CHAPTER 6

CONTROL STRUCTURES

1. Define Control Structure or Control Statement
   ➤ A program statement that causes a jump of control from one part of the program to another is called **control structure** or **control statement**.
   ➤ Control statements are compound statements used to alter the control flow of the process or program depending on the state of the process.

2. What is the use of Control Structure?
   ➤ Programs consist of statements which are executed in sequence, to alter the flow we use control statements.

3. What are the types of Control Structure?
   ➤ There are **3** types of Control Structures or Statements. They are
     ➤ **Sequential** – Statements are executed one after another.
     ➤ **Alternative or Branching** – Skip a statement or set of statements and execute another segment based on condition.
     ➤ **Iteration or Looping** – Executes a set of statement for multiple times.

4. Define Sequential Statement
   ➤ A **Sequential statement** is composed of a sequence of statements which are executed one after another.
   ➤ A code to print your name, address and phone number is an example of sequential statement.
   ➤ Eg:
     
     ```
     print("Hello! This is Shyam")
     print("22, Alagappan Street, Kumbakonam, 9677066334")
     ```

5. What is Branching Statement or Alternative Statement?
   ➤ In our day-to-day life we need to take various decisions and choose an alternate path to achieve our goal.
   ➤ May be we would have taken an alternate route to reach our destination when we find the usual road by which we travel is blocked.
   ➤ The Statements are executed based on the condition.
This type of decision making is what we are to learn through **Alternative or Branching Statement**.

6. **What are the types of Alternative or Branching Statements?**
   - There are 3 types of Alternative or Branching Statements namely,
   - **Simple if** – Simplest of all decision making statements.
   - **If – Else** – To check the true block as well as the false block.
   - **If…elif Statement** – Construct a chain of if statement

7. **Define Simple If Statement and Write the Syntax?**
   - **Simple if** is the simplest of all decision making statements. Condition should be in the form of relational or logical expression.
   - **Syntax:**
     ```python
     if <condition>:
     statements-block1
     ```
   - **Eg:**
     ```python
     x=int(input("Enter your age :"))
     if x>=18:
     print ("You are eligible for voting")
     ```

8. **Define If – Else Statement and Write the Syntax?**
   - **If else** statement provides control to check the true block as well as the false block.
   - **If else** statement provides two possibilities and the condition determines which block is to be executed.
   - **Syntax:**
     ```python
     if <condition>:
     statements-block 1
     else:
     statements-block 2
     ```
   - **Eg:**
     ```python
     a = int(input("Enter any number :"))
     if a%2==0:
     print (a, " is an even number")
     else:
     ```
print (a, " is an odd number")

9. What is the alternate method to write the syntax for if-else?
 Syntax:
variable = variable1 if condition else variable 2
 Eg:
a = int (input("Enter any number :"))
x="even" if a%2==0 else "odd"
print (a, " is ",x)
 Output:
Enter any number : 3
3 is odd

10. Define Nested If..elif…else Statement
 When we need to construct a chain of if statement(s) then ‘elif’ clause can be used instead of ‘else’.
 Syntax:
if <condition-1>:
    statements-block 1
elif <condition-2>:
    statements-block 2
else:
    statements-block n
 In the syntax of if..elif..else mentioned above, condition-1 is tested if it is true then statements-block1 is executed, otherwise the control checks condition-2, if it is true statements-block2 is executed and even if it fails statements-block n mentioned in else part is executed.

11. Write the Syntax and Draw the Flowchart for Nested If..elif…else Statement?
 SYNTAX:
if <condition-1>:
    statements-block 1
elif <condition-2>:
    statements-block 2
else:
12. What is Iteration Statements or Looping Statements?

Iteration or loop are used in situation when the user need to execute a block of code several of times or till the condition is satisfied.

A loop statement allows executing a statement or group of statements multiple times.

There are 2 types of Looping namely,

- While Loop, &
- For Loop.

13. Define While Loop with Syntax?

Syntax:

```python
while <condition>:
    statements block 1
```

[else:
14. Define For Loop with Syntax?

