DATE : 12 june 2024

DAY : Wednesday

TOPICS : Python Functions, lambda function, recursive functions and all categories of Function arguements, list comprehension & dictionary comprehension
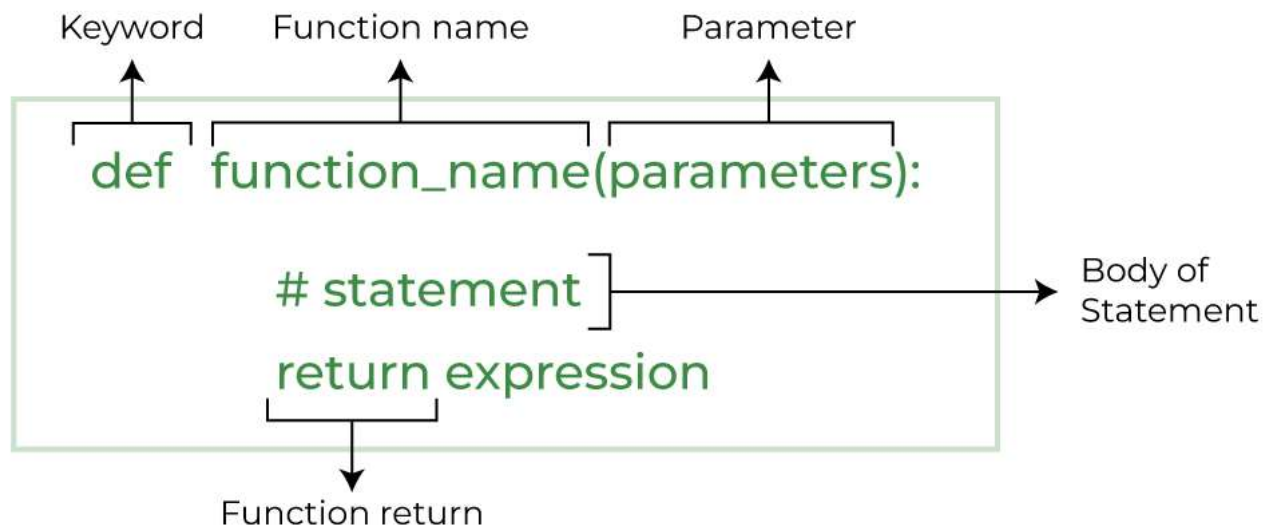
FUNCTIONS

Python Functions is a block of statements that return the specific task. The idea is to put some commonly or repeatedly done tasks together and make a function so that instead of writing the same code again and again for different inputs, we can do the function calls to reuse code contained in it over and over again.

Some Benefits of Using Functions

- Increase Code Readability
- Increase Code Reusability

Function Declaration

The syntax to declare a function is:

Types of Functions

1. Built-in Functions

2. User-defined Functions

3. Anonymous Functions (Lambda Functions)

4. Recursive Functions

1. Built-in Functions: These are pre-defined functions provided by a programming language or environment. Examples include print(), len() in Python, and printf() in C.

2. User-defined Functions: These are functions created by the programmer to perform specific tasks. They follow a defined structure with a name, parameters, and a body.

3. Anonymous Functions (Lambda Functions): These are functions without a name, often used for short, simple operations.

```
#lambda function
add=lambda x,y:x+y
print(add(2,3))
```

➔ 5

```
# A lambda function that adds 10 to the number passed as an argument
add_ten = lambda x: x + 10

# Using the lambda function
result = add_ten(9)
print(result)
```

➔ 19

```
# A lambda function that multiplies two numbers
multiply = lambda x, y: x * y

# Using the lambda function
result = multiply(4, 3)
print(result)
```

➔ 12

4. Recursive Functions: These functions call themselves within their definition, useful for tasks that can be divided into similar subtasks.

```
def factorial(n):
    # Base case: if n is 0, return 1
    if n == 0:
        return 1
    # Recursive case: n! = n * (n-1)!
    else:
        return n * factorial(n - 1)

# Using the recursive function
result = factorial(7)
print(result)
```

➔ 5040

```
def factorial(n):
    # Base case: if n is 0, return 1
    if n == 0:
        return 1
    # Recursive case: n! = n * (n-1)!
    else:
        return n * factorial(n - 1)

# Get user input
number = int(input("Enter a number to calculate its factorial: "))

# Ensure the input is a non-negative integer
if number < 0:
    print("Please enter a non-negative integer.")
else:
    # Using the recursive function
    result = factorial(number)
    print(f"The factorial of {number} is {result}")
```

➔ Enter a number to calculate its factorial: 8
    The factorial of 8 is 40320

⌄ Arguments in Functions

Arguments are values passed to a function when it is called. They can be categorized as follows:

1. Positional Arguments

2. Keyword Arguments

3. Default Arguments

4. Variable-length Arguments

1. Positional Arguments: These are arguments that are passed to a function in a specific order.

```
#positional argument
def subtract(a, b):
    return a - b

#result = add(2, 3)  # 2 and 3 are positional arguments


result = print(subtract(10, 7))
```

    3

```
result1 = print(subtract(9, 15))
```

    -6

```
def info(age, gender):
  print(f" Hi my age is :{age} years old & my gender is {gender}")


shagun = info("14", "Female")
```

     Hi my age is :14 years old & my gender is Female

2. Keyword Arguments: These are arguments passed to a function by explicitly naming each parameter and its corresponding value.

```
#keyword arguments
def greet(name, message):
    return f"{message}, {name}!"


Greeting = print(greet(name = "Harsimar", message = "Hi"))
```

    Hi, Harsimar!

```
Greeting1 = print(greet(message = "Hello", name = "Divyansh"))
```

    Hello, Divyansh!

3. Default Arguments: These are arguments that assume a default value if no value is provided during the function call.

```
#default arguments
def greet(name, message="Hello"):
    return f"{message}, {name}!"

result1 = print(greet("Alice"))          # Uses default message "Hello"
result2 = print(greet("Bob", "Hi"))      # Uses provided message "Hi"
```

    Hello, Alice!
    Hi, Bob!

```
Greet = print(greet("Mehakpreet"))
```

    Hello, Mehakpreet!

4. Variable-length Arguments: *args (Non-keyword arguments): Allows a function to accept any number of positional arguments.

kwargs (Keyword arguments): Allows a function to accept any number of keyword arguments.

```
#variable length arguments(*args)
def sum_all(*a):
    return sum(a)

#result = sum_all(1, 2, 3, 4)  # Accepts multiple positional arguments


Summation = print(sum_all(67, 12, 178, 19, 90, 14, 25, 12, 45))
```

```
462
```

```
#**kwargs
def print_details(**g):
    for key, value in g.items():
        print(f"{key}: {value}")

print_details(name="Alice", age=30, city="New York")
```

```
name: Alice
age: 30
city: New York
```

```
print_details(Locality = "Model Town", Address = "SCF, BLOCK-3, Third floor", H_no= "3", St_no = "4")
```

```
Locality: Model Town
Address: SCF, BLOCK-3, Third floor
H_no: 3
St_no: 4
```

⌄ Questions :

1.Write a Python function that takes a string as input and returns the reverse of the string.

```
def reverse(a):
  print(a[::-1])
str = reverse("heer")
```

```
reeh
```

2.Create a list of numbers. Write a function that finds and returns the maximum value in the list without using the built-in max() function.

```
def Num_list(*num):
  largest = num[0]
  for i in range(1,len(num)):
    if num[i]>largest:
      largest=num[i]
  return(largest)
number= print(Num_list(1,2,9,55))
```

```
55
```

3.Define a function that accepts a list of integers and returns a new list containing only the even numbers from the original list.

```
def is_even_num(num):
    enum = []
    for i in num:
        if i % 2 == 0:
            enum.append(i)
    return enum
print(is_even_num([1, 2, 3, 4, 5, 6, 7, 8, 9]))
```

```
[2, 4, 6, 8]
```

4.Implement a Python function to check if a given word is a palindrome (reads the same backward as forward).

```
def isPalindrome(s):
    return s == s[::-1]
s = "malayalam"
ans = isPalindrome(s)
if ans is True:
    print(f"{s} is Palindrome")
else:
    print(f"{s} is not palindrome")
```

```
malayalam is Palindrome
```

5.Develop a Python function that calculates the sum of squares for a given range of numbers.

```python
def squaresum(n):
  sum = 0
  for i in range(1, n+1):
        sum = sum + (i * i)
  return sum
n = 4
print(squaresum(n))
```

30

6.Write a recursive function that calculates the Fibonacci sequence for a given term.

```python
def fibonacci(n):
    if n <= 0:
        return "Invalid input. The input should be a positive integer."
    elif n == 1:
        return 0
    elif n == 2:
        return 1
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)
def print_fibonacci_series(n):
    if n <= 0:
        print("Invalid input. The input should be a positive integer.")
    else:
        print("Fibonacci Series:")
        for i in range(1, n + 1):
            print(fibonacci(i), end=" ")
num = 10
print_fibonacci_series(num)
```

    Fibonacci Series:
    0 1 1 2 3 5 8 13 21 34

Double-click (or enter) to edit

∨ LIST COMPREHENSION

- List Comprehensions provide an elegant way to create new lists.
- It consists of brackets containing an expression followed by a for clause, then zero or more for or if clauses.


```python
mystring = "WELCOME"
mylist = [ i for i in mystring ] # Iterating through a string Using List Comprehension
mylist
```

