|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| C:\Users\Admin\Desktop\download.jpg | USMAN INSTITUTE OF TECHNOLOGY | | | | | |  |
|  |  |  |  |  |  |  |  |
|  | Department of Computer Science  CS121 Object Oriented Programming | | | | | |  |
|  |  | Lab # 05  Functions in Python | | | |  |  |
|  | Objective:  This experiment introduces the students to the concept of Functions in Python programming language | | | | | |  |
|  | **Name of Student:**  **Roll No: Sec.**  **Date of Experiment:** | | | | | |  |
|  | **Marks Obtained/Remarks:**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Signature:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | | |  |

**Lab 05: Functions In Python**

Functions are an essential part of the Python programming language: you might have already encountered and used some of the many fantastic functions that are built-in in the Python language or that come with its library ecosystem. However, as a Computer Scientist, you’ll constantly need to write your own functions to solve problems that your projects or problems poses to you.

We will introduce you to functions in Python. We will learn the following parts under function:

1. Functions vs Methods
2. Parameters vs Arguments
3. How to define a function: User-Defined Functions (UDFs)

You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task. To carry out that specific task, the function might or might not need multiple inputs. When the task is carried out, the function can or cannot return one or more values. There are three types of functions in Python:

1. **Built-in-Function:**

Built-in functions, such as help() to ask for help, min() to get the minimum value, print() to print an object to the terminal.

1. **User-Defined Functions:**

User-Defined Functions (UDFs), which are functions that users create to help them out.

1. **Anonymous Functions:**

Anonymous functions, which are also called lambda functions because they are not declared with the standard def keyword.

**Functions vs Methods**

A method refers to a function which is part of a class. You access it with an instance or object of the class. A function doesn’t have this restriction: it just refers to a standalone function. This means that all methods are functions but not all functions are methods.

**Parameters vs Arguments**

Parameters are the names used when defining a function or a method, and into which arguments will be mapped. In other words, arguments are the things which are supplied to any function or method call, while the function or method code refers to the arguments by their parameter names. As shown in the following program where we will make a function f which takes one argument name x.

**Example 1:** Write a function f which takes one argument x, it will square the value of x and add 1 in it then return the answer to user

def f(x):

res = x\*\*2 + 1

return res

**How to define a User-Defined Functions (UDFs):**

User defined functions (UDFs) are normally defined by the keyword def followed by the following format:

***def*** *<function name> (argument1, argument2, <multiple arguments>):*

*body of the function*

*return <value>*

Keep in mind the indentation is very important in programming python code.

**Example 2:** Write a Python function which takes no argument and generate and print a list of first and last 6 elements where the values are cube of numbers between 1 and 30 (both included)

def cubeValues():

lst = list()

for i in range(1,31):

lst.append(i\*\*3)

print(lst[:6])

print(lst[-6:]

CubeValues()

# Student Exercise

1. Write a function that computes the sum of the digits in an integer. Use the following function header

**def** sumDigits(n):

Write a test program that prompts the user to enter an integer and displays the sum of all its digits.

**Code:**

print("===== Task 1=========")

def sumDigits(n):

total = 0

while n!=0:

total += n%10

n = n//10

return total

num = int(input('Enter a number: '))

print(sumDigits(num))

**Output:**



1. Write the following function to display three numbers in increasing order

**def** displaySortedNumbers(num1, num2, num3):

Write a test program that prompts the user to enter three numbers and invokes the function to display them in increasing order

**Code:**

print("===== Task 2=========")

def displaySortedNumbers(num1, num2, num3):

if num1>num2 and num1>num3:

if num2>num3:

print(num3,num2,num1)

else:

print(num2,num3,num1)

elif num2>num1 and num2>num3:

if num1>num3:

print(num3,num1,num2)

else:

print(num1,num3,num2)

else:

if num1>num2:

print(num2,num1,num3)

else:

print(num1,num2,num3)

return ''

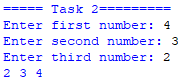
num1 = int(input('Enter first number: '))

num2 = int(input('Enter second number: '))

num3 = int(input('Enter third number: '))

displaySortedNumbers(num1,num2,num3)

**Output:**



1. Write the following function to display an integer in reverse order.

**def** reverse(number):

Write a test program that prompts the user to enter an integer and displays its reversal

**Code:** print("===== Task 3=========")

def reverse(number):

rev = 0

while number!=0:

rev = rev\*10 + number%10

number = number//10

return rev

num = int(input('Enter a number: '))

print(reverse(num))

**Output:**



1. Write the following function which returns True if the number is a palindrome.

**def** isPalindrome(number):

A number is a palindrome if its reversal is the same as itself. Write a test program that prompts the user to enter an integer and reports whether the integer is a palindrome.

Hint: Use the **reverse** function to implement **isPalindrome**

**Code:**

print("===== Task 4=========")

def isPalindrome(number):

rev = 0

temp = number

while number!=0:

rev = rev\*10 + number%10

number = number//10

if temp==rev:

return True

else:

return False

num = int(input('Enter a number: '))

print(isPalindrome(num))

**Output:**



1. Write a function that returns the number of days in a year using the following header

**def** numberOfDaysInAYear(year):

Write a test program that displays the number of days in the year from 2010 to 2020

**Code:**

print("===== Task 5=========")

def numberOfDaysInAYear(year):

if year%4==0 and year%100!=0 or year%400==0:

return 366

else:

return 365

years=0

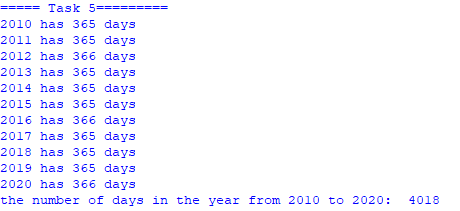
for i in range(2010,2021):

print(f'{i} has {numberOfDaysInAYear(i)} days')

years = years + int(numberOfDaysInAYear(i))

print('the number of days in the year from 2010 to 2020: ',years)

**Output:**



1. Write a function that converts milliseconds to hours, minutes, and seconds using the following header

**def convertMilli(millis):**

The function returns three strings each for hours, minutes, and seconds. For example, convertMillis(5500) returns the strings ‘0’, ‘0’, and ‘5’, convertMillis(100000) returns the strings ‘0’, ‘1, ’40’, and convertMillis(555550000) returns the strings ‘154, ’19’, and ’10’

Write a test program that prompts the user to enter a value for milliseconds and displays a string in the format of hours:minute:seconds

**Code:**

def convertMilli(millis):

sec = millis//1000

min = sec//60

hour = min//60

sec = sec%60

min = min%60

return hour,min,sec

millis = int(input('Enter milliseconds: '))

hour,min,sec = convertMilli(millis)

print(f'{hour}:{min}:{sec}')

**Output:**

