**UNIT – 3**

**CONTROL FLOW, FUNCTIONS**

**PART A**

**1. Define Boolean expression with example.**

A boolean expression is an expression that is either True or False. The values true and false are called Boolean values.

Example:

>>> 5 == 6

False

True and False are special values that belongs to the type bool; they are not strings.

# 2. Explain modulus operator with example.

The modulus operator works on integers and yields the remainder when the first operand is divided by the second. In Python, the modulus operator is a percent sign (%). The syntax is the same as for other operators:

Example:

>>> remainder = 7 % 3

>>> print remainder

1

So 7 divided by 3 is 2 with 1 left over.

# What is conditional execution?

The ability to check the condition and change the behavior of the program accordingly is called conditional execution.

Example: If statement: The simplest form of if statement is:

# Syntax:

if *condition:*

*statement*

# Eg:

|  |  |
| --- | --- |
| >>> x = 22  >>> if x%2 == 0:  print ('x is even') | Output  x is even |

The boolean expression after ‘if’ is called the condition. If it is true, then the indented statement gets executed. If not, nothing happens.

# What is alternative execution?

A second form of if statement is alternative execution, that is, if …else, where there are two possibilities and the condition determines which one to execute.

# Eg:

|  |  |
| --- | --- |
| >>> x = 21  >>> if x%2 == 0:  print ('x is even')  else:  print ('x is odd') | Output  x is odd |

1. **What are chained conditionals?**

Sometimes there are more than two possibilities and we need more than two branches. One way to express a computation like that is a chained conditional:

# Eg:

|  |  |
| --- | --- |
| x = 10  y = 15  if x < y:  print ('x is less than y')  elif x > y:  print ('x is greater than y')  else:  print ('x and y are equal') | Output  x is less than y |

elif is an abbreviation of “else if.” Again, exactly one branch will be executed. There is no limit on the number of elif statements. If there is an else clause, it has to be at the end, but there doesn’t have to be one.

# Explain while loop with example.

# Eg:

|  |  |
| --- | --- |
| >>> def countdown(n):  while n > 0:  print ( n )  n = n-1  print ('Blastoff!') | Output  >>> countdown(5)  5  4  3  2  1  Blastoff! |

More formally, here is the flow of execution for a while statement:

1. Evaluate the condition, yielding True or False.
2. If the condition is false, exit the while statement and continue execution at the next statement.
3. If the condition is true, execute the body and then go back to step 1

# Explain ‘for loop’ with example.

The general form of a for statement is

# Syntax:

for variable in sequence:

code block

# Eg:

|  |  |
| --- | --- |
| >>> x = 4  >>> for i in range(0, x):  print i | Output  0  1  2  3 |

# 8.What is a break statement?

When a break statement is encountered inside a loop, the loop is immediately terminated and the program control resumes at the next statement following the loop.

# Eg:

|  |  |
| --- | --- |
| >>> while True:  line = input('>')  if line == 'done':  break  print (line)  print('Done!') | Output  >>  >  Done!  > |

# 9.What is a continue statement?

The continue statement works somewhat like a break statement. Instead of forcing termination, it forces the next iteration of the loop to take place, skipping any code in between.

# Eg:

|  |  |
| --- | --- |
| for num in range(2,10):  if num%2==0:  print( “Found an even number”, num )  continue  print (“Found a odd number”, num) | output:  Found an even number 2  Found a odd number 3  Found an even number 4  Found a odd number 5  Found an even number 6  Found a odd number 7  Found an even number 8  Found a odd number 9 |

**10.What is the purpose of pass statement?**

Using a pass statement is an explicit way of telling the interpreter to do nothing.

Eg: def bar():

print("Before executing pass")

pass

print("After executing pass")

If the function bar() is called, it produce following output.

>>> bar()

Before executing pass

After executing pass

# 11.Compare return value and composition.

# Return Value:

Return gives back or replies to the caller of the function. The return statement causes our function to exit and hand over back a value to its caller.

|  |  |
| --- | --- |
| Eg:import math  def area(radius):  temp = math.pi \* radius\*\*2  return temp | Output  >>> area(5)  78.53981633974483 |

# Composition:

Calling one function from another is called composition. If F(x) and G(x) are two functions then its composition is given as F(G(x)).

That is passing result of one function as argument to the other function is known as function composition.

# Eg:

>>>def square(a):

‘’’ square() calculate square of a number’’’

return a \* a

>>>def twice\_square(func):

‘’’ twice\_square() implements function composition of square()

return lambda x : func(func(x))

>>>quad = twice\_square(square)

>>> quad(5)

625

# 12. What is recursion?

The process in which a function calls itself directly or indirectly is called recursion and the corresponding function is called as recursive function.

Eg:

def factorial(n):

if n == 1:

return 1

else:

return n \* factorial(n-1)

# Explain global and local scope.

The scope of a variable refers to the places that we can see or access a variable. If we define a variable on the top of the script or module, the variable is called global variable. The variables that are defined inside a class or function is called local variable.

|  |  |
| --- | --- |
| Eg:  def my\_local():  a=10  print(“This is local variable”) | Eg:  a=10  def my\_global():  print(“This is global variable”) |

1. **Compare string and string slices.**

A string is a sequence of character.