- **For** loop is the most comfortable loop.
- It is also an entry check loop.
- The condition is checked in the beginning and the body of the loop (statements-block 1) is executed if it is only True otherwise the loop is not executed.

**Syntax:**

```python
for counter_variable in sequence:
    statements-block 1

[else: # optional block
    statements-block 2]
```

- In Python, for loop uses the `range( )` function in the sequence to specify the initial, final, and increment values.
- `range( )` generates a list of values starting from start till stop one.

15. Write the Syntax for `range( )` with example?

**Syntax:**

```python
range (start,stop,[step])
```

Where,

- `start` – refers to the initial value
- `stop` – refers to the final value
- `step` – refers to increment value, this is optional part.

**Eg:**

- `range (1,30,1)` – will start the range of values from 1 and end at 29
- `range (2,30,2)` – will start the range of values from 2 and end at 28
16. What is the use of Indentation in Python compare to other languages?

- In Python, indentation is important in loop and other control statements.
- Indentation only creates blocks and sub-blocks like how we create blocks within a set of { } in languages like C, C++ etc.
- In most other programming languages, indentation is used only to help make the code look pretty.
- But in Python, it is required to indicate to which block of code the statement belongs to.

17. Define Nested Loop

- A loop placed within another loop is called as nested loop structure.
- One can place a while within another while; for within another for; for within while and while within for to construct such nested loops.

18. What is Jump Statement? What are the Jump Statements in Python?

- The jump statement in Python is used to unconditionally transfer the control from one part of the program to another.
- There are three keywords to achieve jump statements in Python:
  - Break,
  - Continue, &
  - Pass.

19. Define Break Statement

- The break statement terminates the loop containing it.
- Control of the program flows to the statement immediately after the body of the loop.
- A while or for loop will iterate till the condition is tested false.
- One can even transfer the control out of the loop (terminate) with help of break statement.
- When the break statement is executed, the control flow of the program comes out of the loop and starts executing the segment of code after the loop structure.
- If break statement is inside a nested loop (loop inside another loop), break will terminate the innermost loop.
- Syntax:
  
  break
20. **What is Continue Statement? Draw the Flowchart?**

- Continue statement unlike the break statement is used to skip the remaining part of a loop and start with next iteration.

  **Syntax:**
  
  ```
  continue
  ```

21. **Define Pass Statement**

- **Pass** statement in Python programming is a null statement.
- Pass statement when executed by the interpreter it is completely ignored.
- Nothing happens when pass is executed, it results in no operation.
- Pass statement can be used in ‘if’ clause as well as within loop construct, when you do not want any statements or commands within that block to be executed.

  **Syntax:**
  
  ```
  pass
  ```
CHAPTER – 7
PYTHON FUNCTIONS

1. What are Functions?
   - Functions are named blocks of code that are designed to do specific job.
   - When you want to perform a particular task that you have defined in a function, you call the name of the function responsible for it.

2. What are the different types of Functions?
   - User-defined functions – Defined by the Users
   - Built-in functions – In built within Python
   - Lambda functions – Anonymous Un – Named Function
   - Recursion functions – Functions that calls itself

3. What are the Advantages of Function?
   - It avoids repetition and makes high degree of code reusing.
   - It provides better modularity for your application.

4. Write the Syntax for User – Defined Function?
   - Syntax:
     
     ```python
     def <function_name ([parameter1, parameter2…])> :
     <Block of Statements>
     return <expression / None>
     ```
   - Eg:
     ```python
     def hello():
     print ("hello - Python")
     return
     hello()
     ```

5. What are the things that need to be noted when defining Functions?
   - Function blocks begin with the keyword “def” followed by function name and parenthesis ().
   - Any input parameters or arguments should be placed within these parentheses when you define a function.
   - The code block always comes after a colon (:) and is indented.
6. **What is the Use of Return Statement?**
   - The statement “return [expression]” exits a function, optionally passing back an expression to the caller.
   - A “return” with no arguments is the same as return None.

