

# Python Programming Lab Manual (U23CM3L1)

## Program 1

1. Write a Python Script to demonstrate

- + Variable
- + Executing Python from the Command Line
- + Editing Python Files
- + Reserved Words

b) Write a python program to add two numbers.

c) Write a program to demonstrate different number data types in python.

d) Write a program to perform different arithmetic operations on numbers in python.

### Source Code - Variable

```
x = 5  
  
y = "John"  
  
z = 10.5  
  
print(x)  
  
print(y)  
  
print(z)
```

### **Output**

```
5  
  
John  
  
10.5
```

### Source Code - Executing Python from the Command Line

Step 1 Go to Start Type Python Command Line

Step 2 Write print(" Hello World!")

Step 3 Enter

### **Output**

```
Hello World!
```

### Source Code – Editing Python Files

```
f= open("guru99.txt","w+")
```

# Python Programming Lab Manual (U23CM3L1)

```
for i in range(10):  
    f.write("This is line %d\r\n" % (i+1))  
f.close()
```

## Output

This is line 1  
This is line 2  
This is line 3  
This is line 4  
This is line 5  
This is line 6  
This is line 7  
This is line 8  
This is line 9  
This is line 10

## Source Code – Reserved Keywords

```
help("keywords")
```

Here is a list of the Python keywords. Enter any keyword to get more help.

False	break	for	not
None	class	from	or
True	continue	global	pass
__peg_parser__	def	if	raise
and	del	import	return
as	elif	in	try

## b) Write a python program to add two numbers.

#Use the + operator to add two numbers:

# Python Programming Lab Manual (U23CM3L1)

```
x = input("Type a number: ")
y = input("Type another number: ")
```

```
sum = int(x) + int(y)
```

```
print("The sum is: ", sum)
```

## Output

Type a number: 12

Type another number: 3

The sum is: 15

---

c) Write a program to demonstrate different number data types in python.

## #Text Type

#String

```
x = "Hello World"
```

```
#display x:
```

```
print(x)
```

```
#display the data type of x:
```

```
print(type(x))
```

## #Numeric Types

#Int

```
x = 20
```

```
#display x:
```

```
print(x)
```

```
#display the data type of x:
```

```
print(type(x))
```

#Float

```
x = 20.5
```

```
#display x:
```

# Python Programming Lab Manual (U23CM3L1)

```
print(x)
```

```
#display the data type of x:
```

```
print(type(x))
```

```
#complex
```

```
x = 1j
```

```
#display x:
```

```
print(x)
```

```
#display the data type of x:
```

```
print(type(x))
```

## Output

```
Hello World
```

```
<class 'str'>
```

```
20
```

```
<class 'int'>
```

```
20.5
```

```
<class 'float'>
```

```
1j
```

```
<class 'complex'>
```

### d) Write a program to perform different arithmetic operations on numbers in python.

Python provides numerous [built-in functions](#) that are readily available to us at the Python prompt.

Some of the functions like `input()` and `print()` are widely used for standard input and output operations respectively

# Python Programming Lab Manual (U23CM3L1)

## Source Code – To perform Arithmetic Operations

```
x = int(input('Enter a number x: '))
```

```
y = int(input('Enter a number y: '))
```

```
print(x + y)
```

```
print(x - y)
```

```
print(x * y)
```

```
print(x / y)
```

### **Output**

Enter a number x: 10

Enter a number y: 5

15

5

50

2.0

# Python Programming Lab Manual (U23CM3L1)

## Program 2

- a) Write a python program to print a number is positive/negative using if-else.
- b) Write a python program to find largest number among three numbers.
- c) Write a Python program to swap two variables
- d) Python Program to print all Prime Numbers in an Interval

### a) Source Code: Python Program to Check if a Number is Positive, Negative or 0 Using if...elif...else

```
num = float(input("Enter a number: "))
```

```
if num > 0:
```

```
    print("Positive number")
```

```
elif num == 0:
```

```
    print("Zero")
```

```
else:
```

```
    print("Negative number")
```

### Output

Enter a number: 4

Positive number

Enter a number: 0

Zero

Enter a number: -6

Negative number

# Python Programming Lab Manual (U23CM3L1)

*b) Source Code: Write a python program to find largest of three numbers using the logical operator*

```
num1 = int(input("Enter first number: "))  
num2 = int(input("Enter second number: "))  
num3 = int(input("Enter third number: "))  
if (num1 > num2) and (num1 > num3):  
    largest = num1  
elif (num2 > num1) and (num2 > num3):  
    largest = num2  
else:  
    largest = num3  
print("The largest number is",largest)
```

## **Output**

Enter first number: 10

Enter second number: 5

Enter third number: 2

The largest number is 10

# Python Programming Lab Manual (U23CM3L1)

## c) Source Code - Python program to swap two variables

**# To take inputs from the user**

```
x = input('Enter value of x: ')
```

```
y = input('Enter value of y: ')
```

**# create a temporary variable and swap the values**

```
temp = x
```

```
x = y
```

```
y = temp
```

```
print('The value of x after swapping: {}'.format(x))
```

```
print('The value of y after swapping: {}'.format(y))
```

### **Output**

Enter value of x: 5

Enter value of y: 10

The value of x after swapping: 10

The value of y after swapping: 5



# Python Programming Lab Manual (U23CM3L1)

## d) Source Code: Python program to display all the prime numbers in an interval

```
#lower = int(input ("Please, Enter the Lowest Range Value: "))
#upper = int(input ("Please, Enter the Upper Range Value: "))

lower = 2
upper = 15

print("Prime numbers between", lower, "and", upper, "are:")

for num in range(lower, upper + 1):
    # all prime numbers are greater than 1
    if num > 1:
        for i in range(2, num):
            if (num % i) == 0:
                break
        else:
            print(num)
```

### **Output**

Prime numbers between 2 and 15 are:

```
2
3
5
7
11
13
```

# Python Programming Lab Manual (U23CM3L1)

## Program 3

- Write a python program to check whether the given string is palindrome or not.
- Write a program to create, concatenate and print a string and accessing substring from a given string.
- Functions: Passing parameters to a Function, Variable Number of Arguments, Scope, and Passing Functions to a Function**

a) Source Code - python program to check whether the given string is palindrome or not.

```
def isPalindrome(s):  
    return s == s[::-1]
```

```
# Driver code  
s = "malayalam"  
ans = isPalindrome(s)
```

```
if ans:  
    print("Yes")  
else:  
    print("No")
```

**Output**  
Yes

### What is a Palindrome?

A palindrome is nothing but any number or a string which remains unaltered when reversed.

