LAB # 02

FUNCTION , LOOP AND CONDITIONAL STATEMENTS

OBJECTIVE

Familiarization with Python language using function, loop and conditional statement.

THEORY

Functions:

Basically, we can divide functions into the following two types:

1. Built-in functions - Functions that are built into Python.
2. User-defined functions - Functions defined by the users themselves.

A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing.
As you already know, Python gives you many built-in functions like print(), etc. but you can also create your own functions. These functions are called user-defined functions.

Defining a Function:

You can define functions to provide the required functionality. Here are simple rules to define a function in Python.

- Function blocks begin with the keyword **def** followed by the function name and parentheses (( )).
- Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
- The code block within every function starts with a colon (:) and is indented.

Syntax:

```
def functioname ( PARAMETERS ):
    Statement
```

Example:

```
def greet_user(username):
    """Display a simple greeting."""
    print("Hello, " + username.title() + "!")
greet_user('jesse')
```

Arguments:
An argument is a piece of information that is passed from a function call to a function. When we call the function, we place the value we want the function to work with in parentheses.

You can call a function by using the following types of formal arguments:

- Required arguments
- Keyword arguments
- Default arguments

Required Arguments:
Required arguments are the arguments passed to a function in correct positional order. Here, the number of arguments in the function call should match exactly with the function definition.

```
def square(x):
    y = x * x
    a=y+1
    return y

toSquare = 10
result = square(toSquare)
print ("The result of"  str(toSquare) + " squared is " + str(result))
```

Keyword Arguments:
Keyword arguments are related to the function calls. When you use keyword arguments in a function call, the caller identifies the arguments by the parameter name.
This allows you to skip arguments or place them out of order because the Python interpreter is able to use the keywords provided to match the values with parameters.

```
def printinfo( name, age ):
   "This prints a passed info into this function"
   print ("Name: ", name)
   print ("Age ", age)
   return;
# Now you can call printinfo function
printinfo(name="miki", age=50,  )
```

When the above code is executed, it produces the following result –

```
Name:  miki
Age  50
```

Default Arguments:

A default argument is an argument that assumes a default value if a value is not provided in the function call for that argument. The following example gives an idea on default arguments, it prints default age if it is not passed –

```
def printinfo( name, age = 35 ):
   "This prints a passed info into this function"
   print ("Name: ", name)
   print ("Age ", age)
   return;

# Now you can call printinfo function
printinfo(name="miki", age=50 )
printinfo( name="miki" )
```

When the above code is executed, it produces the following result –

```
Name:  miki
Age  50
Name:  miki
Age  35
```

For loop Statement:

Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable. It has the ability to iterate over the items of any sequence, such as a list or a string.

Syntax:

for iterating variable in sequence: Statements(s)

Example:

```
fruits = ['banana', 'apple',  'mango']
for fruit in fruits:
   print ('Current fruit :', fruit)
```

Output:

```
Current fruit : banana
Current fruit : apple
Current fruit : mango
>>>
```

While Loop:

The for loop takes a collection of items and executes a block of code once for each item in the collection. In contrast, while loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.

Syntax:

The syntax of a while loop in Python programming language is −

```
while expression:
statement(s)
```

Example:

```
a = 0
while a < 10:
    a = a + 1
    print ("Result",a)
```

Loop Control Statements:

Loop control statements change execution from its normal sequence. You might face a situation in which you need to exit a loop completely when an external condition is triggered or there may also be a situation when you want to skip a part of the loop and start next execution.

Python provides break and continue statements to handle such situations and to have good control on your loop.

The break Statement:

The break statement in Python terminates the current loop and resumes execution at the next statement, just like the traditional break found in C.

The most common use for break is when some external condition is triggered requiring a hasty exit from a loop. The break statement can be used in both while and for loops.

Example

```
number = 0

for number in range(10):
   number = number + 1

   if number == 5:
      break    # break here

   print('Number is ' + str(number))

print('Out of loop')
```

The continue Statement:

The continue statement in Python returns the control to the beginning of the while loop. The continue statement rejects all the remaining statements in the current iteration of the loop and moves the control back to the top of the loop.

The continue statement can be used in both while and for loops.

```
number = 0

for number in range(10):
   number = number + 1

   if number == 5:
      continue    # continue here

   print('Number is ' + str(number))

print('Out of loop')
```

The pass Statement:

The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute.

The pass statement is a null operation; nothing happens when it executes.

Example 1:

```
for letter in 'Python':
    if letter == 'h':
        pass
    print ('This is pass block')
    print ('Current Letter :', letter)
```

Example 2:

```
number = 0

for number in range(10):
   number = number + 1

   if number == 5:
      pass    # pass here

   print('Number is ' + str(number))

print('Out of loop')
```

CONDITIONAL STATEMENTS:

***If-Else* Statement:**

An *if-else* block is similar to a simple if statement, but the else statement allows you to define an action or set of actions that are executed when the conditional test fails.

Example:

```
cars = ['audi', 'bmw', 'subaru', 'toyota']

for car in cars:
    if car == 'bmw':
        print(car.upper())
    else:
        print(car.title())
```

The *if-elif-else* Statement:

The elif statement allows you to check multiple expressions for TRUE and execute a block of code as soon as one of the conditions evaluates to TRUE.

Syntax:

```
if expression1:
  statement(s)
elif expression2:
  statement(s)
elif expression3:
  statement(s)
else:
  statement(s)
```

Example: (Simple if statements)

```
age = 19
if age >=18:
    print("You are old enough to vote!")
```

Example: (if/else statements)

```
age = 17
if age >=18:
    print("You are old enough to vote!")
    print("Have you registered to vote yet?")
else:
    print("Sorry,you are too young to vote.")
    print("Please register to vote as soon as you turn 18!")
```

Example (*if-elif-else statements*)

For example, consider an amusement park that charges different rates for different age groups:

- Admission for anyone under age 4 is free.
- Admission for anyone between the ages of 4 and 18 is $5.
- Admission for anyone age 18 or older is $10.

```
age = 12
if age < 4:
    print("Your admission cost is $0.")
elif age < 18:
    print("Your admission cost is $5.")
else:
    print("Your admission cost is $10.")
```

You can use as many elif blocks in your code as you like.

Lab Tasks:

1. Write a python script that take a user input and to create the multiplication table (from 1 to 10) of that number.

2. Write a function called describe_city( ) that accepts the name of a city and its country. The function should print a simple sentence, such as **Islamabad is in Pakistan**. Give the parameter for the country a default value. Call your function for three different cities, at least one of which is not in the default country.

3. Write a function called absolute_num() that accepts one parameter, num. The function should return only positive value, and apply condition on it. This function returns the absolute value of the entered number.

4. Write Python Program to check whether an alphabet is a vowel or consonant? (use if, else conditional statement).

5. Write a Python program to check whether a number is prime or not? (use if, else conditional statement).

6. Write a Python program that prints all the numbers from 0 to 6 except 3 and 6. (Hint: Use 'continue' statement).

7. Write a Python program to construct the following pattern. (using nested loop)

   1
   2 2
   3 3 3
   4 4 4 4
   5 5 5 5 5

8. Stages of Life: Write an if-elif-else chain that determines a person's stage of life. Set a value for the variable age, and then:
   - If the person is less than 2 years old, print a message that the person is a baby.
   - If the person is at least 4 years old but less than 13, print a message that the person is a kid.

- If the person is at least 13 years old but less than 20, print a message that the person is a teenager.
- If the person is at least 20 years old but less than 65, print a message that the person is an adult.
- If the person is age 65 or older, print a message that the person is an elder.