7. **Define Block**
   - A block is one or more lines of code, grouped together so that they are treated as one big sequence of statements while execution.
   - Statements in a block are written with indentation. Usually, a block begins when a line is indented (by four spaces) and all the statements of the block should be at same indent level.

8. **What is meant by Nested Block?**
   - A block within a block is called nested block.
   - When the first block statement is indented by a single tab space, the second block of statement is indented by double tab spaces.

9. **What are the Advantages of User – Defined Function?**
   - Functions help us to divide a program into modules. This makes the code easier to manage.
   - It implements code reuse. Every time you need to execute a sequence of statements, all you need to do is to call the function.
   - Functions, allows us to change functionality easily, and different programmers can work on different functions.

10. **How to Pass Parameters in Function?**
    - The parameters that you place in the parenthesis will be used by the function itself.
    - You can pass all sorts of data to the functions.
    - **Syntax:**
        
        ```python
        def funname(parameters separated by comma):
            assume w = 3 and h = 5
            def area(w,h):
                return w * h
            print (area (3,5))
        ```
    - The value of 3 and 5 are passed to w and h respectively, the function will return 15 as output.
11. Define Parameters and Arguments
   - Parameters are the variables used in the function definition.
   - Arguments are the values we pass to the function parameters

12. What are the types of Function Arguments?
   - Arguments are used to call a function and there are primarily 4 types of functions that one can use:
     - Required arguments,
     - Keyword arguments,
     - Default arguments and
     - Variable-length arguments.

13. What is Required Arguments?
   - “Required Arguments” are the arguments passed to a function in correct positional order.
   - The number of arguments in the function call should match exactly with the function definition.
   - At least one parameter to prevent syntax errors to get the required output.
   - Eg:
     ```python
     def printstring(str):
         print("Required arguments ")
         print(str)
         return
     printstring("Welcome")
     ```
   - Output:
     - Required arguments
     - Welcome

14. What is Keyword Arguments?
   - Keyword arguments will invoke the function after the parameters are recognized by their parameter names.
   - The value of the keyword argument is matched with the parameter name and so, one can also put arguments in improper order (not in order).
   - Eg:
     ```python
     def printdata (name):
         print("Keyword arguments")
         print("Welcome")
     ```
   - Output:
     - Keyword arguments
print ("Name:", name)
return
printdata(name = "Gshan")

**Output:**

Keyword arguments
Name: Gshan

15. **What is Default Arguments?**

- Default argument is an argument that takes a default value if no value is provided in the function call.
- The following example uses default arguments, that prints default salary when no argument is passed.
- **Eg:**

```python
def printinfo(name, salary = 3500):
    print ("Name: ", name)
    print ("Salary: ", salary)
    return
printinfo("Mani")
```

**Output:**

Name: Mani
Salary: 3500

16. **What is Variable Length Arguments?**

- In some situations you might need to pass more arguments than have already been specified. Going back to the function to redefine it can be a tedious process.
- Variable-Length arguments can be used instead.
- These are not specified in the function’s definition and an asterisk (*) is used to define such arguments.
- **Eg:**

```python
def printnos (*nos):
    for n in nos:
        print(n)
    return
print ('Printing two values')
```
17. Write the Syntax for Variable Length Arguments?

- **Syntax:**
  ```python
  def function_name(*args):
      function_body
      return_statement
  ```

- **Eg:**
  ```python
  def printnos (*nos):
      for n in nos:
          print(n)
      return
  print ('Printing three values')
  printnos (10,20,30)
  ```

- **Output:**
  Printing two values
  10
  20
  30

18. What are the methods to pass the arguments in Variable Length Arguments?

- There are 2 methods to pass the arguments in Variable Length Arguments.
  - Non keyword variable arguments
  - Keyword variable arguments

- Non-keyword variable arguments are called **tuples**.

19. What is Anonymous Function or Lambda Function?

- Anonymous function is a function that is defined **without a name**.
- Normal functions are defined using the **def** keyword, Anonymous functions are defined using the **lambda** keyword.
Anonymous functions are also called as **lambda** functions.