**Example:** 12321

**Output:** Yes, a Palindrome number

**Example:** RACECAR

**Output:** Yes, a Palindrome string

# Python Programming Lab Manual (U23CM3L1)

*b) Source code - Write a program to create, concatenate and print a string and accessing substring from a given string.*

**# Python program to show string concatenation**

# Defining strings

str1 = "Welcome to"

str2 = "Python Programming Lab ☺ "

# + Operator is used to strings concatenation

str3 = str1 + str2

print("The new concatenated string is " + str3) # Printing the new combined string

str4 = "abc"

# printing original string

print("The original string is : " + str4)

# Get all substrings of string

# Using list comprehension + string slicing

```
res = [str4[i: j] for i in range(len(str4))  
      for j in range(i + 1, len(str4) + 1)]
```

# printing result

print("All substrings of string are : " + str(res))

## Output

The new concatenated string is Welcome to Python Programming Lab ☺

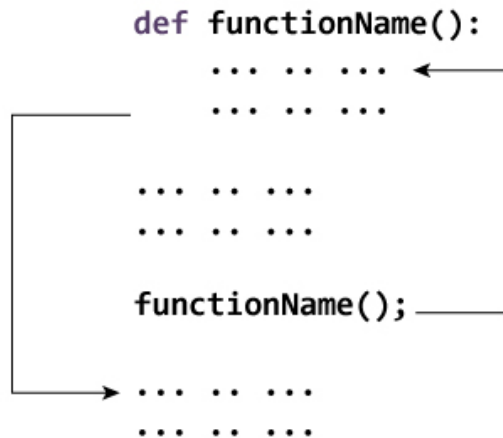
The original string is : abc

All substrings of string are : ['a', 'ab', 'abc', 'b', 'bc', 'c']

# Python Programming Lab Manual (U23CM3L1)

## c) Functions: Passing parameters to a Function, Variable Number of Arguments, Scope, and Passing Functions to a Function

### Explanation – Python Functions



A function is a block of code which only runs when it is called. We can pass data, known as parameters, into a function. A function can return data as a result.

#### **Syntax of Function**

```
def function_name(parameters):  
    """docstring"""  
    statement(s)
```

### Explanation – a. Passing parameters to a Function

Parameters or Arguments?

The terms *parameter* and *argument* can be used for the same thing: information that are passed into a function.

### Source Code: Write a python program to demonstrate how to pass parameters to a function

```
# Function definition is here  
def printme( str):  
    # "This prints a passed string into this function"  
    print(str)  
    return;  
# Now you can call printme function  
printme("I'm first call to user defined function!")  
printme("Again second call to the same function")
```

#### **Output**

When the above code is executed, it produces the following result –

I'm first call to user defined function!

Again second call to the same function

# Python Programming Lab Manual (U23CM3L1)

## Explanation – b. Variable Number of Arguments

### Arguments

Information can be passed into functions as arguments.

### Source Code: Write a python program to demonstrate arguments in function

```
print("Argument Example")
defmy_function(fname):
    print(fname + " khan")
my_function("Saba")
my_function("Salman")
my_function("Zohran")
```

### Output

```
Argument Example
Saba khan
Salman khan
Zohran khan
```

---

## Explanation – c. Scope

### Python Scope

A variable is only available from inside the region it is created. This is called scope.

### Source Code –A program to illustrate the scope of a variable inside a function.

```
# This function has a variable with
# name same as s.
def f():
    # local scope
    s = "Me too."
    print(s)
```

```
# Global scope
s = "I love Python"
f()
print(s)
```

```
def my_func():
    # local scope
    x = 10
    print("Value inside function:", x)
```

```
# Global scope
```

# Python Programming Lab Manual (U23CM3L1)

```
x = 20
my_func()
print("Value outside function:",x)
```

## OUTPUT

```
Me too.
I love Python
Value inside function: 10
Value outside function: 20
```

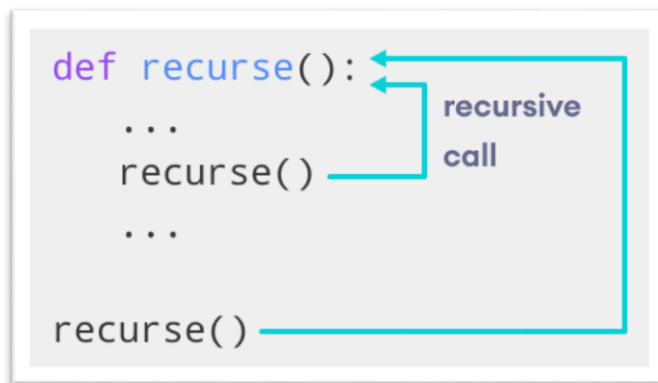
---

### Explanation d. Passing Functions to a Function

#### **Recursion**

Python also accepts function recursion, which means a defined function can call itself. Recursion is a common mathematical and programming concept. It means that a function calls itself. This has the benefit of meaning that you can loop through data to reach a result.

The following image shows the working of a recursive function called recurse.



### Source Code –A program to illustrate the Recursion

```
def factorial(x):
    """This is a recursive function
    to find the factorial of an integer"""

    if x == 1:
        return 1
    else:
        return (x * factorial(x-1))

num = 3
print("The factorial of", num, "is", factorial(num))
```

#### **Output**

The factorial of 3 is 6

# Python Programming Lab Manual (U23CM3L1)

## Program 4

a) Create a list and perform the following methods

1) insert () 2) remove() 3) append() 4) len() 5) pop() 6)clear()

b) Create a dictionary and apply the following methods

1) Print the dictionary items 2) access items 3) useget () 4) change values 5) use len()

c) Create a tuple and perform the following methods

1) Add items 2) len() 3) check for item in tuple 4)Access items

### Explanation – g. List , Tuples, Sets, Dictionary

#### Python Collections (Arrays)

There are four collection data types in the Python programming language:

**List** is a collection which is ordered and changeable. Allows duplicate members.

**Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.

**Set** is a collection which is unordered and unindexed. No duplicate members.

**Dictionary** is a collection which is unordered, changeable and indexed. No duplicate members.

#### a) Source Code - Write a program to create, append, and remove lists in python.

```
print("Create a list")
thislist = ["apple", "banana", "cherry"]
print(thislist)

print("Add an item at a specified index")
thislist = ["apple", "banana", "cherry"]
thislist.insert(1, "orange")
print(thislist)

print("Get the length of a list")
thislist = ["apple", "banana", "cherry"]
print(len(thislist))

print("Add an item to the end of the a list")
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist)

print("Remove an item")
thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print(thislist)

print("Remove the last item")
thislist = ["apple", "banana", "cherry"]
thislist.pop()
print(thislist)
```

# Python Programming Lab Manual (U23CM3L1)

```
print("Remove an item at a specified index")
thislist = ["apple", "banana", "cherry"]
del thislist[0]
print(thislist)
```

```
print("Empty list")
thislist = ["apple", "banana", "cherry"]
thislist.clear()
print(thislist)
```

## Output

### *Create a list*

```
['apple', 'banana', 'cherry']
```

### *Add an item at a specified index*

```
['apple', 'orange', 'banana', 'cherry']
```

### *Get the length of a list*

```
3
```

### *Add an item to the end of the a list*

```
['apple', 'banana', 'cherry', 'orange']
```

### *Remove an item*

```
['apple', 'cherry']
```

### *Remove the last item*

```
['apple', 'banana']
```

### *Remove an item at a specified index*

```
['banana', 'cherry']
```

### *Empty list*

---



# Python Programming Lab Manual (U23CM3L1)

## b) Source Code - Creating Python Dictionary, accessing items , use get() method , change values & Len()

```
print("1.Create and print a dictionary:")
```

```
thisdict = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964  
}  
print(thisdict)
```

```
print("2. Accessing Items:")
```

```
# Get the value of the "model" key:
```

```
thisdict = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964  
}  
x = thisdict["model"]
```

```
print("3. Using get() Method:")
```

```
#There is also a method called get() that will give you the same result:
```

```
print("Get the value of the model key:")
```

```
x = thisdict.get("model")
```

```
print("4. Change values:")
```

```
#Change the "year" to 2022:
```

```
thisdict = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964  
}  
thisdict["year"] = 2022
```

```
print("5.Print the number of items in the dictionary:")
```

```
print(len(thisdict))
```

```
print("empty dictionary")
```

```
mydict = { }
```

### **Output**

1.Create and print a dictionary:

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

2. Accessing Items:

3. Using get() Method:

Get the value of the model key:

4. Change values:

5.Print the number of items in the dictionary:

3

empty dictionary

# Python Programming Lab Manual (U23CM3L1)

c) Source Code - Write a program to Create a tuple and perform the following methods Add items, check for item in tuple, Access items & use len()

```
print("1. Create a tuple")
thistuple = ("apple", "banana", "cherry")
print(thistuple)

print("2.Add Items")
#Convert the tuple into a list, add 'orange', and convert it back into a tuple:
thistuple = ("apple", "banana", "cherry")
y = list(thistuple)
y.append("orange")
thistuple = tuple(y)
print(thistuple)
print("3.Check if a tuple item exists")
thistuple = ("apple", "banana", "cherry")
if "apple" in thistuple:
    print("Yes, 'apple' is in the fruits tuple")

print("4.Access tuple items")
thistuple = ("apple", "banana", "cherry")
print(thistuple[1])

print("Loop through a tuple")
thistuple = ("apple", "banana", "cherry")
for x in thistuple:
    print(x)
print("5.Get the length of a tuple ")
thistuple = ("apple", "banana", "cherry")
print(len(thistuple))
```

## Output

### 1. Create a tuple

('apple', 'banana', 'cherry')

### 2.Add Items

('apple', 'banana', 'cherry', 'orange')

### 3.Check if a tuple item exists

Yes, 'apple' is in the fruits tuple

### 4.Access tuple items

banana

### Loop through a tuple

# Python Programming Lab Manual (U23CM3L1)

apple

banana

cherry

***5.Get the length***

3

# Python Programming Lab Manual (U23CM3L1)

## Program 5

### a. Write a Python Script to demonstrate OOP Concepts

- + Classes
- + File Organization
- + Special Methods
- + Inheritance
- + Polymorphism
- + Special Characters
- + Regular Expressions

### b. Write a python Program to call data member and function using classes and objects

#### Explanation – Classes

#### *Python Classes and Objects*

Python Classes/Objects

Python is an object oriented programming language. Almost everything in Python is an object, with its properties and methods.

A Class is like an object constructor, or a "blueprint" for creating objects.

---

#### Source Code – Write a python program to demonstrate class

#Create a class named Person, use the \_\_init\_\_() function to assign values for name and age:

#Note: The \_\_init\_\_() function is called automatically every time the class is being used to create a new object.

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
```

```
p1 = Person("John", 36)
```

```
print(p1.name)
print(p1.age)
```

#### **OUTPUT**

```
John
36
```

---

# Python Programming Lab Manual (U23CM3L1)

## Explanation – File Organization

### **Python File Open**

File handling is an important part of any web application.

Python has several functions for creating, reading, updating, and deleting files.

---

#### **File Handling**

The key function for working with files in Python is the `open()` function.

The `open()` function takes two parameters; *filename*, and *mode*.

There are four different methods (modes) for opening a file:

"r" - Read - Default value. Opens a file for reading, error if the file does not exist

"a" - Append - Opens a file for appending, creates the file if it does not exist

"w" - Write - Opens a file for writing, creates the file if it does not exist

"x" - Create - Creates the specified file, returns an error if the file exists

In addition, you can specify if the file should be handled as binary or text mode

"t" - Text - Default value. Text mode

"b" - Binary - Binary mode (e.g. images)

---

*All methods of file object are given below:*

`file.close()` Closes the file.

<code>next(file)</code>	Returns the next line from the file each time it is called.
<code>file.read([size])</code>	Reads at a specified number of bytes from the file.
<code>file.readline()</code>	Reads one entire line from the file.
<code>file.readlines()</code>	Reads until EOF and returns a list containing the lines.
<code>file.seek(offset, from)</code>	Sets the file's current position.
<code>file.tell()</code>	Returns the file's current position
<code>file.write(str)</code>	Writes a string to the file. There is no return value.

---

### Source Code – write a python program to demonstrate Python File I/O

#Create a new file

