Python Programming - 2301CS404

Lab - 8

Name: Jadeja Rudrarajsinh

Enrollment No: 23010101411

Roll No: 487

User Defined Function

01) Write a function to calculate BMI given mass and height. (BMI = mass/h**2)

In [2]:
```
def calc_bmi(mass,height):
    bmi = mass/(height**2)
    return bmi

mass = float(input("Enter your mass(weight):"))
height = float(input("Enter yuor weight:"))
bmi = calc_bmi(mass,height)
print(f"your bmi is: {bmi}")
```

```
your bmi is: 0.0015833333333333333
```

02) Write a function that add first n numbers.

In [4]:
```
def add_numbers(n):
    total = (n*(n+1)/2)
    return total

n =10
result = add_numbers(n)
print(f"addtion of numbers is : {result}")
```

```
addtion of numbers is : 55.0
```

03) Write a function that returns 1 if the given number is Prime or 0 otherwise.

In [8]:
```
def is_prime(num):
    if num <= 1:
        return 0
    for i in range (2,int(num**0.5)+1):
        if num % i == 0:
            return 0
    return 1

num = int(input("Enter a number: "))
result = is_prime(num)
print(f"{num} is prime number: {result} ")
```

```
7 is prime number: 1
```

04) Write a function that returns the list of Prime numbers between given two numbers.

In [10]:
```
def primes_in_range(start, end):
    def is_prime(number):
        if number <= 1:
            return False
        for i in range(2, int(number ** 0.5) + 1):
            if number % i == 0:
                return False
        return True

    prime_numbers = [num for num in range(start, end + 1) if is_prime(num)]
    return prime_numbers

start = 10
end = 50
prime_list = primes_in_range(start, end)
print(f"Prime numbers between {start} and {end}: {prime_list}")
```

```
Prime numbers between 10 and 50: [11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```

05) Write a function that returns True if the given string is Palindrome or False otherwise.

In [2]:
```
s = input("Enter a string : ")
def isPalindrome(s):
    return s==s[::-1]
print(isPalindrome(s))
```

```
False
```

06) Write a function that returns the sum of all the elements of the list.

In [4]:
```
li = map(int,input("Enter space separated values : ").split())
def total(li):
    sum=0
```

```
    for i in li:
        sum += i
    return sum
print("Sum = ",total(li))
```

```
Sum =  10
```

07) Write a function to calculate the sum of the first element of each tuples inside the list.

In [29]:
```
def sum_of_first_element(tuple_list):
    return sum(t[0] for t in tuple_list)
tuple_list = [(1,2),(3,4),(5,6)]
result = sum_of_first_element(tuple_list)
print(f"The sum of the first elements is: {result}")
```

```
The sum of the first elements is: 9
```

08) Write a recursive function to find nth term of Fibonacci Series.

In [31]:
```
def fibonacci(n):
    if n<=0:
        return 0
    elif n==1:
        return 1
    else:
        return fibonacci(n-1)+fibonacci(n-2)
n = int(input("Enter the term position (n): "))
result = fibonacci(n)
print(f"The {n}th term of the Fibonacci Series is: {result}")
```

```
The 2th term of the Fibonacci Series is: 1
```

09) Write a function to get the name of the student based on the given rollno.

Example: Given dict1 = {101:'Ajay', 102:'Rahul', 103:'Jay', 104:'Pooja'} find name of student whose rollno = 103

In [33]:
```
def get_student_name(rollno, student_dict):
    return student_dict.get(rollno, "Roll number not found")
dict1 = {101: 'Ajay', 102: 'Rahul', 103: 'Jay', 104: 'Pooja'}
rollno = int(input("Enter the roll number: "))
name = get_student_name(rollno, dict1)
print(f"The name of the student with roll number {rollno} is: {name}")
```

```
The name of the student with roll number 21 is: Roll number not found
```

10) Write a function to get the sum of the scores ending with zero.

Example : scores = [200, 456, 300, 100, 234, 678]

Ans = 200 + 300 + 100 = 600

In [35]:
```
def sum_of_scores_ending_with_zero(scores):
    return sum(score for score in scores if score % 10 == 0)
scores = [200, 456, 300, 100, 234, 678]
result = sum_of_scores_ending_with_zero(scores)
print(f"The sum of scores ending with zero is: {result}")
```