20. **What is the use of Lambda or Anonymous Function?**
   - Lambda function is mostly used for creating small and one-time anonymous function.
   - Lambda functions are mainly used in combination with the functions like filter(), map() and reduce().

21. **What are the uses of Lambda Function?**
   - Lambda function can take any number of arguments and must return one value in the form of an expression.
   - Lambda function can only access global variables and variables in its parameter list.

22. **Write the Syntax with Example for Anonymous Function?**
   - **Syntax:**
     ```python
     lambda [argument(s)] :expression
     ```
   - **Eg:**
     ```python
     sum = lambda arg1, arg2: arg1 + arg2
     print ('The Sum is :', sum(30,40))
     ```
   - **Output:**
     The Sum is : 70

23. **Define Return Statement**
   - The return statement causes your function to exit and returns a value to its caller. The point of functions in general is to take inputs and return something.
   - The return statement is used when a function is ready to return a value to its caller.
   - So, only one return statement is executed at run time even though the function contains multiple return statements.
   - Any numbers of ‘return’ statements are allowed in a function definition but only one of them is executed at run time.

24. **Write the Syntax for Return Statement?**
   - **Syntax:**
     ```python
     return [expression list ]
     ```
This statement can contain expression which gets evaluated and the value is returned. If there is no expression in the statement or the return statement itself is not present inside a function, then the function will return the none object.

**Eg:**
```python
def usr_abs (n):
    if n>=0:
        return n
    else:
        return –n
x=int (input("Enter a number :"))
print (usr_abs (x))
```

**Output:**
Enter a Number : 25
25

25. Define Scope of a Variable and its types?
- Scope of variable refers to the part of the program, where it is accessible, i.e., area where you can refer (use) it.
- We can say that scope holds the current set of variables and their values.
- There are two types of scopes - **local scope** and **global scope**.

26. Define Local Scope
- A variable declared inside the function's body or in the local scope is known as local variable.
- When a variable is created inside the function/block, the variable becomes local to it.
- A local variable only exists while the function is executing.
- The formal arguments are also local to function.

