20MCA131

PROGRAMMING LAB

Mark distribution

Total Marks	CIE	ESE	ESE Duration
100	50	50	3 hours

Continuous Internal Evaluation Pattern:

Maximum Marks: 50	
Attendance	15%
Maintenance of daily lab record and GitHub management	20%
Regular class viva	15%
Timely completion of day to day tasks	20%
Tests/Evaluation	30%

Course Contents and Lab Schedule

Topic
1. Input, Output and Import Functions
2. Operators
3. Data Types
4. Decision Making & Loops
5. Functions
6. Modules and Packages
7. File Handling
8. Object Handling
9. Exception Handling
10. Regular Expressions

End Semester Examination Pattern:

Maximum Marks: 50			
Verification of Daily program record and Git Repository			5 marks
Viva			10 marks
Problem solving (Based on difficulty level, one or more questions may be given)	Flowchart / Algorithm / Structured description of problem to explain how the problem can be solved / Interface Design	15%	35 marks
	Program correctness	50%	
	Code efficiency	15%	
	Formatted output and Pushing to remote Git repository	20%	
Total Marks			50 marks

How to Print Output in Python

In this example, “Hello, World!” is a string literal enclosed within double quotes. When executed, this statement will output the text to the console.

Python

```python
print("Hello, World!")
```

Output

```
Hello, World!
```

In the `print()` function, you output multiple variables, separated by a comma:

Example

```
x = "Python"
y = "is"
z = "awesome"
print(x, y, z)
```

You can also use the `+` operator to output multiple variables:

Example

```
x = "Python "
y = "is "
z = "awesome"
print(x + y + z)
```

Notice the space character after `"Python "` and `"is "`, without them the result would be "Pythonisawesome".

For numbers, the `+` character works as a mathematical operator:

Example

```
x = 5
y = 10
print(x + y)
```

In the `print()` function, when you try to combine a string and a number with the `+` operator, Python will give you an error:

Example

```
x = 5
y = "John"
print(x + y)
```

The best way to output multiple variables in the `print()` function is to separate them with commas, which even support different data types:

Example

```
x = 5
y = "John"
print(x, y)
```

Print Single and Multiple variable in Python

The code assigns values to variables name and age, then prints them with labels.

Python

```python
name = "Alice"
age = 30
print("Name:", name, "Age:", age)
```

Output

```
Name: Alice Age: 30
```

Format Output Handling in Python

Output formatting in Python with various techniques including the format() method, manipulation of the sep and end parameters, f-strings, and the versatile % operator. These methods enable precise control over how data is displayed, enhancing the readability and effectiveness of your Python programs.

Example 1: Using Format()

Python

```python
amount = 150.75
print("Amount: ${:.2f}".format(amount))
```

Output

```
Amount: $150.75
```

Example 4: Using % Operator

We can use ‘%’ operator. % values are replaced with zero or more value of elements. The formatting using % is similar to that of ‘printf’ in the C programming language.

- %d –integer
- %f – float
- %s – string
- %x –hexadecimal
- %o – octal

Python

```python
# taking two inputs at a time
x, y = input("Enter two values: ").split()
print("Number of boys: ", x)
print("Number of girls: ", y)

# taking three inputs at a time
x, y, z = input("Enter three values: ").split()
print("Total number of students: ", x)
print("Number of boys is : ", y)
print("Number of girls is : ", z)
```

Output

```
Enter two values: 5 10
Number of boys:  5
Number of girls:  10
Enter three values: 5 10 15
Total number of students:  5
Number of boys is :  10
Number of girls is :  15
```

Take Multiple Input in Python

The code takes input from the user in a single line, splitting the values entered by the user into separate variables for each value using the split() method. Then, it prints the values with corresponding labels, either two or three, based on the number of inputs provided by the user.

Python

```python
# Taking input from the user
num = int(input("Enter a value: "))

add = num + 5

# Output
print("The sum is %d" %add)
```

Output

```
Enter a value: 50The sum is 55
```

INPUT

Python input() function is used to take user input. By default, it returns the user input in form of a string.

Syntax: *input(prompt)*

How to Change the Type of Input in Python

By default input() function helps in taking user input as string. If any user wants to take input as int or float, we just need to typecast it.

How to Print Names in Python

The code prompts the user to input a string (the color of a rose), assigns it to the variable color, and then prints the inputted color.

Python

```python
# Taking input as string
color = input("What color is rose?: ")
print(color)
```

Output

```
What color is rose?: Red
Red
```

How to Print Numbers in Python

The code prompts the user to input an integer representing the number of roses, converts the input to an integer using typecasting, and then prints the integer value.

Python

```python
# Taking input as int
# Typecasting to int
n = int(input("How many roses?: "))
print(n)
```

Output

```
How many roses?: 8
8
```

How to Print Float/Decimal Number in Python

The code prompts the user to input the price of each rose as a floating-point number, converts the input to a float using typecasting, and then prints the price.

Python

```python
# Taking input as float
# Typecasting to float
price = float(input("Price of each rose?: "))
print(price)
```

Output

```
Price of each rose?: 50.30
50.3
```

Find DataType of Input in Python

In the given example, we are printing the type of variable x. We will determine the type of an object in Python.

