

Assignment - 1

Q.1) What is Python?

→ Python is a high-level, general-purpose, and interpreted programming language used in various sectors including machine learning, artificial intelligence, data analysis, web development, and many more. Python is known for its ease of use, powerful standard library, and dynamic semantics.

Q.2) List what we can do in Python Programming

- i) System Programming.
- ii) GUI Programming.
- iii) Gaming.
- iv) Embed with HTML, XML.
- v) Component Integration.
- vi) Database Management.

Q.3) Write a basic syntax of Python Programming language

```
→ a = b
  b = 7
  if (a > b):
    print("a is greater than b");
  else:
    print("b is greater than a");
```

1. Introduction

Q.4) What are the numeric datatypes of Python?

→ There are three numeric datatypes of Python -

- i) Integers - It contains positive or negative whole numbers. Integers are analogous to natural numbers.
- ii) Float - It is a real number with a floating point representation. It is specified by decimal point. eg. (5.1).
- iii) Complex no. - It is specified as real part + imaginary part. eg. (2+5j).

Q.5) List the arithmetic operations in Python.

→ + - Addition Operator is used to sum up or add the values.

- - Subtraction Operator is used to subtract second operand value from first value.

* - Multiplication Operator is used to find the product of two values.

/ - Division Operator used to find the quotient.

% - Module Operator used to find remainder.

* Exponentiation Operator used to raise the power of first operand to the power of second.

11 - Floor division operator used to find the floor of the quotient.

Q.6) Explain the Python in details with Features, advantages and disadvantages.

→ Python is a high-level, interpreted, and general purpose dynamic programming language that focuses on code readability. It generally has small programs when compared to Java & C.

It was founded in 1991 by developer Guido van Rossum. Python ranks among the most popular and fastest-growing languages in the world. Python is a powerful, flexible and easy-to-use language. In addition, the python community is very active. It is used in many organizations as it supports multiple programming paradigms. It also performs automatic memory management.

Advantages -

- i) It is Open-Source language.
- ii) It is versatile, easy to learn and code.
- iii) User-friendly structures of data.
- iv) It is high-level language that abstracts low-level details.
- v) It supports both object-oriented and procedural programming.
- vi) It allows real-time code execution and testing.
- vii) It uses interpreted language which allows easy debugging and code development.

Disadvantages -

- It can consume a lot of memory, especially when working with large datasets or complex algorithms.
- It is dynamically typed language, which means that the types of variables can change at runtime.
- The Global Interpreter Lock (GIL) is a mechanism in python that prevents multiple threads from executing python code at once. It is implemented so that only one thread can run at a time.

(Q.7) Explain the basic data types in python programming

language. Explain about string, number, list, tuple, dictionary, set, file, etc.

There are several data types that allow you to represent the different kinds of values -

i) Integers (int) -

These represent positive or negative whole numbers without fractions or decimals. ex. $x = 5$

ii) Floating-Point Numbers (float) -

These are real numbers with a decimal point.

ex. $x = 3.14$

iii) Complex Numbers (complex) -

It is represented as (real part) + (imaginary)j

where j denotes the imaginary unit. ex. $x = 2 + 3j$

iv) Boolean (bool) -

This have only two possible values True or False

v) strings (str) → defining the print function ()

It is used to represent text or sequences of characters. It can be enclosed in single or double quotes. ex: message = "Hello, World!"

Q.8) Program to find area of triangle.

→ Code -

a = 5

b = 6 # defining the second argument mentioning Col. 1

c = 7

s = (a + b + c) / 2

calculate the area of the triangle using formula

area = (s * (s - a) * (s - b) * (s - c)) ** 0.5

print('The area of the triangle is : %.2f' % area)

Output - The area of the triangle is 14.70.

Q.9) Python program for swapping of two variables or numbers.

→ a) using 3rd variable -

x = 5

y = 10

temp = x

x = y

y = temp

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

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

b) without using 3rd variable - ~~3 steps~~ against 6

$x = 5$ in first swapping of swap function

$y = 7$ in second swapping of swap function

print("Before swapping:") $x = 5$ contains value

print("Value of x: ", x, "and y: ", y)

After swap function call of swap

$x, y = y, x$

print("After swapping:")

print("Value of x: ", x, "and y: ", y)

a.10) Program to convert hours to minutes.