```
f = open("itworkshop.txt", "w")
```

```
f = open("abc.txt", "w")
```

#Write to an Existing File

#Open the file "demofile2.txt" and append content to the file:

# Python Programming Lab Manual (U23CM3L1)

```
f = open("itworkshop.txt", "a")
f.write("Now the file has more content!")
f.close()
```

#open and read the file after the appending:

```
f = open("itworkshop.txt", "r")
print(f.read())
```

# Close file

```
f = open("itworkshop.txt", "r")
print(f.readline())
f.close()
```

#Check if file exists, then delete it:

```
import os
if os.path.exists("abcd.txt"):
    os.remove("abc.txt")
else:
    print("The file does not exist")
```

#Delete file

```
import os
os.remove("abc.txt")
```

## OUTPUT

Now the file has more content!

Now the file has more content!

The file does not exist

---

## Explanation – Special Methods

### **Python - Magic Methods**

Magic methods in Python are the special methods which add "magic" to your class. Magic methods are not meant to be invoked directly by you, but the invocation happens internally from the class on a certain action. For example, when you add two numbers using the + operator, internally, the `__add__()` method will be called.

Built-in classes in Python define many magic methods. Use the `dir()` function to see the number of magic methods inherited by a class.

### Source Code – To illustrate Magic methods in python

```
dir(int)
```

## OUTPUT

```
['__abs__', '__add__', '__and__', '__bool__', '__ceil__', '__class__']
```

### Source Code - # Python program to demonstrate new (), init (). methods

# don't forget the object specified as base

```
class A(object):
    def __new__(cls):
```

# Python Programming Lab Manual (U23CM3L1)

```
print("Creating instance")
return super(A, cls).__new__(cls)
```

```
def __init__(self):
    print("Init is called")
```

A()

## OUTPUT

Creating instance  
Init is called

### Explanation – Python Inheritance

#### **Python Inheritance**

Inheritance allows us to define a class that inherits all the methods and properties from another class.

**Parent class** is the class being inherited from, also called base class.

**Child class** is the class that inherits from another class, also called derived class.

---

#### **Use the super() Function**

Python also has a **super()** function that will make the child class inherit all the methods and properties from its parent:

---

### Source Code – Write a python program to demonstrate the concept of Inheritance in python.

```
#Inheritance Concept using the Quadrilateral
class quadriLateral:
    def __init__(self, a, b, c, d):
        self.side1=a
        self.side2=b
        self.side3=c
        self.side4=d

    def perimeter(self):
        p=self.side1 + self.side2 + self.side3 + self.side4
        print("perimeter=",p)

q1=quadriLateral(7,5,6,4)
q1.perimeter()

class rectangle(quadriLateral):
    def __init__(self, a, b):
        super().__init__(a, b, a, b)
r1=rectangle(10, 20)
r1.perimeter()
```

## OUTPUT:

# Python Programming Lab Manual (U23CM3L1)

```
perimeter= 22  
perimeter= 60
```

---

## Explanation – Python Polymorphism

Polymorphism means the ability to take various forms. In Python, Polymorphism allows us to define methods in the child class with the same name as defined in their parent class. As we know, a child class inherits all the methods from the parent class. However, you will encounter situations where the method inherited from the parent class doesn't quite fit into the child class. In such cases, we will have to re-implement method in the child class. This process is known as Method Overriding.

## Source Code – Write a python program to demonstrate polymorphism using method overriding

#The following program demonstrates method overriding in action:

```
class A:  
    def explore(self):  
        print("explore() method from class A")  
  
class B(A):  
    def explore(self):  
        print("explore() method from class B")  
  
b_obj =B()  
a_obj =A()  
b_obj.explore()  
a_obj.explore()
```

## OUTPUT

```
explore() method from class B  
explore() method from class A
```

---

## Explanation – Python PythonRegEx

Regular expressions (RegEx), and use Python's re module to work with RegEx (with the help of examples).

A **Regular Expression** (RegEx) is a sequence of characters that defines a search pattern. For example,

`^a...s$`

The above code defines a RegEx pattern. The pattern is: **any five letter string starting with a and ending with s.**

A pattern defined using RegEx can be used to match against a string.

Expression	String	Matched?
<code>^a...s\$</code>	abs	No match
	alias	Match



# Python Programming Lab Manual (U23CM3L1)

	abyss	Match
	Alias	No match
	An abacus	No match

**Source Code - Python has a module named `re` to work with RegEx. Here's an example:**

```
import re

pattern = '^a...s$'
test_string = 'abyss'
result = re.match(pattern, test_string)

if result:
    print("Search successful.")
else:
    print("Search unsuccessful.")
```

## OUTPUT

Search successful.

Here, we used `re.match()` function to search `pattern` within the `test_string`. The method returns a match object if the search is successful. If not, it returns `None`.

## Python RegEx

Python has a module named `re` to work with regular expressions. To use it, we need to import the module.

```
import re
```

The module defines several functions and constants to work with RegEx.

## Character class

A "character class", or a "character set", is a set of characters put in square brackets. The regex engine matches only one out of several characters in the character class or character set. We place the characters we want to match between square brackets. If we want to match any vowel, we use the character set `[aeiou]`.

**Source Code - Program to match vowels using RegEx. (`re.findall()`)**

```
import re
s = 'mother of all battles'
result = re.findall(r'[aeiou]', s)
print result
```

Output

```
['o', 'e', 'o', 'a', 'a', 'e']
```

|

## Quantifiers in Python

A quantifier has the form `{m,n}` where `m` and `n` are the minimum and maximum times the expression to which the quantifier applies must match. We can use quantifiers to specify the number of occurrences to match.

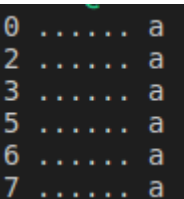
# Python Programming Lab Manual (U23CM3L1)

<b>a</b>	:	Exactly one 'a'
<b>a+</b>	:	At least one 'a'
<b>a*</b>	:	Any number of a's including zero number
<b>a?</b>	:	At most one 'a' i.e either zero number or one number
<b>a{m}</b>	:	Exactly m number of a's
<b>a{m,n}</b>	:	Minimum m number of a's and Maximum n number of a's

## Source Code - # Program to demonstrate Quantifiers in python

