Python

Experiment No. 1

Aim: Write python programs to understand Expressions, Variables, and Quotes, Basic Math operations, Input and outputs.

Problem Statements:

1. Write python programs to understand Comments, Datatypes, Expressions, Input and Output Functions.
2. Write a python program display an employee information like Emp_ID, Name, mail_Id, Mobile No, height, weight and gender, salary
3. Write python programs to implement various mathematical functions.

Theory:

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Variables in Python

Variables need not be declared first in python. They can be used directly. Variables in python are case-sensitive as most of the other programming languages. The assignment statement gives a value to a variable

Variable names and keywords

Variable names can be arbitrarily long. They can contain both letters and digits, but they have to begin with a letter or an underscore. Although it is legal to use uppercase letters, by convention we don't. If you do, remember that case matters. Bruce and bruce are different variables.

The underscore character (_) can appear in a name. It is often used in names with multiple words, such as my_name or price_of_tea_in_china.

There are some situations in which names beginning with an underscore have special meaning, so a safe rule for beginners is to start all names with a letter.

Example:

Code:

```
a = 2
A = 8
print (a)
print (A)
```

Output:

```
2
8
```

Multiple Assignment

Python allows you to assign a single value to several variables simultaneously which means you can create multiple variables at a time.

For example –

a = b = c =1000

Statements

Statement in Python can be extended to one or more lines using parentheses (), braces {}, square brackets [], semi-colon (;), and continuation character slash (\). When the programmer needs to do long calculations and cannot fit his statements into one line, one can make use of these characters.

Example of line continuation:

total = item_one + \

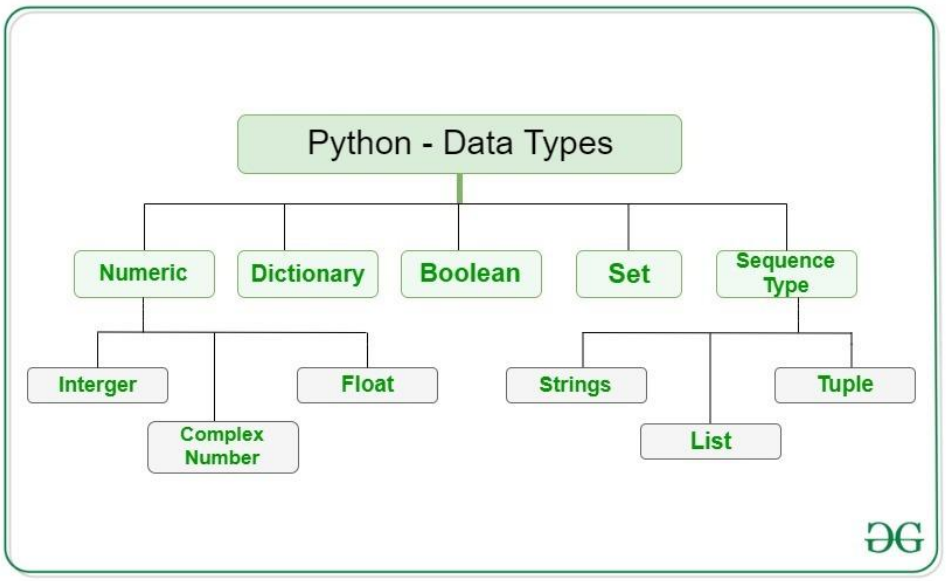
item_two + \

item_three

Data Types in Python

What is Python type() Function?

To define the values of various data types and check their data types we use the type() function. Consider the following examples.



Numeric Data Type in Python

The numeric data type in Python represents the data that has a numeric value. A numeric value can be an integer, a floating number, or even a complex number. These values are defined as Python int, Python float, and Python complex classes in Python.

Integers – This value is represented by int class. It contains positive or negative whole numbers (without fractions or decimals). In Python, there is no limit to how long an integer value can be.

Float – This value is represented by the float class. It is a real number with a floating-point representation. It is specified by a decimal point. Optionally, the character e or E followed by a positive or negative integer may be appended to specify scientific notation.

Complex Numbers – Complex number is represented by a complex class. It is specified as (real part) + (imaginary part)j. For example – 2+3j

Sequence Data Type in Python

The sequence Data Type in Python is the ordered collection of similar or different data types. Sequences allow storing of multiple values in an organized and efficient fashion. There are several sequence types in Python –

Python String

Python List

Python Tuple

String Data Type

Strings in Python 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.

Creating String

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

Example

```
String1 = "GeeksForGeeks"

print("Initial String: ")

print(String1)


# Printing First character

print("\nFirst character of String is: ")

print(String1[0])


# Printing Last character

print("\nLast character of String is: ")

print(String1[-1])
```

Types of statements in Python

Multi-Line Statements, Python Conditional and Loop Statements like Python If-else, Python for loop, Python while loop, Python try-except, Python with statement, Python Expression statements like pass, statement, del statement, return statement, import statement, Python continue and break statement.

