University Of Engineering and Technology, Lahore
Computer Engineering Department

Course Name: Fundamentals of Programming & Data Science	**Course Code:** CMPE-112L
Assignment Type: Lab	**Dated:** 17th January 2024
Semester: 2nd	**Session:** 2023
Lab/Project/Assignment #: 1	**CLOs to be covered: CLO 1**
Lab Title: I/O, Data Types and Conditional Statements in Python	**Teacher Name:** Engr. Afeef Obaid

Lab Evaluation:

CLO 1	Apply appropriate programming techniques to create executable programs to solve well defined problems					
Levels (Marks)	**Level 1**	**Level 2**	**Level 3**	**Level 4**	**Level 5**	**Level 6**
(10)						
					Total	**/10**

Rubrics for Current Lab Evaluation

Scale	**Marks**	**Level**	**Rubric**
Excellent	**10**	L1	Submitted all lab tasks, BONUS task, have good understanding.
Very Good	**8**	L2	Submitted the lab tasks but have good understanding
Good	**6**	L3	Submitted the lab tasks but have weak understanding.
Basic	**4**	L4	Submitted the lab tasks but have no understanding.
Barely Acceptable	**2**	L5	Submitted only one lab task.
Not Acceptable	**0**	L6	Did not attempt

LAB No 1

Lab Goals/Objectives:

By reading this manual, students will be able to:

- Learn about Input and Output commands of Python
- Know about the data types in Python
- Do decision control in Python

Equipment Required: Computer system with Pycharm IDE and python package installed on it

PYTHON print() Function

The print() function prints the specified message to the screen, or other standard output device. The message can be a string, or any other object, the object will be converted into a string before written to the screen.

print(object(s)*,* sep=separator*,* end=end*)*

object(s)	Any object, and as many as you like. Will be converted to string before printed
sep='separator'	Optional. Specify how to separate the objects, if there is more than one. Default is ' '
end='end'	Optional. Specify what to print at the end. Default is '\n' (line feed)

Examples

```
print("UET Lahore")

print('UET Lahore')

print("UET Lahore",end="-")

print("Computer Engineering Dept")

print("UET Lahore","Computer Engineering Dept")

print("UET Lahore","Computer Engineering Dept",sep="--")
```

Escape Sequences

Escape sequences in Python are special characters that are used to represent certain characters that cannot be easily typed or displayed. They are represented by a backslash (\) followed by a letter or combination of letters that represent the desired special character. Here are some commonly used escape sequences in Python:

1. **`\n`**: This represents a new line character. When used in a string, it will start a new line of text.
2. **`\t`**: This represents a tab character. When used in a string, it will create a horizontal tab.
3. **`\\`**: This represents a backslash character. When used in a string, it will display a single backslash.
4. **`\'`**: This represents a single quote character. When used in a string, it will display a single quote.
5. **`\"`**: This represents a double quote character. When used in a string, it will display a double quote.

Examples

```
print("Hello\nworld")

print("First\tSecond\tThird")

print("This is a backslash: \\")

print("She said, \"Hello!\"")

print('He\'s happy')
```

Format Specifier

Format specifiers in Python are used to format and display values in a certain way when printing output to the console or to a file. They are used within curly braces {} in a string and start with a colon : followed by a specifier character that represents the type of formatting desired. Here are some commonly used format specifiers in Python:

1. **%s:** This is used for formatting strings.
2. **%d:** This is used for formatting integers.
3. **%f:** This is used for formatting floating point numbers.
4. **%e or %E:** This is used for formatting numbers in scientific notation.
5. **%x or %X:** This is used for formatting numbers in hexadecimal notation.
6. **%o:** This is used for formatting numbers in octal notation.

Examples

```
Subject = "Calculus"

Grade = "B"

Percent = 75.1275

print("In Subject %s I got %s grade." % (Subject, Grade))

print("My Percentage is %f ." % Percent)

print("The hexadecimal value of 15 is %x." % 15)

print("The octal value of 10 is %o." % 10)
```

Python also provides another way to format strings using format() method.

Example

```
Subject="Calculus"

Grade="B"

Percent = 75.1275

print("In Subject {} i got {} grade and {:.2f}

percentage.".format(Subject,Grade,Percent))
```

In the above example, {} serves as a placeholder for the values passed to the format() method. The :.2f specifier indicates that the value of percentage should be displayed as a floating point number with 2 decimal places.