→ code -

```
hours = int(input("Enter Hours:"))
minutes = hours * 60
```

print(minutes, "Minutes")

Output - ~~swap function~~ ~~swap function~~ ~~swap function~~

Enter Hours : 7

420 minutes.

Mitte Mai Milla Denge

Page No.

Data

Assignment 2

1. Write a syntax of nested if-else statement in C++.
if condition 1 is true then
add #code to execute if condition 1
if condition 2:
add #code to execute if condition 2
is true then
else:
add #code to execute if condition 2
is false or if condition 1 is false.
 2. Write a syntax of nested if-else statement in C++.
if condition 1 is true then
add #code to execute if condition 1
if condition 2:
add #code to execute if condition 2
is true then
else:
add #code to execute if condition 2
is false or if condition 1 is false.

2. Write syntax for for loop and while loop.

for loop syntax:

```
for item in iterable:  
    # code block to be executed for  
    # each item in the iterable
```

while loop syntax:

```
while condition:  
    # code block to be executed as long  
    # as the condition is true.
```

Mitte Mai Milla Denge

Page No.
Date

3. what do you mean by nesting of loop

A nested loop is a loop inside another loop. Although all kinds of loops can be nested, the most common nested loop involves for loops. These loops are particularly useful when displaying multidimensional data.

4. Define range function in Python programming language.

The range function in Python is a built-in function used to generate a sequence of numbers within a specified range. It is commonly used in loops for iteration.

5. Write a python program to check entered number is even or odd.

```
num = int(input("Enter any number  
to test whether it is odd or even"))  
if (num % 2) == 0:  
    print ("The number is even")  
else:  
    print ("The number is odd")
```

Mitte Mai Milla Denge

Page No.
Date

6. What are different conditional statements in Python programming language with example

In Python, conditional statements are used to execute different blocks of code based on whether a certain condition evaluates to true or false.

The main conditional statements in Python are:

if statement:

$x = 10$

$\text{if } x > 5 \text{ print("x is greater than 5")}$

$\text{print("x is greater than 5")}$

if-else statement:

$x = 3$

$\text{if } x > 5 \text{ print("x is greater than 5")}$

$\text{else: print("x is not greater than 5")}$

if-elif-else statement:

$x = 7$

$\text{if } x > 10 \text{ print("x is greater than 10")}$

$\text{elif } x > 5 \text{ print("x is greater than 5")}$

$\text{print("x is greater than 5 but not greater than 10")}$

Mitte Mai Milla Denge

1. if statement from if loop finding minimum print ("x is not greater than 5")

b) if else statement

Nested if statements:

a) if else conditions and if else if statements: if max of both then
condition print ("x is greater than 5")
but on if else x>10 condition instead of
else condition print ("x is also greater than
than 10")

else: true nothing is

print ("x is not greater than
than 10")

else: true nothing is

print ("x is not greater than 5")

("a nested condition in ") false

8. what do you mean by continue,
break and pass statement in detail
with example.

("Break Statement") false

The break statement in Python is
used to terminate the loop or
statement in which it is present.
After that, the control will pass to
the statements that are present
after the break statement, if
available.

("Continue Statement") false

Example of break statement

False and continue false

("or break statement") false

Mitte Mai Milla Denge

Page No. _____
Date _____

`s = 'geeks for geeks'`
`for letter in s:`
 `print(letter)`

`if letter == 'e' or letter == 's':`
 `break`

`print("out of for loop")`

`continue statement`
continues a loop control
statement just like the break
statement it continues a loop
it forces to execute the next iteration
of the loop while skipping the
portion initialized to continue

Example of continue statement

```
for i in range(1, 11):  
    if i == 6:  
        continue  
    else:  
        print(i, end = " ")
```

~~Pass Statement~~

The "pass statement" in Python is used
when a statement is required
syntactically but you do not want
any command or code to execute
It is like null operation, as nothing
will happen if it is executed.