['W', 'E', 'L', 'C', 'O', 'M', 'E']

```python
mylist1 = [ i for i in range(40) if i % 2 == 0] # Display all even numbers between 0 and 40
mylist1
```

[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]

```python
mylist2 = [ i for i in range(40) if i % 2 == 1] # Display all odd numbers between 0 and 40
mylist2
```

    [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39]

```python
mylist3 = [num**2 for num in range(10)] # calculate square of all numbers between 0 and 10
mylist3
```

    [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

```python
# Multiple whole list by 10
list1 = [2,3,4,5,6,7,8]
list1 = [i*10 for i in list1]
list1
```

    [20, 30, 40, 50, 60, 70, 80]

```python
#List all numbers divisible by 3 , 9 & 12 using nested "if" with List Comprehensi
mylist4 = [i for i in range(200) if i % 3 == 0 if i % 9 == 0 if i % 12 == 0]
mylist4
```

    [0, 36, 72, 108, 144, 180]

```python
# Extract numbers from a string
mystr = "One 1 two 2 three 3 four 4 five 5 six 6789"
numbers = [i for i in mystr if i.isdigit()]
numbers
```

    ['1', '2', '3', '4', '5', '6', '7', '8', '9']

```python
# Extract letters from a string
mystr = "One 1 two 2 three 3 four 4 five 5 six 6789"
numbers = [i for i in mystr if i.isalpha()]
numbers
```

    ['O',
     'n',
     'e',
     't',
     'w',
     'o',
     't',
     'h',
     'r',
     'e',
     'e',
     'f',
     'o',
     'u',
     'r',
     'f',
     'i',
     'v',
     'e',
     's',
     'i',
     'x']

⌄ DICTIONARY COMPREHENSION

```
double = {i:i*2 for i in range(10)} #double each value using dict comprehension
double
```

```
{0: 0, 1: 2, 2: 4, 3: 6, 4: 8, 5: 10, 6: 12, 7: 14, 8: 16, 9: 18}
```

```
square = {i:i**2 for i in range(10)}
square
```

```
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}
```

```
key = ['one' , 'two' , 'three' , 'four' , 'five']
value = [1,2,3,4,5]
mydict = {k:v for (k,v) in zip(key,value)} # using dict comprehension
mydict
```

```
{'one': 1, 'two': 2, 'three': 3, 'four': 4, 'five': 5}
```

```
mydict1 = {'a':10 , 'b':20 , 'c':30 , 'd':40 , 'e':50}
mydict1 = {k:v/10 for (k,v) in mydict1.items()} # Divide all values in a dictionary by 10
mydict1
```

```
{'a': 1.0, 'b': 2.0, 'c': 3.0, 'd': 4.0, 'e': 5.0}
```

```
str1 = "Natural Language Processing"
mydict2 = {k:v for (k,v) in enumerate(str1)} # Store enumerated values in a dictionary
mydict2
```

```
{0: 'N',
 1: 'a',
 2: 't',
 3: 'u',
 4: 'r',
 5: 'a',
 6: 'l',
 7: ' ',
 8: 'L',
 9: 'a',
 10: 'n',
 11: 'g',
 12: 'u',
 13: 'a',
 14: 'g',
 15: 'e',
 16: ' ',
 17: 'P',
 18: 'r',
 19: 'o',
 20: 'c',
 21: 'e',
 22: 's',
 23: 's',
 24: 'i',
 25: 'n',
 26: 'g'}
```

```
str1 = "abcdefghijklmnopqrstuvwxyz"
mydict3 = {i:i.upper() for i in str1} # Lower to Upper Case
mydict3
```

```
{'a': 'A',
 'b': 'B',
 'c': 'C',
 'd': 'D',
 'e': 'E',
 'f': 'F',
 'g': 'G',
 'h': 'H',
 'i': 'I',
 'j': 'J',
 'k': 'K',
 'l': 'L',
 'm': 'M',
 'n': 'N',
 'o': 'O',
 'p': 'P',
 'q': 'Q',
 'r': 'R',
 's': 'S',
 't': 'T',
 'u': 'U',
 'v': 'V',
 'w': 'W',
 'x': 'X',
 'y': 'Y',
 'z': 'Z'}
```

```python
ls = 1
ps = [ 12, 14, 17, 9, 13, 20, 1, 23, 12]
for i in ps:
  if i == ls:
    print(f"Lowest score found at index {ps.index(i)}")
```

```
Lowest score found at index 6
```

```python
# List of player scores
scores = [85, 92, 78, 64, 71, 88, 79]

# Initialize the minimum score to the first score in the list
min_score = scores[0]

# Iterate through the scores to find the minimum
for score in scores:
    if score < min_score:
        min_score = score

# Print the lowest score
print("The lowest score is:", min_score)
```

```
The lowest score is: 64
```