**Why Python? #shiva625#1**

Python is a popular programming language known for its simplicity, readability, and versatility. Its numerous benefits and features have gained widespread adoption in various fields. Some of the main reasons why Python is widely used are:

1. Web Development
2. Game Development
3. Artificial Intelligence and Machine Learning

### Print Statement in Python

print("Hii! Shiva good Morning!")

**Output: Hii! Shiva good Morning!**

Possible Mistakes:

* Spelling Mistakes: **prnt**(”Hello World!”)
* Uppercase ‘P’: **P**rint(”Hello World!”)
* Missing quotes: print(**Hello World!**)

#### Calculations

addition, subtraction, multiplication, division

Python

Copy

print(2+5)

#Output: 7

print(2-5)

#Output: -3

print(2\*5)

#Output: 10

print(6/3)

#Output: 2.0

​

#### Assigning Variables

Python

Copy

a = "shiva"

b = "Loves"

c = "Python"

print(a+" "+b+" "+c)

#Output: **shiva Loves Pytho**

a = 2

b = 5

print(a+b)

#Output: 7

### Taking Input from User:

I take a one variable “a” from user and store into a variable

a = input("Enter your name: ")

print("Your name is "+a)

Output: Enter your name: Shiva varma

Your name is shiva varma

**Questions to Practice (Day 1):**

1. Print your name.
2. Print the result of adding two numbers.
3. Print the result of subtracting two numbers.
4. Print the result of multiplying two numbers.
5. Print the result of dividing two numbers.

**#Shiva625#2**

#### Variables

Variables it is just like a containers is stroes a values

Python

Copy

a = "shiva"

print(a) #Output: **shiva**

a = "mintu"

print(a) #Outpu mintu

​

#### Data Types

There are mainly four types of datatypes

String (Example ki “shiva” , “My Name is shiva” , “Nen python nerchkunta”) Simple terms la cheppalante Stream of Characters. • Capital Letters ( A – Z ) • Small Letters ( a – z ) • Digits ( 0 – 9 ) • Special Characters (~ ! @ # $ % ^ . ?,) • Space

Integer (Example 1,2,-9,0)

Float (Any number with Decimal points, Example ki -3.90, 4.5)

Boolean (True, False)

Python

Copy

# Integer data type

my\_integer = 42

print(my\_integer) # Output: 42

# Float data type

my\_float = 3.14

print(my\_float) # Output: 3.14

# String data type

my\_string = "Hello, World!"

print(my\_string) # Output: Hello, World!

# Boolean data type

is\_true = True

is\_false = False

print(is\_true) # Output: True

print(is\_false) # Output: False

**Questions to Practice (Day - 2):**

1. Declare two variables a and b, assign integer values to them, and print their sum.
   * **Expected Output:** The sum of a and b.
2. Create a variable name and assign your name to it. Print a greeting message using your name.
   * **Expected Output:** Greeting message with your name, e.g., "Hello, John!"
3. Define a variable pi and assign the value of π (pi) to it. Print the value of pi.
   * **Expected Output:** The value of π (pi), e.g., 3.14159.
4. Define a variable is\_raining and ask the user to input either "True" or "False" (as a string). Convert the input to a boolean and print its type.
   * **Expected Input:** "True" or "False"
   * **Expected Output:** The data type of the converted boolean.
5. Create a string variable sentence containing any sentence of your choice. Ask the user to input a number, convert it to an integer, and print the sentence repeated that number of times.
   * **Expected Input:** A number (e.g., "3")
   * **Expected Output:** The sentence repeated the specified number of times.
6. Given two variables x and y, perform the following operations and print the results:
   * Addition of x and y.
   * Subtraction of y from x.
   * Multiplication of x and y.
   * Division of x by y.
   * x raised to the power of y.
   * The remainder when x is divided by y.
   * The floor division of x by y.
7. Define a variable value and assign any numerical value to it. Ask the user to input a new value. Update the variable value with the new input and print the updated value.