Eg: fruit = ‘banana’

# String Slices :

A segment of a string is called string slice, selecting a slice is similar to selecting a character.

**Eg:** >>> s ='Monty Python'

>>> print s[0:5]

Monty

>>> print s[6:12]

Python

# Define string immutability.

Python strings are immutable. ‘a’ is not a string. It is a variable with string value. We can’t mutate the string but can change what value of the variable to a new string.

|  |
| --- |
| Program:  a = "foo"  # a now points to foo  b=a  # b now points to the same foo that a points to  a=a+a  # a points to the new string “foofoo”, but b points to the same old “foo”  print a  print b  Output:  foofoo  foo #It is observed that ‘b’ hasn’t changed even though ‘a’ has changed |

**16.What will be the output of print str[2:5] if str = “hello world!”?**

>>> str = "hello world!"

>>> print(str[2:5])

llo

**17.Differentiate for loop and while loop?**

|  |  |
| --- | --- |
| **For loop** | **While loop** |
| Definite loop | Indefinite Loop |
| A for loop iterates (loops) over the elements of an iterable (ie., list, tuple, string, range()) | A while loop iterates until the condition becomes false |
| >>> for x in [1, 2, 3]:  print(x)  1  2  3 | >>> x = 1  >>> while x < 4:  ... print(x)  ... x += 1  1  2  3 |

**18. How to split strings and what function is used to perform that operation?**

The str.split() method is used to split strings up.

>>>book=’Problem Solving and Python Programming’

>>>print(book.split())

[‘Problem’, ‘Solving’, ‘and’, ‘Python’, ‘Programing’]

# Mention a few string functions.

s.captilize() – Capitalizes first character of string

s.count(sub) – Count number of occurrences of sub in string

s.lower() – converts a string to lower case

s.split() – returns a list of words in string

# What are string methods?

A method is similar to a function—it takes arguments and returns a value—but the syntax is different. For example, the method upper takes a string and returns a new string with all uppercase letters:

Instead of the function syntax upper(word), it uses the method syntax word.upper()

>>> word = 'banana'

>>> new\_word = word.upper()

>>> print new\_word BANANA

# Explain about string module.

The string module contains number of useful constants and classes, as well as some deprecated legacy functions that are also available as methods on strings.

|  |
| --- |
| Eg:**Program:**  import string  text = "Monty Python's Flying Circus"  print ("upper", "=>", string.upper(text) )  print ("lower", "=>", string.lower(text) )  print ("split", "=>", string.split(text))  print ("join", "=>", string.join(string.split(text),"+"))  print ("replace", "=>", string.replace(text, "Python", "Java"))  print ("find", "=>", string.find(text, "Python"), string.find(text, "Java"))  print ("count", "=>", string.count(text, "n"))  Output  upper => MONTY PYTHON'S FLYING CIRCUS  lower => monty python's flying circus  split => ['Monty', "Python's", 'Flying', 'Circus']  join => Monty+Python's+Flying+Circus  replace => Monty Java's Flying Circus  find => 6 -1  count => 3 |

1. **What is Anonymous Function in Python?**

Anonymous function are the function that are not defined like normal function by using the keyword def. Anonymous function are defined by keyword lambda

Syntax: lambda arguments: expression

**Example:**

>>> add = lambda value1,value2 : value1 + value2

>>>

>>> add(5, 9)

14

1. **Define Iteration.**

Iteration is otherwise called as looping. There are situations when programmers need to execute a block of code several number of times. Repeated execution of a set of statements is called iteration or looping.

1. **State the difference between Linear Search and Binary Search.**

Linear Search also known as sequential search is the simplest search algorithm. It searched for a specified value in a list by checking every element in the list

Binary Search is also a method used to locate a specified value in a sorted list. Binary search method involves the number of elements checked thereby reducing the time taken.

1. **Differentiate range() and xrange()?**

|  |  |
| --- | --- |
| **range() Method** | **xrange() method** |
| range()returns a Python list object | xrange() returns an xrange object. |
| range() function produced all numbers instantaneously before the for loop started executing. | xrange doesn’t actually generate a static list at run-time like range does. xrange() creates the values on demand as you need them using a technique called yielding. |
| range() function is it uses a very large amount of memory if you are producing a huge range of numbers. | xrange() function always produces the next number on the fly i.e. only keeps one number in memory at a time to consume less memory. |
| Syntax: range(start, stop, step)  Example: range(1,6) will produce [1,2,3,4,5] | Syntax: xrange(start, stop, step)  Example: xrange(1,6) will produce [1,2,3,4,5] |

**PART – B**

**10 – Marks**

1. Explain If structure in detail. Explain with suitable examples
2. Explain iteration structure in detail with suitable examples
3. i) Define methods in a string module with an example program using atleast 5 methods.

ii) How to access characters of a string?

1. i) Write a python program to find sum of N natural numbers

ii)Explain in detail about break, continue, pass control constructs?

1. Write a python program to find the factorial of a given number with and without recursion.
2. Write a python program to implement linear search and binary search.
3. Write a python program to generate first ‘N’ Fibonacci numbers with and without recursion.
4. Explain recursive function. How do recursive function works. Explain with an example code.
5. What is call by value and call be reference. Explain with suitable examples.
6. i)Discuss the methods to manipulate the arrays in python.

ii) Write about function composition.