**Eg:**
```python
def loc():
    y=0 # local scope
    print(y)
loc()
```

**Output:**
0
27. Define Global Scope
   - A variable, with global scope can be used anywhere in the program. It can be created by defining a variable outside the scope of any function/block.
   - When we define a variable outside a function, it’s global by default.
   - You don’t have to use global keyword.
   - **Eg:**
     ```python
c = 1
def add():
    print(c)
add()
```
   - **Output:**
     
     1

28. What are the Rules for the Local Variable?
   - A variable with local scope can be accessed only within the function/block that it is created in.
   - When a variable is created inside the function/block, the variable becomes local to it.
   - A local variable only exists while the function is executing.
   - The formal arguments are also local to function.

29. What are the Rules for Global Keyword?
   - When we define a variable outside a function, it’s global by default. You don’t have to use global keyword.
   - We use global keyword to read and write a global variable inside a function.
   - Use of global keyword outside a function has no effect

30. Define abs() function with syntax and example
   - Returns an absolute value of a number. The argument may be an integer or a floating point number.
   - **Syntax:**
     ```python
     abs(x)
     ```
   - **Eg:**
     ```python
     x= -23.2
     print(‘x=’,abs(x))
     ```
   - **Output:**
31. Define ord() function with syntax and example
   - It returns the ASCII value for the given Unicode Character. This function is inverse of chr() function.
   - Syntax:
     ```python
     ord(c)
     ```
   - Eg:
     ```python
c='a'
print('c=', ord(c))
```
   - Output:
     ```
c=97
``` 
32. Define chr() function with syntax and example
   - Returns the Unicode character for the given ASCII value. This function is inverse of ord() function.
   - Syntax:
     ```python
     chr(i)
     ```
   - Eg:
     ```python
c=65
print(chr(c))
```
   - Output:
     ```
A
``` 
33. Define bin() function with syntax and example
   - Returns a binary string prefixed with “0b” for the given integer number.
   - Syntax:
     ```python
     bin(i)
     ```
   - Eg:
     ```python
i=15
print(bin(i))
```
   - Output:
     ```
0b1111
```
34. Define type() function with syntax and example
   - Returns the type of object for the given single object. **Note:** This function used with single object parameter.
   - **Syntax:**
     ```python
     type(object)
     ```
   - **Eg:**
     ```python
     x=15.2
     print(type(x))
     ```
   - **Output:**
     ```python
     <class 'float'>
     ```

35. Define id() function with syntax and example
   - **id( )** Return the “identity” of an object. i.e. the address of the object in memory. The address of x and y may differ in your system.
   - **Syntax:**
     ```python
     id(object)
     ```
   - **Eg:**
     ```python
     x='a'
     print(id(x))
     ```
   - **Output:**
     13480736

36. Define min() function with syntax and example
   - It is used to return the minimum value in a list.
   - **Syntax:**
     ```python
     min(list)
     ```
   - **Eg:**
     ```python
     A=[1,2,3,4]
     print('Minimum is:',min(A))
     ```
   - **Output:**
     Minimum is:1

37. Define max() function with syntax and example
   - It is used to return the maximum value in a list.
38. Define sum() function with syntax and example
   - It is used to return the sum of the values
   - Syntax:
     ```python
     sum(list)
     ```
   - Eg:
     ```python
     A=[1,2,3,4]
     print('Sum is:',sum(A))
     ```
   - Output:
     Sum is:10

39. Define round() function with syntax and example
   - It is used to return the nearest integer to its input.
   - The first argument is used to specify the value to be rounded.
   - The second argument is used to specify the number of decimal digits desired after rounding.
   - Syntax:
     ```python
     round(number,[digits])
     ```
   - Eg:
     ```python
     x=15.89
     print(round(x,0))
     print(round(x,1))
     ```
   - Output:
     15.0
     15.9

40. Define pow() function with syntax and example
   - It is used to return the computation of ab i.e. (a**b). a is raised to the power of b.
41. Define floor() function with syntax and example
   - It is used to return the largest integer less than or equal to x.
   - **Syntax:**
     ```python
     math.floor(x)
     ```
   - **Eg:**
     ```python
     x=15.7
     print(math.floor(x))
     ```
   - **Output:**
     15

42. Define ceil() function with syntax and example
   - It is used to return the smallest integer greater than or equal to x.
   - **Syntax:**
     ```python
     math.ceil(x)
     ```
   - **Eg:**
     ```python
     x=15.7
     print(math.ceil(x))
     ```
   - **Output:**
     16

43. Define sqrt() function with syntax and example
   - It is used to return the square root of x. x should always be greater than 0.
   - **Syntax:**
     ```python
     math.sqrt(x)
     ```
   - **Eg:**
44. What is Composition in Functions?
   - The value returned by a function may be used as an argument for another function in a nested manner. This is called **composition**.
   - If we wish to take a numeric value or an expression as a input from the user, we take the input string from the user using the function **input()** and apply **eval()** function to evaluate its value.

45. What is Infinite Iteration?
   - A recursive function calls itself.
   - Imagine a process would iterate indefinitely if not stopped by some condition.
   - Such a process is known as **Infinite Iteration**.

46. What is Recursive Function?
   - When a function calls itself is known as recursion.
   - Recursion works like loop but sometimes it makes more sense to use recursion than loop. You can convert any loop to recursion.

47. How the Recursive Function Works?
   - Recursive function is called by some external code.
   - If the base condition is met then the program gives meaningful output and exits.
   - Otherwise, function does some required processing and then calls itself to continue recursion.

48. What is the base condition in Recursive Function?
   - The condition that is applied in any recursive function is known as base condition.
   - A base condition is must in every recursive function otherwise it will continue to execute like an infinite loop.

49. Write a Python Program to calculate the factorial using Recursive Function?
   - **Coding:**
def fact(n):
    if n == 0:
        return 1
    else:
        return n * fact(n-1)

print (fact (0))
print (fact (5))

Output:
1
120