Functions

Python Functions

A Python function is a block of statements designed to perform a specific task. The idea behind functions is to group commonly or repeatedly executed tasks together, creating a function. This way, instead of duplicating code for different inputs, we can simply call the function to reuse the code contained within it.

Benefits of Using Functions

Using functions in Python offers several advantages:

- **Increase Code Readability:** Functions make code more readable by breaking it into smaller, logical parts.
- **Increase Code Reusability:** Code contained within functions can be easily reused throughout your program.

Python Function Declaration

The syntax for declaring a function in Python is straightforward:

```python
def function_name(parameters):
    # Function code goes here
```

Types of Functions in Python

In Python, there are primarily two types of functions:

1. **Built-in Library Functions:** These are standard functions in Python that are provided as part of the Python language and are readily available for use. Examples of built-in functions include `print()`, `len()`, and `input()`.

2. **User-Defined Functions:** User-defined functions are functions that we can create ourselves based on our specific programming requirements. These functions allow us to define custom logic and encapsulate a set of instructions for reuse in our programs.

```python
# 1. WAP that takes 2 numbers as inputs from the user, it also
displays a menu of operations that the user is allowed to perform.
Prompt the user to enter the desired option and then apply the
operation on the numbers.

import string
```

```
def operation(a: float, b: float, opr: string):
    if opr =='+':
        return a+b
    elif opr =='-':
        return a-b
    elif opr =='*':
        return a*b
    elif opr =='/':
        return a/b
    elif opr =='%':
        return a%b
    elif opr =='**':
        return a**b
    elif opr =='//':
        return a//b
    else:
        return 'Invalid Operation'

a = float(input('Enter first number: '))
b = float(input('Enter second number: '))

opr = input("Please enter + for addition\nPlease enter - for
subtraction\nPlease enter * for multiplication\nPlease enter / for
division\nPlease enter % for modulus\nPlease enter ** for
exponentiation\nPlease enter // for integer division\nPlease enter
your character: ")
print(operation(a, b, opr))
```

```
200.0
```

Lambda Function:

A lambda function is a small anonymous function. It is an anonymous function because it does not have a name. It is defined using the lambda keyword, as opposed to the def keyword used for regular functions. It takes any number of arguments, but can only have one expression. It can be used to create a function object and assign it to a variable.

Syntax:

lambda arguments : expression

Example:

x = lambda a : a + 10 print(x(5))

Output:

15

```
# perform sum, sub, div, mult, mod from lambda function

sum = lambda a, b: a+b
sub = lambda a, b: a-b
div = lambda a, b: a/b
mult = lambda a, b: a*b
mod = lambda a, b: a%b

print(sum(a, b))
print(sub(a, b))
print(div(a, b))
print(mult(a, b))
print(mod(a, b))
```

```
30.0
-10.0
0.5
200.0
10.0
```

```
# WAP to calculate simple interest- Supose a person is a senior
citizen his r.o.i is 12% and rest all 10%

def simple_interest(p: float, r: float, t: float):
    return (p*r*t)/100

p = float(input('Enter principal amount: '))
t = float(input('Enter time: '))
age = int(input('Enter age: '))
if age >= 60:
    r = 12
else:
    r = 10
print(simple_interest(p, r, t))
```

```
50000.0
```