There are different types of statements in Python language such as Assignment statements, Conditional statements, Looping statements, etc. The token character NEWLINE is used to end a statement in Python. It signifies that each line of a Python script contains a statement. These all help the user to get the required output.

Expressions in Python

A combination of operands and operators is called an expression. The expression in Python produces some value or result after being interpreted by the Python interpreter. An expression in Python is a combination of operators and operands.

An example of expression can be: x=x+10. In this expression, the first 10 is added to the variable x. After the addition is performed, the result is assigned to the variable x.

An expression in Python is very different from statements in Python. A statement is not evaluated for some results. A statement is used for creating variables or for displaying values.

Code:

```
x = 25          # a statement
x = x + 10      # an expression
print(x)
```

Output:

```
True
True
True
```

An expression in Python can contain **identifiers**, **operators**, and **operands**. Let us briefly discuss them.

An **identifier** is a name that is used to define and identify a class, variable, or function in Python.

An **operand** is an object that is operated on. On the other hand, an **operator** is a special symbol that performs the arithmetic or logical computations on the operands. There are many types of operators in Python, some of them are:

Python divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- **Identity operators**

Identity operators compare the memory locations of two objects. There are two Identity operators explained below

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

Code:

```
x = ["chiku","mango","apple"]

y = ["chiku","mango","apple"]
z = x
print(x is z)
# returns True because z is the same object as x
print(x isnot y)
# returns True because x is not the same object as y, even if
they have the same content
print(x == y)
# to demonstrate the difference betweeen "is" and "==": this
comparison returns True because x  is equal to y
```

Output:

```
True
True
True
```

- **Membership operators**

Python's membership operators test for membership in a sequence, such as strings, lists, or tuples. There are two membership operators as explained below −

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified	x in y, here in results in a 1 if x is a member of sequence y.

	sequence and false otherwise.	
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	x not in y, here not in results in a 1 if x is not a member of sequence y.

Code:

```
x = ["chiku","mango"]
print("mango"in x)

x = ["chiku","mango"]
print("grapes"notin x)
```

Output:

```
True
True
```

- Bitwise operators

Code:

```
# Bitwise AND (&)
a = 5    # 101
b = 3    # 011

print("Bitwise AND (&) of", a, "and", b, "is", a & b)

# Bitwise OR (|)
a = 5    # 101
b = 3    # 011

print("Bitwise OR (|) of", a, "and", b, "is", a | b)

# Bitwise XOR (^)
a = 5    # 101
b = 3    # 011

print("Bitwise XOR (^) of", a, "and", b, "is", a ^ b)

# Bitwise NOT (~)
a = 5    # 101
```

```
print("Bitwise NOT (~) of", a, "is", ~a)

# Bitwise Left Shift (<<)
a = 5    # 101

print("Bitwise Left Shift (<<) of", a, "by 1 is", a << 1)

# Bitwise Right Shift (>>)
a = 5    # 101

print("Bitwise Right Shift (>>) of", a, "by 1 is", a >> 1)
```

Output:

```
Bitwise AND (&) of 5 and 3 is 1
Bitwise OR (|) of 5 and 3 is 7
Bitwise XOR (^) of 5 and 3 is 6
Bitwise NOT (~) of 5 is -6
Bitwise Left Shift (<<) of 5 by 1 is 10
Bitwise Right Shift (>>) of 5 by 1 is 2
```

Quotes

Python accepts single ('), double (") and triple ('" or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

The triple quotes are used to span the string across multiple lines. For example, all the following are legal −

word = 'word'
sentence = "This is a sentence."
paragraph = """This is a paragraph. It is
made up of multiple lines and sentences."""

Mathematical operations:

Python has support for both mathematical operations and functions.

Operation	**Result**
x + y	sum of x and y.
x * y	multiplication of x and y.

x – y	difference of x and y.
x / y	division of x by y.
x % y	remainder of x/y
x ** y	x to the power of y
abs(x)	absolute value of x
sqrt(x)	square root of x

Mathematical Functions:
Python supports a wide variety of mathematical functions.

Function	Returns	Example
abs(x)	Returns the absolute value of x.	x = -35 x =abs(x) **print**(x)
cmp(x,y)	Returns -1 if x < y Returns 0 if x equals to y Returns 1 if x > y.	x =6 y =4 **print**(cmp(x,y))
exp(x)	Returns the exponential of x	**import**math x =6 **print**(math.exp(x))
log(x)	The natural logarithm of x	**import**math x =6 **print**(math.log(x))
log10(x)	The base-10 logarithm of x	**import**math x =6 **print**(math.log10(x))
pow(x,y)	The result of x**y	**import**math x =6 **print**(math.pow(x,2))
sqrt(x)	The square root of x	**import**math x =6 **print**(math.sqrt(x))

Input and outputs:

While Python provides us with two inbuilt functions to read the input from the keyboard.

input (prompt)

raw_input(prompt)

input (): This function first takes the input from the user and converts it into a string. The type of the returned object always will be <class 'str'>. It does not evaluate the expression it just returns the complete statement as String. For example, Python provides a built-in function called input which takes the input from the user. When the input function is called it stops the program and waits for the user's input. When the user presses enter, the program resumes and returns what the user typed.

```
val = input("Enter your value: ")

print(val)
```

When input() function executes program flow will be stopped until the user has given input.