* **Expected Input:** A numerical value (e.g., "42")
* **Expected Output:** The updated value of the variable.

These questions cover a range of scenarios related to variables and data types in Python. You can use input() function to get user input for interactive questions

**#SHIVA625#3**

Order of Instructions

Python is line by line Execution

Python

print(a)

a = "iam shiva varma!"

​

**Output:**

**ERROR! Traceback (most recent call last): File "<string>", line 1, in <module> NameError: name 'a' is not defined**

#### Follow proper syntax

For example, Do not giva a space from beginning write a code

Python

a = 10 + 5

b = 5

print(a/b)

​

**Output:** **ERROR! File "<string>", line 3 print(a/b) IndentationError: unexpected indent**

#### BODMAS

The BODMAS rule stands for:

B - Brackets first

O - Orders (exponents and roots, like square roots) next

DM - Division and Multiplication, from left to right

AS - Addition and Subtraction, from left to right

Python

# Example expression: 10 + 5 \* (2 \*\* 3) - 6 / 2

result = 10 + 5 \* (2 \*\* 3) - 6 / 2

# Step 1: Evaluate the expression within the brackets first

# 2 \*\* 3 = 8

# So, the expression becomes: 10 + 5 \* 8 - 6 / 2

# Step 2: Perform Multiplication

# 5 \* 8 = 40

# So, the expression becomes: 10 + 40 - 6 / 2

# Step 3: Perform division

# 6 / 2 = 3

# So, the expression becomes: 10 + 40 - 3

# Step 4: Perform addition

# 10 + 40 = 50

# So, the final result is: 50 - 3 = 47

print(result) # Output: 47.0

​

Python

print(10 / 2 + 3) # Output: 8.0

print(10 / (2 + 3)) # Output: 2.0

​

#### Questions to practice (Day 3):

What are the expected output of the following expressions?

Python

# 1

result = 10 + 3 \* 2 - 8 / 4

​

Python

# 2

result = 4 \*\* 2 + 5 / 2 \* 3

​

Python

# 3

result = (8 + 4) \* 3 / 2

​

Python

# 4

result = 16 / 4 + 2 \*\* 3 - 6

​

Python

# 5

result = 10 - 3 \* (4 + 2) / 5

#SHIVA625#4

**String Concatenation**

Combining a string is called Concatenation

a = "shiva" + " " + "varma"

print(a)

# Output: shiva varma

**Possible Mistakes**

It is add only strings there is no chance of add numbers

a = "Hi" + 10

print(a)

**Output:**

**ERROR! Traceback (most recent call last): File "<string>", line 1, in <module> TypeError: can only concatenate str (not "int") to str**

**String Repetition**

\*\*\*\*\* operator (it is used to repeat the strings many times)

a = "shiva " \* 5

print(a)

**Output:**

**Shiva shiva shiva shiva shiva**

s = "Arrow"

s = ("\* " \* 3) + s + (" \*" \* 3)

print(s)

**Output:** \*\*\* \* \* Arrow \* \* \*\*\*

**Length of String**

**len()** built-in function

name = input()

length = len(name)

print(length)