PYTHON input() Function

The input() function is a built-in function in Python that allows you to accept user input from the keyboard. When you call the input() function, the program will wait for the user to enter some text followed by the Enter key, and then it will return that text as a string

Example

```
name = input("What is your name? ")

print("Hello, {} !".format(name))
```

It's important to note that the input() function always returns a string, even if the user enters a number. If you need to convert the user's input to a different data type, you can use the appropriate conversion function (e.g., int(), float(), etc.).

Data Types

Data types are the classification or categorization of data items. It represents the kind of value that tells what operations can be performed on a particular data

Python has five standard data types:

- Numeric
- String
- List
- Tuple
- Sets
- Dictionary

Boolean (True/False) is also considered as data type in programming languages.

Numeric Data Type

In Python, there are two main types of numeric data types: integers (int) and floating-point numbers (float).

1. **Integers**: Integers are whole numbers with no decimal point. In Python, the int data type represents integers. For example:

   x = 10 *# x is an integer*

2. **Floating-Point Numbers:** Floating-point numbers are decimal numbers. In Python, the float data type represents floating-point numbers. For example:

   y = 3.14 *# y is a floating-point number*

Python also supports some additional numeric data types like:

3. **Complex Numbers:** Complex numbers are numbers with a real and imaginary part. In Python, the complex data type represents complex numbers. For example:

   z = 3 + 4j # z is a complex number with real part 3 and imaginary part 4

4. **Booleans:** Booleans are a special type of numeric data type that can only take two values, True or False. In Python, the bool data type represents booleans. For example:

   a = True *# a is a boolean with value True*

   b = False *# b is a boolean with value False*

String Data Type

- In Python, Strings are arrays of bytes representing Unicode characters. A string is a collection of one or more characters put in a single quote, double-quote or triple quote. In python there is no character data type, a character is a string of length one. It is represented by str class.

- Strings in Python can be created using single quotes or double quotes or even triple quotes.

  Uni='uet Lahore'

0	1	2	3	4	5	6	7	8	9
u	e	t		L	a	h	o	r	e
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

  print(Uni[0])

  print(Uni[3])

  print(Uni[-10])

  print(Uni[78])

- **Length of String**

  print("uet Lahore is a",len(Uni),"Letter string")

- **String Slicing**

print(Uni[0:6])	print(Uni[::3])
print(Uni[0:10])	print(Uni[::344])
print(Uni[0:78])	print(Uni[0:9:2])
print(Uni[0:])	print(Uni[0:9:3])
print(Uni[:7])	print(Uni[::])

- **String Methods**

Python has a set of built-in methods that you can use on strings.
Note: All string methods returns new values. They do not change the original string.

- **upper()** Converts a string into upper case

  print(Uni.upper())

- **lower()** Converts a string into lower case

  print(Uni.lower())

- **replace()** Returns a string where a specified value is replaced with a specified value print(Uni.replace(“uet”, “UCP”))

- **capitalize()** Converts the first character to upper case

  print(Uni.capitalize())

- **center()** Returns a centered string

  print(Uni.center(90))

- **count()** Returns the number of times a specified value occurs in a string

  print(Uni.count(‘e’))

- **endswith()** Returns true if the string ends with the specified value

  print(Uni.endswith(‘ore’))

  print(Uni.endswith(‘ore’,2,5))

- **find()** Searches the string for a specified value and returns the position of where it was found

  ```
  print(Uni.find(“aho”))
  print(Uni.find(“aho”, 3, 8))
  ```

- **isalnum()** Returns True if all characters in the string are alphanumeric

  ```
  print(Uni.isalnum())
  ```

- **isalpha()** Returns True if all characters in the string are alphabets

  ```
  print(Uni.isalpha())
  ```

- **swapcase()** Swaps cases, lower case becomes upper case and vice versa

  ```
  print(Uni.swapcase())
  ```

- **String Formatting**

  String formatting in Python allows you to embed values into a string by replacing placeholders with values. This is useful when you want to create dynamic strings that change based on the values of variables or other factors.

  There are several ways to format strings in Python, including:

1. **Using the % operator:**

   ```
   Subject = "Calculus"
   Grade = "B"
   Percent = 75.1275
   print("In Subject %s I got %s grade." % (Subject, Grade))
   ```

2. **Using the str.format() method:**

   ```
   print("In Subject {} I got {} grade.".format(Subject, Grade))
   print("In Subject {1} I got {0} grade.".format(Grade,Subject))
   ```

3. **Using f-strings (formatted string literals):**

   ```
   print(f"In Subject {Subject} I got {Grade} grade.")
   ```

Conditional Statements

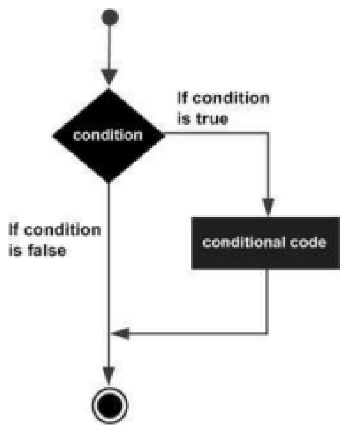
Decision making is anticipation of conditions occurring while execution of the program and specifying actions taken according to the conditions. Decision structures evaluate multiple expressions which produce TRUE or FALSE as outcome.

