DAY 11 **[07 - JULY - 2025]**

FUNCTIONS IN PYTHON

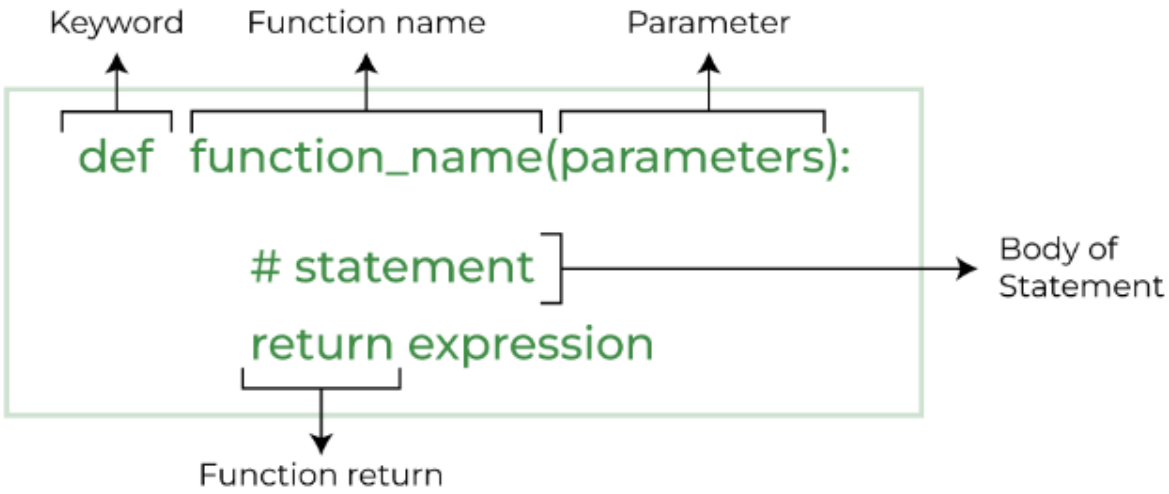
Python Functions is a block of statements that does a specific task. The idea is to put some commonly or repeatedly done task together and make a function so that instead of writing the same code again and again for different inputs, we can do the function calls to reuse code contained in it over and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

Benefits of Using Functions

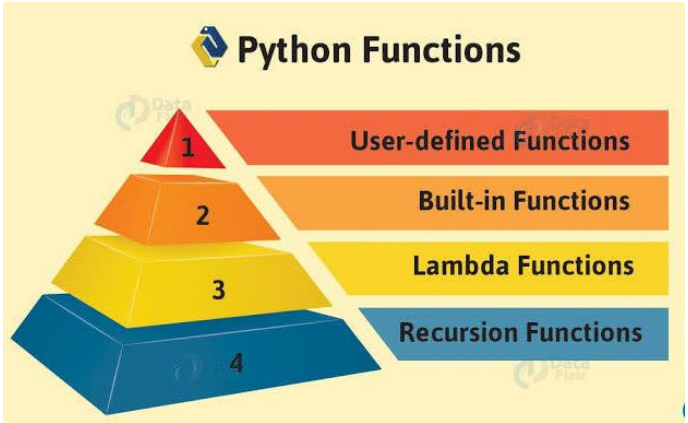
- Code Reuse
- Reduced code length
- Increased redability of code

Python Function Declaration

The syntax to declare a function is:



Types of functions:



- **Built-in function:** These are Standard functions in Python that are available to use.
- **User-defined function:** We can create our own functions based on our requirements.
- **Lambda Function:** a small, anonymous function defined without a name, often used for concise, single-expression operations.
- **Recursive Function:** a function that calls itself within its own code.

PRACTICE QUESTIONS:

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

```
[ ]  def reverse(string):
         return string[::-1]
     string = input("Enter a string: ")
     print("Reverse of the string is:", reverse(string))
```

```
Enter a string: QWERTY
Reverse of the string is: YTREWQ
```

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 max(numbers):
    max_value = numbers[0]
    for number in numbers:
        if number > max_value:
            max_value = number
    return max_value


numbers = input("Enter a list of numbers:")
print("Maximum value in the list is:", max(numbers))
```

```
Enter a list of numbers:7 5 8 2 9 3 1 4 0 6
Maximum value in the list is: 9
```

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 even(numbers):
        even_numbers = []
        for number in numbers:
            if number % 2 == 0:
                even_numbers.append(number)
        return even_numbers


    numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    print("Even numbers in the list are:", even(numbers))
```

```
Even numbers in the list are: [2, 4, 6, 8, 10]
```

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

```
def palindrome(word):
  if word == word[::-1]:
      print("The word is a palindrome")
  else:
      print("The word is not a palindrome")


word = input("Enter a word: ")
palindrome(word)
```

```
Enter a word: YAHOO
The word is not a palindrome
```