**OUT put:** Arrow 5

**Questions to Solve (Day - 4):**

1. Concatenate two strings str1 and str2, and print the result.
   * **Expected Input:** str1 = "Hello", str2 = "World"
   * **Expected Output:** "HelloWorld"
2. Ask the user to enter their name and a greeting. Concatenate the name and greeting to form a personalized message and print it.
   * **Expected Input:** name = "John", greeting = "Hi"
   * **Expected Output:** "Hi John"
3. Create a string word and repeat it 5 times. Print the result.
   * **Expected Input:** word = "Python"
   * **Expected Output:** "PythonPythonPythonPythonPython"
4. Ask the user to enter a word and a number. Repeat the word as many times as the given number and print the result.
   * **Expected Input:** word = "Hello", number = 3
   * **Expected Output:** "HelloHelloHello"
5. Create a string sentence and find its length. Print the length of the sentence.
   * **Expected Input:** sentence = "This is a sample sentence."
   * **Expected Output:** 27
6. Ask the user to input a sentence. Find the length of the sentence, and print the last character of the sentence.
   * **Expected Output:** Length of the sentence and the last character.
7. Create two strings str1 and str2. Find the lengths of both strings and concatenate them. Print the concatenated string.
   * **Expected Input:** str1 = "Hello", str2 = "World"
   * **Expected Output:** "HelloWorld"
8. Ask the user to enter two words, word1 and word2. Concatenate the two words with a space in between and print the result.
   * **Expected Input:** word1 = "Hello", word2 = "Python"
   * **Expected Output:** "Hello Python"
9. Create a string pattern containing "\*" character and repeat it to form a pattern. The pattern should have 5 rows. Print the resulting pattern.
   * **Expected Output:**
   * \*
   * \*\*
   * \*\*\*
   * \*\*\*\*
   * \*\*\*\*\*

Remember to use appropriate string concatenation methods, repeat the strings as required, and utilize the len() function to find the length of strings. String manipulation in Python can be fun and useful in various applications!

**#SHIVA625#5**

**Type Conversion**

Type conversion python it is used to convert the one data type to another data type

Example ki int(), float(), str() etc.

For example, "123” ane string ni integer laki maarchali ante int() vaadtam

num\_string = "123"

print(type(num\_string))

num\_int = int(num\_string)

print(type(num\_int))

**Output:**

**<class 'str'> <class 'int'>**

Paina unna example la int() function **num\_string** ni teeskoni, integer loki maarchi **num\_int** lo store chestondi.

Alane oka float value ni integer loki maarusthe? Kinda chudandi 3.14 ni 3 ga ela convert chesamo int() vaadi.

num\_float = 3.14

num\_int = int(num\_float)

print(num\_int)

print(type(num\_int))

**Output:**

**3 <class 'int'>**

Manam integer and float values in strings ga kuda convert cheyachu, kinda oka sari chuseyyandi. 123 and 3.14 ni string type loki ela convert chestunnamo.

num\_int = 123

print(type(num\_int)) # Output: <class 'int'>

num\_string = str(num\_int)

print(num\_string) # Output: 123

print(type(num\_string)) # Output: <class 'str'>

num\_float = 3.14

print(type(num\_float)) # Output: <class 'float'>

num\_string = str(num\_float)

print(num\_string) # Output: 3.14

print(type(num\_string)) # Output: <class 'str'>

**Questions to practice (Day - 5)**

Question 1: Convert the integer 42 to a string.

Expected Input:

value = 42

Expected Output:

result = "42"

Question 2: Convert the string "123" to an integer.

Expected Input:

value = "123"

Expected Output:

result = 123

Question 3: Convert the float 3.14 to an integer.

Expected Input:

value = 3.14

Expected Output:

result = 3

Question 4: Convert the string "5.5" to a floating-point number.

Expected Input:

value = "5.5"

Expected Output:

result = 5.5

Question 5: Convert the integer 100 to a boolean.

Expected Input:

value = 100

Expected Output:

result = True

Question 6: Convert the boolean True to an integer.

Expected Input:

value = True

Expected Output:

result = 1

Question 7: Convert the string "False" to a boolean.

Expected Input:

value = “False”

Expected Output:

result = True

#**Shiva625#6**

**String Indexing**

String Indexing it is used to extract the individual characters in the string

Index of first character - **0** Index of last character - (**length - 1**)

Example chusey mama neat ga artham avtadi

# Define a string

my\_string = "Hello, world!"

# Access the first character of the string

first\_character = my\_string[0]

# Print the result

print(first\_character)

The output will be H.

We can use the negative index extract the characters

# Define a string

my\_string = "Hello, world!"