The text or message displayed on the output screen to ask a user to enter an input value is optional i.e. the prompt, which will be printed on the screen is optional.

Whatever you enter as input, the input function converts it into a string. if you enter an integer value still input() function converts it into a string. You need to explicitly convert it into an integer in your code using typecasting.

There are various function that are used to take as desired input few of them are : –

```
int(input())

float(input())
```

raw_input(): This function works in older version (like Python 2.x). This function takes exactly what is typed from the keyboard, converts it to string, and then returns it to the variable in which we want to store it.

Taking multiple inputs from user in Python

Python user can take multiple values or inputs in one line by two methods.

Using split() method:

This function helps in getting multiple inputs from users. It breaks the given input by the specified separator. If a separator is not provided then any white space is a separator. Generally, users use a split() method to split a Python string but one can use it in taking multiple inputs.

Syntax :

input().split(separator, maxsplit)

Code:

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

Output:

```
Enter two values: 40 40
Number of boys:  40
Number of girls:  40
```

Using List comprehension:

List comprehension is an elegant way to define and create a list in Python. We can create lists just like mathematical statements in one line only. It is also used in getting multiple inputs from a user.

Code:

```
# taking two input at a time
x, y = [int(x)for x ininput("Enter two values: ").split()]
print("First Number is: ", x)
print("Second Number is: ", y)
```

Output:

```
Enter two values: 40 100
First Number is:  40
Second Number is:  100
```

Output

Python **print()** function prints the message to the screen or any other standard output device.

Example

In this example, we have created three variables integer, string and float and we are printing all the variables with print() function in Python.

Code:

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

Output:

```
Name: John
Age: 30
```

Practice problems:

1. WAP to find square root of a number

Code:

```
import math

n=int(input("Enter a number : "))
print("Square root of the number is : ",math.sqrt(n))
```

Output:

```
Enter a number : 10000
Square root of the number is :  100.0
```

OR

Code:

```
n=int(input("Enter a number : "))

print("Square root of the number is : ",n**(0.5))
```

Output:

```
Enter a number : 64
Square root of the number is :  8.0
```

2. WAP to find area of rectangle,circle, triangle

Code:

```
# Area of rectangle
print("\n---- Area of Rectangle ----")
l=int(input("Enter the length : "))
b=int(input("Enter the breadth : "))
print("Area of rectangle = ",l*b)

# Area of circle
print("\n---- Area of Circle ----")
r=int(input("Enter the radius : "))
print("Area of circle = ",3.14*r*r)

# Area of triangle
print("\n---- Area of Triangle ----")
h=int(input("Enter the height : "))
a=int(input("Enter the base length : "))
print("Area of triangle = ",0.5*h*a)
```

Output:

```
---- Area of Rectangle ----
Enter the length : 50
Enter the breadth : 20
Area of rectangle =  1000

---- Area of Circle ----
Enter the radius : 100
Area of circle =  31400.0

---- Area of Triangle ----
Enter the height : 100
Enter the base length : 10
Area of triangle =  500.0
```

Q. Write a python program display an employee information like Emp_ID, Name, mail_Id, Mobile No, height, weight and gender, salary

Code:

```
emp_id=int(input('Enter your Employee ID : '))
name=str(input('Enter your Name : '))
mail_id=str(input('Enter your valid email id : '))
mobile_no=int(input('Enter your valid mobile number : '))
height=float(input('Enter your height (in cm) : '))
weight=float(input('Enter your weight (in kg) : '))
gender=str(input('Enter your gender : '))
salary=int(input('Enter your salary amount (in Rs.) : '))

print("\n---- Employee Details ----")
print("\nName - ",name,"\nID - ",emp_id,"\nMail id -
",mail_id,"\nMobile no. - ",mobile_no,"\nGender -
",gender,"\nHeight - ",height,"\nWeight - ",weight,"\nSalary -
",salary)
```

Output:

```
Enter your Employee ID : 60018230034
Enter your Name : Chaitanya
Enter your valid email id : sendittochaitanya@gmail.com
Enter your valid mobile number : 8356881366
Enter your height (in cm) : 180
Enter your weight (in kg) : 55
Enter your gender : Male
Enter your salary amount (in Rs.) : 6000000

---- Employee Details ----

Name -  Chaitanya
ID -  60018230034
Mail id -  sendittochaitanya@gmail.com
Mobile no. -  8356881366
Gender -  Male
Height -  180.0
Weight -  55.0
Salary -  6000000
```

Lab Outcome: Implemented Expressions, Variables, Quotes, Basic Math operations, Input and outputs through various examples.