Mitte Mai Milla Denge

Page No. _____
Date _____

9. Write a Python program to find the percentage of student on the basis of 5 subject marks.

```

# Input marks for 5 subjects
subject 1 = int(input("Enter marks
for subject 1:"))
subject 2 = int(input("Enter marks
for subject 2:"))
subject 3 = int(input("Enter marks
for subject 3:"))
subject 4 = int(input("Enter marks
for subject 4:"))
subject 5 = int(input("Enter marks
for subject 5:"))

# calculate total marks in 5 subjects
total = subject1 + subject2 + subject3 +
+ (int(i) subject4 + subject5

# calculate percentage
percentage = (total / 500) * 100
print("Your percentage is: " + str
percentage + "%")

```

Mitte Mai Milla Denge

10. Write a python program to find factorial of a entered number.

```
num = int(input("Enter a number"))
def fact(num):
    if num == 0:
        return 1
    else:
        for i in range(1, num+1):
            print(i)
            fact(num - 1))
```

Mitte Mai Milla Denge

Page No.

Date

Assignment 3 To Python

1. What is function?

A function is a block of code designed to perform a specific task. It can take inputs, perform operations or calculations, and return an output. Functions help organize code, make it more modular, reusable, and easier to understand.

2. What are keyword arguments?

Keyword arguments, also known as named arguments, are a feature in many programming languages that allow you to specify the name of the arguments explicitly when calling a function.

3. What are Global scope of variables?

The global scope of variables refers to variables that are defined outside of any function, class, or block and are accessible from any part of the code. These variables have a global lifetime, meaning they exist for the duration of the program's execution.

Mitte Mai Milla Denge

Page No. _____
Date _____

4. what is Recursion?

Recursion is a programming technique where a function calls itself in order to solve a problem. Each recursive call should bring the problem closer to a base case, which is a condition that stops the recursion.

5. What is System function in python?

In Python, the 'system' function is provided by the 'os' module and is used to execute a command in the system shell. It allows you to run shell commands from within a Python script.

6. Explain about scope of function and explain Local, Global and Scope function under function in python

In Python, the scope of a function refers to the region of the code where a particular variable is accessible. There are several scopes to be aware of: local, global and non-local.

Mitte Mai Milla Denge

Page No.

Date

1. Local Scope

Variables defined inside a function have a local scope and are only accessible within the function.

Example:

```
def my_function():
    local-var = 10
    print(local-var)
my_function()
```

2. Global Scope

Variables defined outside any function or in the global scope are accessible throughout the module including inside functions.

Example:

```
global-var = 20
def my_function():
    print(global-var)
my_function()
print(global-var)
```

Q. Write a python program to accept two integers and a character from (+, -, *, /, %). write a function which will perform respective operation based on character and display output.

Mitte Mai Milla Denge

Page No.
Date

```
def perform_operation(num1, num2,  
                      operator):  
    if operator == '+':  
        return num1 + num2  
    elif operator == '-':  
        return num1 - num2  
    elif operator == '*':  
        return num1 * num2  
    elif operator == '%':  
        return num1 % num2  
    elif operator == '/':  
        return num1 / num2  
    else:  
        return "Invalid operator"  
  
num1 = int(input("Enter first  
integer:"))  
num2 = int(input("Enter second  
integer:"))  
operator = input("Enter an operator  
([+, -, *, /, %]):")  
  
if operator in "+-*/%":  
    result = perform_operation(num1, num2,  
                               operator)  
    print(f"The result is: {result}")
```

Mitte Mai Milla Denge

Page No.

Date

8. Write a Python function to check whether the entered string is palindrome or not a palindrome.

```
def is_palindrome(s):
    s = s.lower().replace(" ", "")
    return s == s[::-1]
input_string = input("Enter a string:")
if is_palindrome(input_string):
    print("The entered string is a palindrome")
else:
    print("The entered string is not a palindrome")
```

9. Write a Python function to find "the greatest number among three numbers.

```
def find_greatest_number():
    num1 = float(input("Enter first number:"))
    num2 = float(input("Enter second number:"))
    num3 = float(input("Enter third number:"))
    if num1 > num2 and num1 > num3:
        print("Number 1 is greater than 2 and 3")
    elif num2 > num1 and num2 > num3:
        print("Number 2 is greater than 1 and 3")
    else:
        print("Number 3 is greater than 1 and 2")
```