```
import re
matcher=re.finditer("a","abaabaaab")
for match in matcher:
    print(match.start(),".....",match.group())
```

### Output



```
0 ..... a
2 ..... a
3 ..... a
5 ..... a
6 ..... a
7 ..... a
```

Dot (.) metacharacter

The dot (.) metacharacter stands for any single character in the text.

## Source Code - # Program to demonstrate dot metacharacter in python

```
import re

words = ('seven', 'even', 'prevent', 'revenge', 'maven',
        'eleven', 'amen', 'event')

pattern = re.compile(r'.even')

for word in words:
    if re.match(pattern, word):
        print(f"The {word} matches")
```

### Output

The seven matches  
The revenge matches

## re.findall()

The `re.findall()` method returns a list of strings containing all matches.

## Source Code - # Program to extract numbers from a string using RegEx. (re.findall())

*#Example 1: re.findall()*

# Python Programming Lab Manual (U23CM3L1)

# Program to extract numbers from a string

```
import re
```

```
string = 'hello 12 hi 89. Howdy 34'
```

```
pattern = '\d+'
```

```
result = re.findall(pattern, string)
```

```
print(result)
```

## OUTPUT

```
['12', '89', '34']
```

If the pattern is not found, `re.findall()` returns an empty list.

---

## *re.split()*

The `re.split()` method splits the string where there is a match and returns a list of strings where the splits have occurred.

---

### **Source Code - # Program to split a string using RegEx. (*re.split()*)**

#### **#Example 2: *re.split()***

```
import re
```

```
string = 'Twelve:12 Eighty nine:89.'
```

```
pattern = '\d+'
```

```
result = re.split(pattern, string)
```

```
print(result)
```

```
# Output: ['Twelve:', ' Eighty nine:', '.']
```

## OUTPUT

```
['Twelve:', ' Eighty nine:', '.']
```

If the pattern is not found, `re.split()` returns a list containing the original string.

---

### **Source Code - # Program to Program to remove all whitespaces using RegEx. (*re.sub()*)**

#### **#Example 3: *re.sub()***

#### **# Program to remove all whitespaces**

```
import re
```

```
# multiline string
```

```
string = 'abc 12\'
```

# Python Programming Lab Manual (U23CM3L1)

```
de 23 \n f45 6'
```

```
# matches all whitespace characters
pattern = '\s+'
```

```
# empty string
replace = ''
```

```
new_string = re.sub(pattern, replace, string)
print(new_string)
```

```
# Output: abc12de23f456
```

## OUTPUT

```
abc12de23f456
```

If the pattern is not found, `re.sub()` returns the original string.

---

## *re.search()*

The `re.search()` method takes two arguments: a pattern and a string. The method looks for the first location where the RegEx pattern produces a match with the string.

If the search is successful, `re.search()` returns a match object; if not, it returns `None`.

```
match = re.search(pattern, str)
```

---

## **Source Code - # Program to Program to demonstrate re.search()**

### *#Example 5: re.search()*

```
import re
string = "Python is fun"
# check if 'Python' is at the beginning
match = re.search('^APython', string)
```

```
if match:
    print("pattern found inside the string")
else:
    print("pattern not found")
```

```
# Output: pattern found inside the string
```

## OUTPUT

```
pattern found inside the string
```

---

## **b) Write a python Program to call data member and function using classes and objects**

#Insert a function that prints a greeting, and execute it on the p1 object:

#The self parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.

# Python Programming Lab Manual (U23CM3L1)

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def myfunc(self):
        print("Hello my name is " + self.name)
```

```
p1 = Person("John", 36)
p1.myfunc()
print(p1.age)
OUTPUT
```

```
Hello my name is John
36
```

# Python Programming Lab Manual (U23CM3L1)

## Program 6

- a) Write a program to double a given number and add two numbers using lambda()
- b) Write a program for filter () to filter only even numbers from a given list.
- c) Write a Python Program to Make a Simple Calculator

### Explanation – Python Lambda

A lambda function is a small anonymous function. A lambda function can take any number of arguments, but can only have one expression.

---

#### **Syntax**

lambda *arguments* :*expression*

The expression is executed and the result is returned:

Example

Add 10 to argument **a**, and return the result:

```
x = lambda a: a + 10
```

```
print(x(5))
```

#### **Output**

15

### a) Source Code – Write a program to show the use of lambda functions

**#Use lambda functions when an anonymous function is required for a short period of time.**

```
double = lambda x: x * 2
```

```
print(double(5))
```

**#Lambda functions can take any number of arguments: Addition argument a with argument b and return the result:**

```
x = lambda a, b : a + b
```

```
print(x(5, 6))
```

#### **OUTPUT**

10

11

# Python Programming Lab Manual (U23CM3L1)

## b) Source Code – Write a program to filter out only the even items from a list using Lamda function

```
my_list = [1, 5, 4, 6, 8, 11, 3, 12]
new_list = list(filter(lambda x: (x%2 == 0) , my_list))
print(new_list)
```

### **Output**

```
[4, 6, 8, 12]
```

## c)Source Code – Write a Program to make a simple calculator using functions

```
# This function adds two numbers
```

```
def add(x, y):
    return x + y
```

```
# This function subtracts two numbers
```

```
def subtract(x, y):
    return x - y
```

```
# This function multiplies two numbers
```

```
def multiply(x, y):
    return x * y
```

```
# This function divides two numbers
```

```
def divide(x, y):
    return x / y
```

```
print("Select operation.")
print("1.Add")
print("2.Subtract")
print("3.Multiply")
print("4.Divide")
```

```
while True:
```

```
    # Take input from the user
```

```
    choice = input("Enter choice(1/2/3/4): ")
```

```
    # Check if choice is one of the four options
```

```
    if choice in ('1', '2', '3', '4'):
```

```
        num1 = float(input("Enter first number: "))
```

```
        num2 = float(input("Enter second number: "))
```

```
    if choice == '1':
```

```
        print(num1, "+", num2, "=", add(num1, num2))
```

```
    elif choice == '2':
```

# Python Programming Lab Manual (U23CM3L1)

```
print(num1, "-", num2, "=", subtract(num1, num2))

elif choice == '3':
    print(num1, "*", num2, "=", multiply(num1, num2))

elif choice == '4':
    print(num1, "/", num2, "=", divide(num1, num2))
    break

else:
    print("Invalid Input")
```

## Output

```
Select operation.
1.Add
2.Subtract
3.Multiply
4.Divide
Enter choice(1/2/3/4): 1
Enter first number: 6
Enter second number: 9
6.0 + 9.0 = 15.0
```



# Python Programming Lab Manual (U23CM3L1)

## Program 7

- a) Demonstrate a python code to print try, except and finally block statements
- b) Write a python program to open and write “hello world” into a file and check the access permissions to that file?
- c) Python program to sort the elements of an array in ascending order and Descending order

### Explanation – Exception Handling

When an error occurs, or exception as we call it, Python will normally stop and generate an

These exceptions can be handled using the **try** statement:

Example

The **try** block will generate an exception, because **x** is not defined:

### Syntax

**try:**

```
print(x)
```

**except:**

```
print("An exception occurred")
```

Since the try block raises an error, the except block will be executed.