# Access the last character of the string

last\_character = my\_string[-1]

# Print the result

print(last\_character)

The output will be !.

s = "SHIVA"

# Accessing individual characters using negative indexing

last\_char = s[-1] # 'R'

second\_last\_char = s[-2] # 'A'

third\_last\_char = s[-3] # 'H'

fourth\_last\_char = s[-4] # 'I'

first\_char = s[-5] # 'N'

print("Last character:", last\_char)

print("Second last character:", second\_last\_char)

print("Third last character:", third\_last\_char)

print("Fourth last character:", fourth\_last\_char)

print("First character:", first\_char)

Output:

Last character: R Second last character: A Third last character: H Fourth last character: I First character: N

**String Slicing**

We can use the slicing extract the particular portion

Example chusey mowa!

# Define a string

my\_string = "Hello, world!"

# Extract the first five characters of the string

substring = my\_string[0:5]

# Print the result

print(substring)

The output will be Hello.

Inkoka example chusey!

# Define a string

my\_string = "Hello, world!"

# Extract a range of characters from the string

substring = my\_string[1:8]

# Print the result

print(substring)

The output will be ello, w.

**Questions to practice (Day 6):**

1. Example: Get the first character of the sentence. Input: "The sun is shining." Output: "T"
2. Example: Get the last character of the sentence. Input: "She sells seashells by the seashore." Output: "."
3. Example: Get the character at index 3. Input: "I love Python!" Output: "o"
4. Example: Get the second last character of the sentence. Input: "Life is beautiful." Output: "l"
5. Example: Get a substring from index 7 to index 14 (exclusive). Input: "Welcome to Python programming." Output: " to Pyt"
6. Example: Get a substring from index -9 to -3. Input: "The future is bright." Output: "s brig"
7. Example: Get the first six characters of the sentence. Input: "Good things take time." Output: "Good t"
8. Example: Reverse the sentence using slicing. Input: "Python is awesome!" Output: "!emosewa si nohtyP"
9. Example: Get the length of the sentence using indexing. Input: "Coding is fun!" Output: 14

#**SHIVA625#7**

**Relational Operators**

It is used compare the two values

There are mainly 6 types

* Equal to (==)
* Not equal to (!=)
* Greater than (>)
* Less than (<)
* Greater than or equal to (>=)
* Less than or equal to (<=)

Example

x = 5

y = 7

print(x == y) # False

print(x != y) # True

print(x > y) # False

print(x < y) # True

print(x >= y) # False

print(x <= y) # True

Note: Double equals sign (==) It is used to comparison, single equals sign (=) Is uses to variable assignment

Guess the out put

a = 5

b = 2-7

print(a == b)

print(a != b)

print(a > b)

print(a < b)

print(a >= b)

print(a <= b)

**Questions to practice (Day 7):**

1. Example: Greater than operator (>) Input: 5 > 3 Output: True
2. Example: Less than operator (<) Input: 10 < 20 Output: True
3. Example: Greater than or equal to operator (>=) Input: 7 >= 7 Output: True
4. Example: Less than or equal to operator (<=) Input: 15 <= 12 Output: False
5. Example: Equal to operator (==) Input: "hello" == "hello" Output: True
6. Example: Not equal to operator (!=) Input: 10 != 20 Output: True
7. Example: Comparing integers and floats Input: 5.0 > 4 Output: True
8. Example: Comparing strings with different cases Input: "Hello" == "hello" Output: False
9. Example: Using relational operators with booleans Input: True == False Output: False
10. Example: Comparing strings Input: "apple" < "banana" Output: True
11. Example: Comparing None with a string Input: None == "Python" Output: False
12. Example: Mixing different types in comparisons Input: 5 > "3" Output: TypeError
13. Example: Using relational operators with negative numbers Input: -10 < -5 Output: True
14. Example: Comparing a string with a number Input: "42" == 42 Output: False
15. Example: Using relational operators with floating-point precision Input: 0.1 + 0.1 + 0.1 == 0.3 Output: False

**#SHIVA625#8**

**Logical Operators**

There are mainly three types of operators. x and y ni operands

1. and: Returns True if both operands are True.
2. or: Returns True if at least one of the operands is True.
3. not: Returns the opposite boolean value of the operand.