You need to determine which action to take and which statements to execute if outcome is TRUE or FALSE otherwise.

if Statement

The if statement is used to execute one or more statement depending on whether a condition is True or not. The syntax or correct format of the if statement is given as:

```
if condition:
        statements
```



First, the condition is tested. If the condition is True, then the statements given after colon (:) are executed. We can write one or more statements after colon (:). If the condition is False, then the statements mentioned after colon are not executed

<u>Example</u>

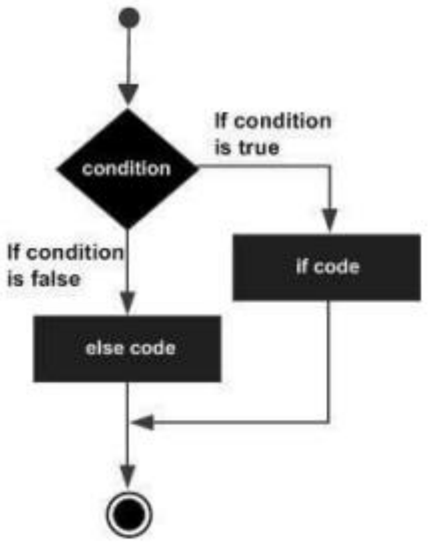
```
x=int(input("Enter your marks "))
if(x>33):
   print("Pass")
```

if else Statement

The if else statement executes a group of statements when a condition is True otherwise, it will execute another group of statements.
The syntax of if else statement is given as:

```
if condition:
        statement1
else:
        statement2
```

If the condition is True, then it will execute statement1.
If the condition is False, then it will execute statement2.



Example

```
x=int(input("Enter your obtained marks in ECAT "))
if(x>=132):
   print("You can apply for Engineering Programs")
else:
   print("You can apply for non-Engineering Programs")
print("Best of Luck")
```

elif 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.

Similar to the else, the elif statement is optional. However, unlike else, for which there can be at most one statement, there can be an arbitrary number of elif statements following an if.
The syntax of elif statement is given as:

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

Core Python does not provide switch or case statements as in other languages, but we can use if..elif...statements to simulate switch case

Example

```
username = input("Enter your username: ")
if (username =="admin" or username =="Admin"):
   print("Welcome, administrator.")
elif (username == "guest" or username == "Guest"):
   print("Welcome, guest.")
else:
   print("Access denied.")
```

Nested if Statement

Nested if conditions are simply if conditions that are placed inside another if or else statement. They allow us to create more complex branching logic in our programs by executing a set of conditions if a previous condition is met.

In Python, nested if conditions are created by indenting one or more if statements within another if or else statement. When the initial if statement is evaluated as true, the program will enter the nested if statement and execute its conditions. If the initial if statement is evaluated as false, the nested if statement will be skipped.

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

Example

```
sentence = input("Enter a sentence: ")
if "Python" in sentence:
   if sentence.startswith("Python"):
      print("The sentence starts with 'Python'")
   elif sentence.endswith("Python"):
      print("The sentence ends with 'Python'")
   else:
      print("The sentence contains 'Python'")
else:
   print("The sentence does not contain 'Python'")
```

Lab Tasks

- Write a program that takes bytes from the user and convert it into Mega Bytes and Giga Bytes
- Write a program that takes your ECAT Marks, Intermediate Part 1 Marks and Matric Marks from user and calculate the aggregate according to given formula.

  ECAT 33% Intermediate 50% Matric 17%
- Write a program to check whether the given number is palindrome or not.
- Write a program that takes two points from the user in the form of $(x_1, y_1) and\ (x_2, y_2)$ and tell their quadrants and distance between them by using distance formula as given below

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

- Write a program that determine the nature of the roots of the Quadratic Equation $ax^2 + bx + c = 0$ by calculating the Discriminant $(b^2 - 4ac)$ as

$(b^2 - 4ac) = 0$	Roots are real, equal and rational
$(b^2 - 4ac) > 0$	Roots are real, distinct and irrational
$(b^2 - 4ac) < 0$	Roots are imaginary

And then calculates the roots by using Quadratic Formula.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$