```
The sum of scores ending with zero is: 600
```

11) Write a function to invert a given Dictionary.

hint: keys to values & values to keys

Before : {'a': 10, 'b':20, 'c':30, 'd':40}

After : {10:'a', 20:'b', 30:'c', 40:'d'}

In [37]:
```
def invert_dictionary(original_dict):
    return {value: key for key, value in original_dict.items()}
original_dict = {'a': 10, 'b': 20, 'c': 30, 'd': 40}
inverted_dict = invert_dictionary(original_dict)
print("Original Dictionary:", original_dict)
print("Inverted Dictionary:", inverted_dict)
```

```
Original Dictionary: {'a': 10, 'b': 20, 'c': 30, 'd': 40}
Inverted Dictionary: {10: 'a', 20: 'b', 30: 'c', 40: 'd'}
```

12) Write a function to check whether the given string is Pangram or not.

hint: Pangram is a string containing all the characters a-z atlest once.

"the quick brown fox jumps over the lazy dog" is a Pangram string.

In [49]:
```
def is_pangram(s):
    s = s.lower()
    alphabet_set = set("abcdefghijklmnopqrstuvwxyz")
    return alphabet_set.issubset(set(s))
input_string = input("Enter a string: ")
if is_pangram(input_string):
    print("The string is a pangram.")
else:
    print("The string is not a pangram.")
```

```
The string is not a pangram.
```

13) Write a function that returns the number of uppercase and lowercase letters in the given string.

example : Input : s1 = AbcDEfgh ,Ouptput : no_upper = 3, no_lower = 5

In [47]:
```
def count_upper_lower(s):
    no_upper = 0
    no_lower = 0
    for char in s:
        if char.isupper():
            no_upper += 1
```

```
        elif char.islower():
            no_lower += 1
    return no_upper, no_lower
s1 = input('Enter a String:')
no_upper, no_lower = count_upper_lower(s1)
print(f"Number of uppercase letters: {no_upper}")
print(f"Number of lowercase letters: {no_lower}")
```

```
Number of uppercase letters: 0
Number of lowercase letters: 3
```

14) Write a lambda function to get smallest number from the given two numbers.

In [51]:
```
smallest = lambda x, y: x if x < y else y
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
result = smallest(num1, num2)
print(f"The smallest number is: {result}")
```

```
The smallest number is: 1.0
```

15) For the given list of names of students, extract the names having more that 7 characters. Use filter().

In [53]:
```
students = ["Alice", "Bob", "Alexander", "Catherine", "David", "Elizabeth"]
def has_more_than_7_chars(name):
    return len(name) > 7
filtered_names = list(filter(has_more_than_7_chars, students))
print("Names with more than 7 characters:", filtered_names)
```

```
Names with more than 7 characters: ['Alexander', 'Catherine', 'Elizabeth']
```

16) For the given list of names of students, convert the first letter of all the names into uppercase. use map().

In [56]:
```
students = ["alice", "bob", "alexander", "catherine", "david", "elizabeth"]
def capitalize_first_letter(name):
    return name.capitalize()
capitalized_names = list(map(capitalize_first_letter, students))
print("Names with first letter capitalized:", capitalized_names)
```

```
Names with first letter capitalized: ['Alice', 'Bob', 'Alexander', 'Catherine',
'David', 'Elizabeth']
```

17) Write udfs to call the functions with following types of arguments:

1. Positional Arguments
2. Keyword Arguments
3. Default Arguments
4. Variable Legngth Positional(*args) & *variable length Keyword Arguments (*kwargs)
5. Keyword-Only & Positional Only Arguments

In [60]:
```
def positional_args(a, b):
    return a + b
```

```
def keyword_args(a, b):
    return a - b

def default_args(a, b=5):
    return a * b


def variable_length_positional(*args):
    return sum(args)


def variable_length_keyword(**kwargs):
    return kwargs


def keyword_only_args(*, a, b):
    return a + b


def positional_only_args(a, b, /):
    return a * b

print("Positional Arguments Result:", positional_args(10, 20))
print("Keyword Arguments Result:", keyword_args(b=20, a=10))
print("Default Arguments Result:", default_args(10))
print("Variable-Length Positional Arguments Result:", variable_length_positional
print("Variable-Length Keyword Arguments Result:", variable_length_keyword(name=
print("Keyword-Only Arguments Result:", keyword_only_args(a=10, b=20))
print("Positional-Only Arguments Result:", positional_only_args(10, 20))
```

```
Positional Arguments Result: 30
Keyword Arguments Result: -10
Default Arguments Result: 50
Variable-Length Positional Arguments Result: 15
Variable-Length Keyword Arguments Result: {'name': 'Alice', 'age': 25, 'city': 'N
ew York'}
Keyword-Only Arguments Result: 30
Positional-Only Arguments Result: 200
```