Tip: True and True ⇒ True True and False ⇒ False False and True ⇒ False False and False ⇒ False

True or True ⇒ True True or False ⇒ True False or True ⇒ True False or False ⇒ False

and Operator: The and operator returns True if both operands are True. Otherwise, it returns False.

# Example 1: Using 'and' with boolean variables

a = True

b = False

result = a and b

print(result) # Output: False

# Example 2: Using 'and' with expressions

x = 10

y = 5

result = (x > 0) and (y < 10)

print(result) # Output: True

result = (x > 0) and (y > 10)

print(result) # Output: False

or Operator: The or operator returns True if at least one of the operands is True. If both operands are False, it returns False.

# Example 1: Using 'or' with boolean variables

a = True

b = False

result = a or b

print(result) # Output: True

# Example 2: Using 'or' with expressions

x = 10

y = 5

result = (x > 0) or (y < 10)

print(result) # Output: True

result = (x < 0) or (y > 10)

print(result) # Output: False

not Operator: The not operator returns the opposite boolean value of the operand. If the operand is True, it returns False, and vice versa.

# Example 1: Using 'not' with boolean variable

a = True

result = not a

print(result) # Output: False

# Example 2: Using 'not' with an expression

x = 10

y = 5

result = not (x > y)

print(result) # Output: False

result = not (x < y)

print(result) # Output: True

we can use the parenthesis we can perform more complex logical operatotrs

# Complex example using all logical operators

x = 5

y = 10

z = 15

result = (x < y) and (y < z) or (x == z)

print(result) # Output: True

In this example, (x < y) and (y < z) evaluates to True, and then the whole expression becomes True or (x == z), which is also True. Hence, the final output is True.

**Questions to practice (Day 8):**

1. Example: "and" operator with two True conditions Input: (10 > 5) and ("apple" == "apple") Output: True
2. Example: "and" operator with one False condition Input: (3 < 2) and ("banana" == "orange") Output: False
3. Example: "and" operator with one True and one False condition Input: (5 >= 3) and (10 != 10) Output: False
4. Example: "or" operator with two True conditions Input: ("car" == "car") or (7 < 9) Output: True
5. Example: "or" operator with one False condition Input: ("dog" == "cat") or (6 < 10) Output: True
6. Example: "or" operator with both False conditions Input: (2 == 3) or (8 > 15) Output: False
7. Example: "not" operator with True condition Input: not (4 <= 3) Output: True
8. Example: "not" operator with False condition Input: not ("orange" == "orange") Output: False
9. Example: "not" operator with "and" and "or" Input: not ((5 > 3) and ("apple" != "banana")) Output: False
10. Example: "and" and "not" operators combined Input: (10 > 5) and not (3 < 2) Output: True
11. Example: "or" and "not" operators combined Input: ("cat" == "cat") or not (6 > 10) Output: True
12. Example: Using parentheses for grouping expressions Input: ((5 >= 3) and (10 != 10)) or (8 > 15) Output: False
13. Example: Combining multiple "and" operators Input: (2 < 5) and (10 == 10) and ("hello" != "world") Output: True
14. Example: Combining multiple "or" operators Input: (7 < 3) or (5 >= 5) or ("apple" == "apple") Output: True
15. Example: Using "not" operator with an expression Input: not (10 > 5 and "car" != "car") Output: True
16. Example: Using "not" operator with "or" and "and" Input: not (5 > 3 or "dog" == "cat" and 7 < 5) Output: False
17. Example: Combining "and" and "or" operators Input: (5 > 3 and "apple" != "banana") or (8 == 8 or 6 < 10) Output: True
18. Example: Combining "or" and "not" operators Input: ("apple" == "banana" or not (6 > 10)) Output: True
19. Example: Complex combination of "and", "or", and "not" Input: not (2 < 5 and (7 > 3 or "hello" == "world")) Output: False
20. Example: Nested use of "and", "or", and "not" operators Input: (not (5 > 3) and (10 != 10 or "car" == "car")) Output: False

#**SHIVA625#9**

#### Conditional Statements

If

, elif (short for "else if"),

and else.

