'dogs', 'mice', sep=',')** cats,dogs,mice	
4b.	**Explain the concept of Local scope and global scope of the variables in Python with example code snippets.**	3 +3M
	▯ 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. ❶ A variable named eggs that exists in a local scope when spam() is called.	

```
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'
```

	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.	
5 a	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**.** **ProgramProgram +Correct O/P(4+2 marks)**	4+2M
	even=0 **odd=0** **evenl=[]** **oddl=[]** **while(even<=14 or odd<=14):** **rand=random.randint(1,100)** **if rand%2==0:** **even=even+1** **if(even<=15):** **evenl.append(rand)** **else:** **odd=odd+1** **if(odd<=15):** **oddl.append(rand)** **print(evenl)** **print(oddl)** [66, 22, 16, 8, 44, 48, 40, 38, 16, 24, 12, 30, 70, 98, 14] [17, 11, 97, 9, 1, 29, 67, 61, 89, 55, 43, 87, 31, 35, 7]	
5b	Discuss what exception below program gives and How will you handle exception in below program. Write a code which handle the exception occur in below code. def spam(divideBy): return 42 / divideBy print(spam(2)) print(spam(12)) print(spam(0)) print(spam(1))	2+2M
	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

```
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))
```

Output

```
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.

6a	**Explain with examples the way how indexing and slicing can be done in Lists.** **Slicing** **Indexing**	2+2M

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']

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

6b	Write a program Python program to check if the number is an Armstrong number or not Program+correct output(4+2 M)	4+2m

	# Python program to check if the number is an Armstrong number or not
	# take input from the user
	num = int(input("Enter a number: "))
	
	# initialize sum
	sum = 0
	
	# find the sum of the cube of each digit
	temp = num
	while temp > 0:
	  digit = temp % 10
	  sum += digit ** 3
	  temp //= 10
	
	# display the result
	if num == sum:
	  print(num,"is an Armstrong number")
	else:
	  print(num,"is not an Armstrong number")	