Output

An exception occurred

### a) Source Code - Demonstrate a python code to print try, except and finally block statements

#The finally block gets executed no matter if the try block raises any errors or not:

**try:**

```
thislist = ["apple", "banana", "cherry"]
```

```
thislist.remove("orange")
```

**except:**

```
print("Something went wrong")
```

**finally:**

```
print("The 'try except' is finished")
```

```
print(thislist)
```

Output

Something went wrong

The 'try except' is finished

['apple', 'banana', 'cherry']

### b) Source Code - Write a python program to open and write “hello world” into a file and check the access permissions to that file?

# Python Programming Lab Manual (U23CM3L1)

```
#Create a new file
f = open("PPLab.txt", "w")
#Write to an Existing File
#Open the file "PPLab.txt", and append content to the file:
f = open("PPLab.txt", "a")
f.write("Hello World ! the file has more content!")
f.close()

#open and read the file after the appending:
f = open("PPLab.txt", "r")
print(f.read())
f = open("PPLab.txt", "r")
#stores all the data of the file into the variable content
content = f.read(4)
# prints the type of the data stored in the file
print(type(content))
#prints the content of the file
print(content)

# Close file
f = open("PPLab.txt ", "r")
print(f.readline())
f.close()
#x: it creates a new file with the specified name. It causes an error a file exists with the same
name.
f1 = open("PPLab.txt","x")
print(f1)
if f1:
    print("File created successfully")
```

## Output

```
Hello World ! the file has more content!
<class 'str'>
Hell
Hello World ! the file has more content!
Traceback (most recent call last):
  FileExistsError: [Errno 17] File exists: 'PPLab.txt'
```

# Python Programming Lab Manual (U23CM3L1)

## c) Source Code - Python program to sort the elements of an array in ascending order and Descending order

```
#Initialize array
arr = [5, 2, 8, 7, 1]
```

```
temp = 0;
```

```
#Displaying elements of original array
```

```
print("Elements of original array: ")
```

```
for i in range(0, len(arr)):
```

```
    print(arr[i],end = "")
```

```
#Sort the array in ascending order
```

```
for i in range(0, len(arr)):
```

```
    for j in range(i+1, len(arr)):
```

```
        if(arr[i] > arr[j]):
```

```
            temp = arr[i]
```

```
            arr[i] = arr[j]
```

```
            arr[j] = temp
```

```
print()
```

```
#Displaying elements of the array after sorting
```

```
print("Elements of array sorted in ascending order: ")
```

```
for i in range(0, len(arr)):
```

```
    print(arr[i])
```

```
print("_____*****_____")
```

```
#Sort the array in descending order
```

**Original array:**

5	2	8	7	1
---	---	---	---	---

**Array after sorting:**

1	2	5	7	8
---	---	---	---	---

# Python Programming Lab Manual (sem-4)

```
for i in range(0, len(arr)):
    for j in range(i+1, len(arr)):
        if(arr[i] < arr[j]):
            temp = arr[i]
            arr[i] = arr[j]
            arr[j] = temp
```

```
print()
```

```
#Displaying elements of array after sorting
```

```
print("Elements of array sorted in descending order: ")
```

```
for i in range(0, len(arr)):
```

```
    print(arr[i])
```

**Original array:**

5	2	8	7	1
---	---	---	---	---

**Array after sorting:**

8	7	5	2	1
---	---	---	---	---

## Output

Elements of original array:

52871

Elements of array sorted in ascending order:

1

2

5

7

8\*\*\*\*\*

Elements of array sorted in descending order:

8

7

5

2

1

# Python Programming Lab Manual (U23CM3L1)

## Program 8

- a) Write a python program to open a file and check what are the access permissions acquired by that file using os module.
- b) Write a program to perform basic operations on random module.

### Explanation - OS Module Working

#### *Creating the new directory*

The **makedirs()** method is used to create the directories in the current working directory. The syntax to create the new directory is given below.

#### **Syntax**

makedirs(directory name)

#### *The getcwd() method*

This method returns the current working directory.

#### **Syntax**

os.getcwd()

#### *Changing the current working directory*

The chdir() method is used to change the current working directory to a specified directory.

The syntax to use the chdir() method is given below.

#### **Syntax**

chdir("new-directory")

#### *Removing the directory*

Deleting directory

The rmdir() method is used to delete the specified directory.

#### **Syntax**

os.rmdir("directory\_name")

**Note:** You can only remove *empty* folders.

#### *Rename File*

The syntax to use the **rename()** method is given below.

#### **Syntax**

rename(current-name, new-name)

# Python Programming Lab Manual (U23CM3L1)

## ***Remove the specified file***

The os module provides the **remove ()** method which is used to remove the specified file.

### **Syntax**

```
remove(file-name)
```

### **a) Source Code – Demonstrating Python OS module**

```
import os
os.getcwd()

import os
os.chdir('C:/Users/admin/Documents')
print(os.getcwd())
import os
#creating a new directory with the name new
os.mkdir("New Folder")

os.rename("abc.txt","demofile.txt")
print("The above code renamed current abc.txt to demofile.txt")
import os
os.remove("demofile.txt")
print("the file has been removed")
os.rmdir("New Folder")
print("the folder has been deleted")
```

### **Output**

**C:\Users\admin\Documents**

**The above code renamed current abc.txt to demofile.txt**  
the file has been removed")

# Python Programming Lab Manual (U23CM3L1)

## b) Source Code - Write a program to perform basic operations on random module.

### Explanation – Generate Random Numbers within a Defined Range

The random.randrange() function selects an item randomly from the given range defined by the start, the stop, and the step parameters. By default, the start is set to 0. Likewise, the step is set to 1 by default.