Syntax conditional statements

Python

if condition1: # Code block executed if condition1 is True elif condition2: # Code block executed if condition1 is False and condition2 is True else: # Code block executed if both condition1 and condition2 are False

​

Detailed explanation chuskunte

Using if statement: The if statement is used to execute a block of code if a specified condition is True.

Python

# Example 1: Using 'if' to check a condition age = 18 if age >= 18: print("You are an adult.")

​

Output:

Plain Text

You are an adult.

​

Using if and else statements: The else statement is used to execute a block of code when the condition specified in the if statement is False.

Python

# Example 2: Using 'if' and 'else' to check a condition age = 15 if age >= 18: print("You are an adult.") else: print("You are a minor.")

​

Output:

Plain Text

You are a minor.

​

Using if, elif, and else statements: The elif statement is used to check additional conditions after the if condition. It is short for "else if".

Python

# Example 3: Using 'if', 'elif', and 'else' to check multiple conditions age = 25 if age < 18: print("You are a minor.") elif age >= 18 and age < 65: print("You are an adult.") else: print("You are a senior citizen.")

​

Output:

Plain Text

You are an adult.

​

Decision teeskodaaniki chaala baaga upayoga padtay mawa conditional statements!

#### Questions to practice (Day 9):

**Question 1:** Write a program that takes a number as input and prints "Even" if it's an even number, and "Odd" if it's an odd number.

**Expected Input:**

Plain Text

Enter a number: 7

​

**Expected Output:**

Plain Text

Odd

​

**Question 2:** Write a program that takes two numbers as input and prints the larger number.

**Expected Input:**

Plain Text

Enter the first number: 15 Enter the second number: 22

​

**Expected Output:**

Plain Text

22

​

**Question 3:** Write a program that takes a character as input and prints "Vowel" if it's a vowel (a, e, i, o, u), and "Consonant" otherwise.

**Expected Input:**

Plain Text

Enter a character: a

​

**Expected Output:**

Plain Text

Vowel

​

**Question 4:** Write a program that takes a year as input and prints "Leap Year" if it's a leap year, and "Not a Leap Year" otherwise.

**Expected Input:**

Plain Text

Enter a year: 2024

​

**Expected Output:**

Plain Text

Leap Year

​

**Question 5:** Write a program that takes a grade as input (A, B, C, D, or F) and prints "Pass" if it's A, B, C, or D, and "Fail" if it's F.

**Expected Input:**

Plain Text

Enter your grade: C

​

**Expected Output:**

Plain Tex

Pass

#**SHIVA625#10**

#### Nested Conditional Statements

It is used to if conditional statement is complex

Examples

Python

# Example 1: Nested 'if' statement x = 10 if x > 0: print("x is positive.") if x % 2 == 0: print("x is even.") else: print("x is odd.") else: print("x is not positive.")

​

Output:

Plain Text

x is positive. x is even.

​

Explanation: In this example, we have a nested if statement inside the outer if statement. The outer if checks whether x is positive or not. If it is positive, it enters the inner if statement to determine if x is even or odd. If x is not positive, it directly executes the else block.

# Example 2: Nested 'if-elif-else' statements score = 85 if score >= 90: print("Grade: A") elif score >= 80: print("Grade: B") if score >= 85: print("Good job!") elif score >= 70: print("Grade: C") else: print("Grade: Below C")

Output:

Plain Text

Grade: B Good job!