Mitte Mai Milla Denge

Page No. _____
Date _____

```
if (num1 > num2) and (num1 > num3):
    largest = num1
elif (num2 > num1) and (num2 > num3):
    largest = num2
else:
    print("The largest number is:", largest)
```

Q find-greatest-number() aitne
parde () rakhni hao ()-rakhni

10. Python program to display the first 'n' numbers in the Fibonacci sequence using recursion

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

n = int(input("Enter the no of terms:"))
if n<=0:
    print("Please enter the integer")
else:
    print("Fibonacci Sequence:")
    for i in range(n):
        print(fibonacci(i))
```

Mitte Mai Milla Denge

Page No.

Date

Assignment

How do we find a substring in Python programming?

- In python, you can find a substring within a string using the 'find()' method or the 'index()' method.
2. Explain the differences between upper() and isupper() string functions.
1. 'upper()' function:
- This function is used to convert all lowercase letters in a string to uppercase.
 - It returns a new string with all uppercase letters.
2. 'isupper()' function:
- This function checks if all characters in a string are uppercase.
 - It returns a boolean value ('True' or 'False').
3. What is a list? Why do we use a list in python?

Mitte Mai Milla Denge

Page No.

Date

A list in Python is a data structure that can hold a collection of elements used to store multiple items in a single variable.

Lists are ordered, mutable (changeable), and allow duplicates.

elements in lists are enclosed in brackets.

Why do we use a list?

1. Grouping Data

2. Sequential Access

3. Mutability

4. Dynamic Size

5. Versatility

• A list or mapping nothing in given

4. What is a tuple? How is a tuple different than a list?

(1) Grouping - Hard

A tuple is similar to a list in that

it is also a data structure used to

store a collection of elements.

Tuples are immutable and

created using parentheses, whereas

lists are mutable and created

using square brackets.

Tuples are used when you need

to store fixed collection of

elements, while lists are used

for dynamic collections

(1) Grouping - Hard

Mitte Mai Milla Denge

Page No.

Date

5. What is a dictionary in Python?

A dictionary is a data structure that stores key-value pairs.

An dictionary is a collection of key-value pairs. It is very versatile and powerful data type used for mapping keys to corresponding values.

Dictionaries are unordered, mutable, and allow for duplicate values (though not duplicate keys).

6. Write a python program to find if given reverse of entered string.

```
def checkPalindrome():
```

```
    userinput = input("Enter a string: ")
```

```
    reversedinput = userinput
```

```
    for i in range(len(userinput) - 1, -1, -1):
```

```
        reversedinput += userinput[i]
```

```
    if userinput == reversedinput:
```

```
        print("The entered string is a palindrome")
```

```
    else:
```

```
        print("The entered string is not a palindrome")
```

```
checkPalindrome()
```

Mitte Mai Milla Denge

Page No.
Date

7. Write a Python program to convert all the lower case vowels to uppercase in the string accepted from the user.

```
def convert_vowels_to_uppercase():
    input_string = input("Enter a string: ")
    output_string = ""
    for char in input_string:
        if char.lower() in vowels:
            output_string += char.upper()
        else:
            output_string += char
    return output_string
```

(~~def convert_vowels_to_uppercase():
 user_input = input("Enter a string: ")
 result = convert_vowels_to_uppercase
 print("Modified string is", result)~~)

8. Write a Python program to accept marks of 10 students and display them in a list.

```
("Name or roll") student_name = input()
marks = []
for i in range(10):
    mark = float(input("Enter marks of " + student_name + " student " + str(i+1) + ":"))
    marks.append(mark)
```

Mitte Mai Milla Denge

Page No.	
Date	

8. marks = []
for i in range(10):
 marks.append(int(input("Enter marks of student " + str(i+1) + ": ")))
print("Marks of 10 students are: ")
print(marks)

9. Write a Python program to accept a decimal number, create a function to convert this decimal number to binary and display the output.

```
def decimal_to_binary(decimal):  
    return bin(decimal).replace("0b", "")  
  
decimal_number = int(input("Enter a decimal number: "))  
binary_number = decimal_to_binary(decimal_number)  
print("The binary representation of decimal number {} is: {}".format(decimal_number, binary_number))
```

10. Python program to accept a string from user and store it as a list.