#### Source Code

```
import random
num = random.randrange(1, 10)
print( num )
num = random.randrange(1, 10, 2)
print( num )
num = random.randrange(0, 101, 10)
print( num )
```

#### Output:

```
4
9
20
```

### Explanation – Select Random Elements

The random.choice() function selects an item from a non-empty series at random. An IndexError is thrown when the parameter is an empty series.

#### Source Code

```
import random
random_s = random.choice('Random Module') #a string
print( random_s )
random_l = random.choice([23, 54, 765, 23, 45, 45]) #a list
print( random_l )
random_s = random.choice((12, 64, 23, 54, 34)) #a set
print( random_s )
```

#### Output:

```
M
765
54
```

# Python Programming Lab Manual (U23CM3L1)

## **Explanation –Shuffle Elements Randomly**

A general sequence, like integers or floating-point series, can be a group of things like a List / Set. The random module contains methods that we can use to add randomization to the series.

The random.shuffle() function shuffles the entries in a list at random.

### **Code**

```
a_list = [34, 23, 65, 86, 23, 43]
random.shuffle( a_list )
print( a_list )
random.shuffle( a_list )
print( a_list )
```

### **Output**

```
[23, 43, 86, 65, 34, 23]
```


```
[65, 23, 86, 23, 34, 43]
```



# Python Programming Lab Manual (U23CM3L1)

## Program 9

- a) Write a python program to practice some basic library modules

 NumPy

 SciPy

### a) Explanation –NumPy Module

#### Addition

The **add()** function sums the content of two arrays, and return the results in a new array.

#### Subtraction

The **subtract()** function subtracts the values from one array with the values from another array, and return the results in a new array.

#### Multiplication

The **multiply()** function multiplies the values from one array with the values from another array, and return the results in a new array.

#### Division

The **divide()** function divides the values from one array with the values from another array, and return the results in a new array.

#### Power

The **power()** function rises the values from the first array to the power of the values of the second array, and return the results in a new array.

### Source Code – Write python program to demonstrate the basic functionality of NumPy Module

```
print("\n*****Addition*****")
```

```
import numpy as np
```

```
arr1 = np.array([10, 11, 12, 13, 14, 15])
```

```
arr2 = np.array([20, 21, 22, 23, 24, 25])
```

```
newarr = np.add(arr1, arr2)
```

```
print(newarr)
```

```
print("\n*****Subtraction*****")
```

## Python Programming Lab Manual (U23CM3L1)

```
import numpy as np

arr1 = np.array([10, 20, 30, 40, 50, 60])

arr2 = np.array([20, 21, 22, 23, 24, 25])

newarr = np.subtract(arr1, arr2)

print(newarr)

print("\n*****Multiplication*****")
```

```
import numpy as np

arr1 = np.array([10, 20, 30, 40, 50, 60])

arr2 = np.array([20, 21, 22, 23, 24, 25])

newarr = np.multiply(arr1, arr2)

print(newarr)

print("\n*****Division*****")
```

```
import numpy as np

arr1 = np.array([10, 20, 30, 40, 50, 60])

arr2 = np.array([3, 5, 10, 8, 2, 33])

newarr = np.divide(arr1, arr2)

print(newarr)

print("\n*****Power*****")
```

```
import numpy as np

arr1 = np.array([10, 20, 30, 40, 50, 60])
```

# Python Programming Lab Manual (U23CM3L1)

```
arr2 = np.array([3, 5, 6, 8, 2, 33])
```

```
newarr = np.power(arr1, arr2)
```

```
print(newarr)
```

## Output

```
*****Addition*****
```

```
[30 32 34 36 38 40]
```

```
*****Subtraction*****
```

```
[-10 -1  8 17 26 35]
```

```
*****Multiplication*****
```

```
[ 200  420  660  920 1200 1500]
```

```
*****Division*****
```

```
[ 3.33333333  4.      3.      5.     25.     1.81818182]
```

```
*****Power*****
```

```
[    1000    3200000  729000000 6553600000000    2500      0]
```

# Python Programming Lab Manual (U23CM3L1)

## b) Explanation of –SciPy Module

### Constants in SciPy

As SciPy is more focused on scientific implementations, it provides many built-in scientific constants.

These constants can be helpful when you are working with Data Science.

**PI is an example of a scientific constant.**

### Constant Units

A list of all units under the constants module can be seen using the `dir()` function.

Source Code – Write python program to demonstrate the basic functionality of SciPy Module

```
print("*****Constants in SciPy*****")
```

```
from scipy import constants
```

```
print(constants.pi)
```

```
print("*****Constant Units*****")
```

```
from scipy import constants
```

```
print(dir(constants))
```

```
#Time:
```

```
from scipy import constants
```

```
print(constants.minute)    #60.0
```

```
print(constants.hour)      #3600.0
```

```
print(constants.day)       #86400.0
```

```
print(constants.week)      #604800.0
```

```
print(constants.year)      #31536000.0
```

# Python Programming Lab Manual (U23CM3L1)

```
print(constants.Julian_year) #31557600.0
```

## Output

```
*****Constants in SciPy*****
```

```
3.141592653589793
```

```
*****Constant Units*****
```

```
['Avogadro', 'Boltzmann', 'Btu', 'Btu_IT', 'Btu_th', 'ConstantWarning', 'G', 'Julian_year', 'N_A',  
'Planck', 'R', 'Rydberg', 'Stefan_Boltzmann', 'Wien', '__all__', '__builtins__', '__cached__',  
'__doc__',
```

Measurement of Time in seconds:

```
60.0
```

```
3600.0
```

```
86400.0
```


```
604800.0
```

```
31536000.0
```

```
31557600.0
```

# Python Programming Lab Manual (U23CM3L1)

## Program10

 Introduction to basic concept of GUI Programming and Develop desktop based application with python basic Tkinter() Module.