File Edit Format Run Options Window Help

```
a = "Hello World"
b = 10
c = 11.22
d = ("S1 ", "MCA", "CET")
e = ["S1", "MCA", "CET"]
f = {"S1": 1, "MCA":2, "CET":3}


print(type(a))
print(type(b))
print(type(c))
print(type(d))
print(type(e))
print(type(f))
```

IDLE Shell 3.12.6

File Edit Shell Debug Options Window Help

```
Python 3.12.6 (tags/v3.12.6:a4a2d2b, Sep  6 2024, 20:11:23) [MSC v.1940 64 bit
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/sabina/AppData/Local/Programs/Python/Python312/P2.py
<class 'str'>
<class 'int'>
<class 'float'>
<class 'tuple'>
<class 'list'>
<class 'dict'>
>>>
```

How to Write Output in Python?

Writing output in Python can be done in several ways:

1. ***Printing to the Console**: Use the `print()` function to send output to the standard output (usually the console).**Example:***
2. ***Writing to a File**: Use Python's file handling capabilities to write data to a file.**Example:***
3. ***Using Standard Output Streams**: For more control, you can use the `sys.stdout` to write directly to the standard output stream, which is useful for more complex console applications.**Example:***

```
print("Hello, World!")
```

```
with open('output.txt', 'w') as file:
    file.write("This is some output data.")
```

```
import sys
sys.stdout.write("This is some output data.\n")
```

Each method of output is suited to different scenarios, from simple scripts to complex applications interacting with file systems and external interfaces

Multiple Variables in Python

Python allows you to work with multiple variables simultaneously by defining them in a single statement or line. This can be done using various way , here we are explaining some generally used method :

- By Using Unpacking
- Multiple Assignment
- Using F-Strings

Python Print Variable by using Unpacking

Unpacking is a process of assigning values from data structure to multiple variables in a one way. In this example, we have defined three variables **'name', 'age', and 'city'** and assigned values to them simultaneously. Each variable receives its corresponding value from the right-hand side of the assignment.

Python3

```
name, age, city = "John", 30, "New York"
print(name)
print(age)
print(city)
```

Output:

```
John
30
New York
```

Python Print Variable by Multiple Assignment

You can assign multiple values to multiple variables using multiple assignment operators. In below code we assigned the values **1, 2, and 3** to variables **a, b, and c,** and then print the values using Multiple Assignment.

Python3

```
a,b,c = 1,2,3
print(a,b,c)
```

Output:

```
1 2 3
```

Python Print Variable using F-Strings

You can assign multiple values to multiple variables using F-Strings . Here, F-strings are utilized to insert variables (`name`, `age`, and `city`) into the string, allowing the creation of a customizable formatted message.

Python3

```
name = "Alice"
age = 30
city = "Wonderland"

# Using F-string to incorporate multiple variables
message = f"Hello, my name is {name}, I am {age} years old, and I live in {city}."

# Printing the formatted message
print(message)
```

Output :

```
Hello, my name is Alice, I am 30 years old, and I live in Wonderland.
```

Creating Variables

Python has no command for declaring a variable.

A variable is created the moment you first assign a value to it.

Example

```
x = 5
y = "John"
print(x)
print(y)
```

```
5
John
```

Variables do not need to be declared with any particular *type*, and can even change type after they have been set.

Example

```
x = 4       # x is of type int
x = "Sally" # x is now of type str
print(x)
```

Casting

If you want to specify the data type of a variable, this can be done with casting.

Example

```
x = str(3)    # x will be '3'
y = int(3)    # y will be 3
z = float(3)  # z will be 3.0
```

Get the Type

You can get the data type of a variable with the `type()` function.

Example

```
x = 5
y = "John"
print(type(x))
print(type(y))
```

```
x = 5
y = "John"
print(type(x))
print(type(y))
```

```
<class 'int'>
<class 'str'>
```

Single or Double Quotes?

String variables can be declared either by using single or double quotes:

Example

```
x = "John"
# is the same as
x = 'John'
```

Case-Sensitive

Variable names are case-sensitive.

Example

This will create two variables:

```
a = 4
A = "Sally"
#A will not overwrite a
```

Variable Names

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume). Rules for Python variables:

- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (age, Age and AGE are three different variables)
- A variable name cannot be any of the Python keywords.

Example

Legal variable names:

```
myvar = "John"
my_var = "John"
_my_var = "John"
myVar = "John"
MYVAR = "John"
myvar2 = "John"
```

Example

Illegal variable names:

```
2myvar = "John"
my-var = "John"
my var = "John"
```

Many Values to Multiple Variables

Python allows you to assign values to multiple variables in one line:

Example

```
x, y, z = "Orange", "Banana", "Cherry"
print(x)
print(y)
print(z)
```

One Value to Multiple Variables

And you can assign the *same* value to multiple variables in one line:

Example

```
x = y = z = "Orange"
print(x)
print(y)
print(z)
```

Unpack a Collection

If you have a collection of values in a list, tuple etc. Python allows you to extract the values into variables. This is called *unpacking*.

Example

Unpack a list:

```
fruits = ["apple", "banana", "cherry"]
x, y, z = fruits
print(x)
print(y)
print(z)
```

Built-in Data Types

In programming, data type is an important concept.

Variables can store data of different types, and different types can do different things.

Python has the following data types built-in by default, in these categories:

Text Type:	`str`
Numeric Types:	`int`, `float`, `complex`
Sequence Types:	`list`, `tuple`, `range`
Mapping Type:	`dict`
Set Types:	`set`, `frozenset`
Boolean Type:	`bool`
Binary Types:	`bytes`, `bytearray`, `memoryview`
None Type:	`NoneType`