FUNCTIONS

In Python, a function is defined using the def keyword, followed by the function name, parentheses containing any parameters, and a colon. The function body is indented and contains the code to be executed. .

Types of 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

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

```
#define a function
#syntax
def function_name(paramenters):
  #statement
  pass
```

```
def greet():
  print("welcome")
```

```
#call a function
greet()
```

```
welcome
```

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

```
#Recursive function
def factorial(n):
  if n ==1:
    return 1
  else:
    return n*factorial(n-1)
```

Arguments in Functions

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

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

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

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

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 **bold text

```
#Arbitary keyword
def info(**kwargs):
  for key,value in kwargs:
    print("kwargs.iteam()")
```

```
#positinal 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
```

```
Mohit = info("Male", "21")
```

```
 Hi my age is :Male years old & my gender is 21
```

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

#result = greet(name="Alice", message="Hello")  # name and message are keyword arguments
```

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

```
Hi, Harsimar!
```

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

```
Hello, Divyansh!
```

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

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

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

```
Hello, Mehakpreet!
```

```
#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")
```

```
  File "<ipython-input-36-1dbe4a7c4040>", line 1
    print_details(Locality = "Model Town", Address = "SCF, BLOCK-3, Third floor", H_no.= "3", St.no = "4")
                                                                                        ^
SyntaxError: positional argument follows keyword argument
```

PRACTICE QUESTIONS ON PYTHON FUNCTIONS:

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

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.

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

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

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.

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

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

8.Create a class called Rectangle with methods to calculate the area and perimeter. Initialize the class with the length and width as attributes.

9.Implement a function that takes a list of strings and returns a new list with the strings sorted by their lengths in ascending order.

10.Use a lambda function to filter a list of integers and return a new list containing only the numbers greater than 10.

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

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

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

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.

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.

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

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

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

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

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

```
# Secret message string
secret_message = "77722245672yugdegwd.90eddedudj3d."

# Initialize the index
index = 0

# Use a while loop to print each character until a period is found
while index < len(secret_message):
    if secret_message[index] == '.':
        break
    print(secret_message[index], end='')
    index += 1

print()  # To move to the next line after the loop ends
```

```
77722245672yugdegwd
```

```
#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
string=input("enter a name:-")
def rev(string):
   k=string
   if len(k)>0:
     return(k[::-1])
print(rev(string))
```

```
enter a name:-lofer
refol
```

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

```
list =([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
even_numbers =[num for num in list if num % 2 == 0]
print(even_numbers)
```

```
[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).

```
string=input("enter a name:-")
def check_paliindrome(string):
  if string == string[::-1]:
    return True
  else:
    return False
print(check_paliindrome(string))
```

```
enter a name:-navi
False
```