Source Code – Write python program to demonstrate the basic functionality of SciPy

```
# Import Module
from tkinter import *

# create root window
root = Tk()

# root window title and dimension
root.title("Welcome to Python Programming Lab 😊")
# Set geometry(widthxheight)
root.geometry('350x200')

# adding menu bar in root window
# new item in menu bar labelled as 'New'
# adding more items in the menu bar
menu = Menu(root)
item = Menu(menu)
item.add_command(label='New')
menu.add_cascade(label='File', menu=item)
root.config(menu=menu)

# adding a label to the root window
lbl = Label(root, text = "Are you a Student?")
lbl.grid()

# adding Entry Field
txt = Entry(root, width=10)
txt.grid(column =1, row =0)

# function to display user text when
# button is clicked
def clicked():

    res = "You wrote" + txt.get()
    lbl.configure(text = res)

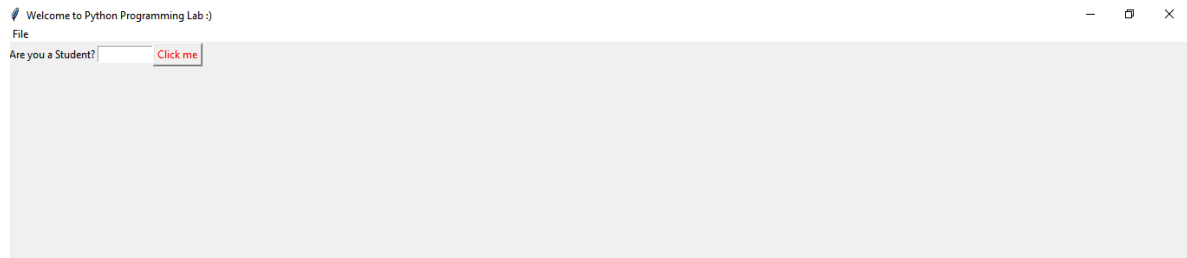
# button widget with red color text inside
btn = Button(root, text = "Click me" ,
             fg = "red", command=clicked)

# Set Button Grid
btn.grid(column=2, row=0)

# Execute Tkinter
root.mainloop()
```


# Python Programming Lab Manual (U23CM3L1)

## Output



# Python Programming Lab Manual (U23CM3L1)

## Program 11

 Write a python program to create a package (college),sub package (all dept),modules(it, cse) and create admin function to module.

**#All department code....**

```
import it_module1 as it
import cse_module2 as cse
```

```
print("From all department")
print(it.it_student1)
```

```
print(cse.cse_student1)
```

**#College Package**

```
import alldept_subpackage as alldept
```

```
print("From Collage")
print(alldept.it.it_student1)
```

```
print(alldept.cse.cse_student1)
```

**# cse department all code goes here**

```
cse_student1={
    "name":"Md Saad",
    "rollno":501,
    "department":"CSE",
    "percent":94.2
}
```

```
cse_student2={
    "name":"Rahul",
    "rollno":502,
    "department":"CSE",
    "percent":83
}
```

**# It department all code goes here**

```
it_student1={
    "name":"Md Saad",
    "rollno":501,
    "department":"IT",
    "percent":90.9
}
```

```
it_student2={
    "name":"Rahul",
    "rollno":502,
```



# Python Programming Lab Manual (U23CM3L1)

```
"department": "IT",  
"percent": 85.5  
}
```

## #Admin Code

```
import sys  
sys.path.insert(0, '../python/collage') #setting packages path  
  
import collage_package as collage  
  
print("From collage")  
  
print(collage.alldept.it.it_student1)  
print(collage.alldept.cse.cse_student1)
```

## Output

```
From all depertment  
{'name': 'Md Saad', 'rollno': 501, 'department': 'IT', 'percent': 90.9}  
{'name': 'Md Saad', 'rollno': 501, 'department': 'CSE', 'percent': 94.2}  
From Collage  
{'name': 'Md Saad', 'rollno': 501, 'department': 'IT', 'percent': 90.9}  
{'name': 'Md Saad', 'rollno': 501, 'department': 'CSE', 'percent': 94.2}  
From collage  
{'name': 'Md Saad', 'rollno': 501, 'department': 'IT', 'percent': 90.9}  
{'name': 'Md Saad', 'rollno': 501, 'department': 'CSE', 'percent': 94.2}
```

# Python Programming Lab Manual (U23CM3L1)

## Program 12

✚ Write a python program to create a package (Engg), sub package (years), modules (sem) and create staff and student function to module.

**#Sem Code goes here**

```
sem1_techers = {
    "techer1" : {
        "name" : "Moksud Alam Mallik",
        "year" : 2011
    },
    "techer2" : {
        "name" : "Kalimulla Alam",
        "year" : 2017
    },
    "techer3" : {
        "name" : "Md nur ",
        "year" : 2015
    }
}

sem1_student = {
    "student1" : {
        "name" : "Nijamuddin Ahammed",
        "year" : 2022
    },
    "student2" : {
        "name" : "Abdul Rahman",
        "year" : 2022
    },
    "student3" : {
        "name" : "Minhaj",
        "year" : 2022
    }
}
```

**#Year Code goes here**

```
import sem

print("All Sem")

print(sem.sem1_techers)
print(sem.sem1_student)
```

# Python Programming Lab Manual (U23CM3L1)

**#Engineering code goes here**

```
import sys
sys.path.insert(0, './engineering') #setting packages path
#import engineering.years as engg

import years

print("From Enginggring")
print(years.sem.sem1_techers)
print(years.sem.sem1_student)
```

## Output

All Sem

```
{'teacher1': {'name': 'Moksud Alam Mallik', 'year': 2011}, 'teacher2': {'name': 'Kalimulla Alam', 'year': 2017}, 'teacher3': {'name': 'Md nur ', 'year': 2015}}
{'student1': {'name': 'Nijamuddin Ahammed', 'year': 2022}, 'student2': {'name': 'Abdul Rahman', 'year': 2022}, 'student3': {'name': 'Minhaj', 'year': 2022}}
```

From Enginggring

```
{'teacher1': {'name': 'Moksud Alam Mallik', 'year': 2011}, 'teacher2': {'name': 'Kalimulla Alam', 'year': 2017}, 'teacher3': {'name': 'Md nur ', 'year': 2015}}
{'student1': {'name': 'Nijamuddin Ahammed', 'year': 2022}, 'student2': {'name': 'Abdul Rahman', 'year': 2022}, 'student3': {'name': 'Minhaj', 'year': 2022}}
```

# Python Programming Lab Manual (U23CM3L1)