```
user_input = input("Enter a string: ")  
string_list = list(user_input)  
print("List from the input string: ", string_list)
```

Mitte Mai Milla Denge

Page No.
Date

Ques. What is exception? Explain with an example.

Ans. An exception is an event that occurs during the execution of a program, disrupting its normal flow. Exceptions are typically errors such as trying to divide by zero or accessing a nonexistent file. For example:

```
try:
    print("cannot divide by zero!")
except ZeroDivisionError:
    print("division by zero")
```

Ques. List the different keywords to handle exception.

Ans. try: used to wrap the code that may or may not raise an exception.

except: used to catch and handle exceptions that occur in try block.

else: executed if no exceptions occur in try block.

finally: executed regardless of whether an exception occurs or not. Typically used for cleanup or finalization actions.

Note: finally is better than raise because raise must be followed by a string, while finally can contain any valid Python code.

Mitte Mai Milla Denge

Page No.
Date

3. How many except statements can a try-except block have?

A try block in Python can have multiple except statements to handle different types of exceptions. There is no fixed limit to the number of except statements you can include as many as needed to catch and handle different exceptions. Separately, this allows for more granular and specific error handling.

4. When will the else part of try-except-else be executed?

The else part of a try-except-else block in Python is executed only if no exceptions are raised in the try block. If an exception occurs, the else block is skipped, and the corresponding except block is executed instead.

5. When is the finally block executed?

The finally block in Python is executed always, regardless of whether an exception was raised or not in the try block. This ensures that any necessary cleanup actions or resources releases are performed.

Mitte Mai Milla Denge

Page No.

Date

6. Why use Exceptions in Python?

Explaining in detail :
Using exceptions in Python is crucial for several reasons, all contributing to robust and maintainable code.

1. Error Handling and Separation of Concerns :

• **Explicit Handling:** Exceptions allow for explicit handling of errors, making it clear where potential problems might occur and how they are addressed.

- **Separation of Concerns -** By isolating error handling code from the main program logic, developers can focus on the core functionality.

2. Robustness and Reliability:

- **Graceful Degradation:** Using exceptions ensures that programs can handle unexpected situations gracefully.

3. Resource Management: The finally block

ensures that resources (like files or network connections) are released properly regardless of whether an error occurred during the program's execution.

Mitte Mai Milla Denge

Page No.

Date

3. Improved debugging

- Stack trace: provided when exception occurs, detailed report of the error, including where it happened & state of program at that time.

4. Code clarity and maintainability

- Readability: code that uses exception for error handling is often cleaner and more readable.

- Maintainability: adding new exception types and handlers is often easier than refactoring extensive conditional error checks.

5. Flow control

- Breaking out of deep logic: exception provide a way to exit from deeply nested structures without having to pass error codes back through multiple layers.

6. Custom Exceptions

- Tailored Error messages: developers can define custom exception types for their specific application needs.

- Enhanced handling: custom exceptions can carry additional information making it easier to handle complex error scenarios in a structured way.

Mitte Mai Milla Denge

Page No.

Date

Explain exception handling in python with example

Catching exceptions in Python is a way to handle errors gracefully during the execution of a program. When an error occurs, instead of the program crashing, you can catch the exception and decide what to do next, such as logging the error, retrying the operation, or providing a user-friendly message.

Example :

```
def divide(a, b):
    try:
        result = a / b
    except ZeroDivisionError:
        print("You can't divide by zero!")
    else:
        print(f"The result is {result}")
    finally:
        print("Execution complete")
```

- Try : This block contains code that may raise an exception.
- Except : This block catches the exception and allows you to handle it.
- Else : This block executes if no exception were raised in the try block.
- Finally : This block contains code that will run no matter what, whether an exception is raised or not.

Mitte Mai Milla Denge

8. Explain regular expression in details in python.

Regular expressions (regex) are sequences of characters that define search patterns. They are used for matching, searching, and manipulating strings based on specific patterns. Python's built-in 're' module provides support for working with regular expressions.

Common Regex Functions in Python:

1. re.match()

- Tries to match a pattern at the beginning of a string.
- Returns a match object if successful.

Example:

```
import re
pattern = r'^Hello'
result = re.match(pattern, "Hello world!")
print(result)
```

2. re.search()

- Searches through the string for a pattern.

Example:

```
import re
pattern = r'world'
result = re.search(pattern, "Hello world!")
```

3. re.findall()

- Returns all non-overlapping matches of pattern in string as a list.

Example:

```
import re
pattern = r'world'
result = re.findall(pattern, "Hello world!")
```

4. re.sub()

- Replaces first occurrence of pattern with replacement.

Mitte Mai Milla Denge

Mitte Mai Milla Denge

Page No.	Date
3. re.findall()	non-overlapping matches of a pattern in the string.
Example:	import re pattern = r'\d+' if result := re.findall(pattern, 'There are 123 apples and 456 oranges') print(result) # ['123', '456']
4. re.finditer()	Returning an iterator yielding match objects for all non overlapping matches.
Example:	import re pattern=re.finditer(pattern, 'There are 123 apples and 456 oranges') for match in results: print(match.group())
5. re.sub()	Replacing all occurrences of a pattern with a replacement string.
Example:	import re pattern = r'apple(s .)' # 'apple' or 'apples' replacement = 'bananas' result = re.sub(pattern, replacement, 'I like apples') print(result)

Mitte Mai Milla Denge

Page No.

Date

9. Write a function that accepts a string and searches it for a valid phone number.

```
import re
```

```
def find-phone-number(input_string):
```

$\text{d}\varepsilon_{1,339} = -1.9 \times 10^{-14} \text{ eV}$

Einige Jahre später, im Jahr 1943, wurde die Siedlung wieder aufgelöst.

matches = phone - pattern.findall

(input&string)

```
phone_numbers = ["{}".join(match)]
```

and grade 1 areas) ratibiform marsh in marshy (poorly-drained) habitats.

return phone_numbers[0]

~~input_string = "contact us at +1-800-123-4567"~~

-1234 or (123) 456-7890

Digitized by srujanika@gmail.com

phone-numbers = find-phone-number

print (phone-numbers) : reading (input-string)

the same old "Innamorato

www.mathematik-testen.de

1998-09-09 10:00:00

(unrest) salary

For more information about the study, please contact Dr. Michael J. Hwang at (310) 794-3000 or via email at mhwang@ucla.edu.

Mitte Mai Milla Denss

Mitte Mai Milla Denge

Mitte Mai Milla Denge

Mitte Mai Milla Denge

Page No. _____
Date _____

Q. Explain advantages and disadvantages of Regular Expression in Python.

Regular expressions (regex) are powerful tools for pattern matching and text processing in Python; but they come with their own set of advantages and disadvantages.

Advantages:

1. Powerful Pattern Matching

- Regex allows for complex pattern matching that can be difficult to achieve with other string handling methods.

2. Concise code

- Regex patterns can replace many lines of code with a single pattern, making the code more concise and often more readable once the patterns are understood.

3. Flexibility

- Regular expressions can be used to match almost any text pattern, making them suitable for a variety of applications.

4. Cross Language compatibility

Mitte Mai Milla Denge

Page No.

Date

• ~~RegEx~~ RegEx are not specific to Python, they are used in many programming languages.

• ~~RegEx~~ Python's `re` module provides a robust set of functions for working with regular expressions, including searching, matching and replacing.

• Disadvantages of using RegEx:

- 1. Complexity: Regular expressions can become complex and hard to read, especially for those who are not familiar with the syntax.

- 2. Performance overhead: RegEx operations can be slow due to computationally expensive, especially for complex patterns or large text inputs.

- 3. Readability: While RegEx can make code concise, it can also make it less readable and difficult to maintain.

- 4. Learning curve: Implementing regular expressions in Python requires a learning curve.

- There is a steep learning curve associated with regular expressions. Understanding and mastering the syntax requires practice and time.
- 5. Limited by Syntax
 - While powerful, regex is not suitable for all parsing tasks. It can struggle with nested structures or patterns that require recursive matching, where more advanced parsing techniques might be necessary.