5.Create a dictionary with student names as keys and their corresponding ages as values. Write a function to find and print the names of students who are above a certain age.

```
def above_age(students, age_above):
  above_age_students = []
  for student, age in students.items():
    if age > age_above:
      above_age_students.append(student)
  return above_age_students


students = {"Jot": 20, "Mehak": 19, "Aran": 22, "Karan": 21}
age_above = 20

print("Students above age", age_above, "are:", above_age(students, age_above))
```

```
Students above age 20 are: ['Aran', 'Karan']
```

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

```
[ ] def sum_is(start, end):
      sum = 0
      for i in range(start, end + 1):
        sum = sum + ( i ** 2 )
      return sum


    print("Sum of squares for a given range of numbers is:", sum_is(1, 5))          # (1*1)+(2*2)+(3*3)+(4*4)+(5*5)=55
```

```
Sum of squares for a given range of numbers is: 55
```

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

```
def fibonacci(n):
  if n <= 0:
    return 0
  elif n == 1:
    return 1
  else:
    return fibonacci(n-1) + fibonacci(n-2)

m= int(input("Enter the number:"))
print("Fibonacci sequence for a given term is:", fibonacci(m))
```

```
Enter the number:13
Fibonacci sequence for a given term is: 233
```

11.Write a function that takes a list of numbers and returns a new list containing only the unique elements.

```
def unique(numbers):
  unique_numbers = []
  for number in numbers:
    if number not in unique_numbers:
      unique_numbers.append(number)
  return unique_numbers


numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1, 2, 3]
print("Unique elements in the list are:", unique(numbers))
```

```
Unique elements in the list are: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

12.Create a Python function that accepts a string and counts the occurrences of each character. Return the result as a dictionary.

```
def count_characters(string):
  character_count = {}
  for character in string:
    if character in character_count:
      character_count[character] += 1
    else:
      character_count[character] = 1
  return character_count

string = input("Enter a string:")
print("The number of occurrences of each character is:", count_characters(string))
```

```
Enter a string:yahooo
The number of occurrences of each character is: {'y': 1, 'a': 1, 'h': 1, 'o': 3}
```

13.Implement a function to calculate the average of a list of numbers without using the built-in sum() and len() functions.

```
def calculate_average(numbers):
    total = 0
    count = 0
    for num in numbers:
        total += num
        count += 1
    if count == 0:
        return 0

    average = total / count
    return average


nums = [10, 20, 30, 40, 50]
print("Average:", calculate_average(nums))
```

```
Average: 30.0
```

14.Write a function that checks if a given year is a leap year. A leap year is divisible by 4 but not divisible by 100 unless it is divisible by 400.

```
def is_leap_year(year):
    if (year % 4 == 0):
        if (year % 100 != 0) or (year % 400 == 0):
            return True
    return False

print(is_leap_year(2025))
```

```
False
```

15.Design a function that takes a list of strings and returns a new list with only the strings that have more than 5 characters.

```
def filter_long_strings(strings):
    result = []
    for s in strings:
        if len(s) > 5:
            result.append(s)
    return result

words = ["apple", "banana", "kiwi", "strawberry", "pear", "orange"]
print(filter_long_strings(words))
```

```
['banana', 'strawberry', 'orange']
```

16.Create a function that accepts a sentence and counts the number of vowels (a, e, i, o, u) in it.

```
def count_vowels(sentence):
    vowels = 'aeiouAEIOU'
    count = 0
    for char in sentence:
        if char in vowels:
            count += 1
    return count

s = "I am study Machine Learning."
print("Number of vowels:", count_vowels(s))
```

```
Number of vowels: 9
```

17.Write a Python function that takes a number and checks whether it is a prime number.

```
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

print(is_prime(7))
```

```
True
```

18.Implement a function that reverses the order of words in a given sentence.

```
def reverse_words(sentence):
    words = sentence.split()
    reversed_words = words[::-1]
    reversed_sentence = ' '.join(reversed_words)
    return reversed_sentence


s = "I am study now!!!"
print(reverse_words(s))
```

```
now!!! study am I
```

19.Create a function to find and return the second-largest number in a list of integers.

```
def second_largest(numbers):
    nums = list(set(numbers))
    if len(nums) < 2:
        return None
    nums.sort()
    return nums[-2]


print(second_largest([10, 20, 4, 45, 99]))
```

```
45
```

20.Write a function that takes a string and returns True if it is a palindrome (ignoring spaces and case), and False otherwise.

```
[ ] def is_palindrome(s):
        s = s.replace(" ", "").lower()
        return s == s[::-1]

    print(is_palindrome("Race car"))
```

```
True
```