​

Explanation: In this example, we have nested if-elif-else statements. The outer if-elif-else checks the score to assign a grade. If the score is greater than or equal to 90, it prints "Grade: A." If the score is between 80 and 89, it enters the inner if statement to check if the score is greater than or equal to 85 and prints "Good job!" along with "Grade: B." If the score is between 70 and 79, it prints "Grade: C." If the score is less than 70, it prints "Grade: Below C."

Note: Nested conditional statements complex situations handled

#### Questions to practice (Day - 10)

Question 1: Check if a given number num is positive, negative, or zero.

Expected Input:

Python

num = 8

​

Expected Output:

Plain Text

Positive

​

Question 2: Determine the type of a given number num: even or odd, and whether it is positive or negative.

Expected Input:

Python

num = -5

​

Expected Output:

Plain Text

Negative Odd

​

Question 3: Classify a given age into four categories: baby, child, teenager, or adult.

Expected Input:

Python

age = 16

​

Expected Output:

Plain Text

Teenager

​

Question 4: Assign a grade to a given percentage, considering the following grade scale: A (90-100), B (80-89), C (70-79), D (60-69), and F (below 60).

Expected Input:

Python

percentage = 78

​

Expected Output:

Plain Text

Grade: C

​

Question 5: Check if a given year is a leap year, and if it is, find the number of days in February for that year.

Expected Input:

Python

year = 2000

​Expected Output:Leap Year, February has 29 day

**#SHIVA625#11**

#### While Loop

In Python, a while loop is used to repeatedly execute a block of code as long as a specified condition is True. The loop continues until the condition becomes False

Basic syntax of a while loop chuskunte:

Python

Copy

while condition: # Code block to be executed while the condition is True

​

Example 1:

Python

Copy

# Example: Printing numbers from 1 to 5 using a while loop num = 1 while num <= 5: print(num) num += 1

​

Output:

Plain Text

Copy

1 2 3 4 5

​

Explanation: In this example, we initialize a variable num with the value 1. The while loop starts executing when the condition num <= 5 is True. It prints the value of num and then increments it by 1 (num += 1). The loop continues executing the code block as long as the condition num <= 5 remains True. When num becomes 6, the condition becomes False, and the loop terminates.

It's important to ensure that the condition in the while loop eventually becomes False; otherwise, the loop will run indefinitely, leading to an infinite loop

Example 2:

# Example: Finding the sum of numbers from 1 to 10 using a while loop num = 1 sum\_of\_numbers = 0 while num <= 10: sum\_of\_numbers += num num += 1 print("Sum of numbers from 1 to 10:", sum\_of\_numbers)

​

Output:

Sum of numbers from 1 to 10: 55

​

Explanation: In this example, we use a while loop to find the sum of numbers from 1 to 10. We initialize a variable num with the value 1 and a variable sum\_of\_numbers with the initial value 0. The while loop starts executing when the condition num <= 10 is True. Inside the loop, we add the current value of num to the sum\_of\_numbers and then increment num by 1 (num += 1). The loop continues executing the code block as long as the condition num <= 10 remains True. When num becomes 11, the condition becomes False, and the loop terminates.

The loop iterates ten times, adding the numbers 1 to 10 to the sum\_of\_numbers, and then stops when the condition becomes False. The final value of sum\_of\_numbers is printed as the sum of numbers from 1 to 10, which is 55.

**Note:**

while loops are useful when you don't know the exact number of iterations needed in advance, and the loop's termination depends on a certain condition being met. Be cautious while using while loops to avoid infinite loops and ensure your loop eventually terminates.

#### Questions to practice (Day - 11)

Question 1: Write a while loop that prints numbers from 1 to 10. Expected Output:

1 2 3 4 5 6 7 8 9 10

​

Question 2: Create a while loop that calculates the sum of numbers from 1 to n, where n is the input. Expected Input: 5 Expected Output: 15 (1 + 2 + 3 + 4 + 5)

Question 3: Write a while loop that prints even numbers from 2 to 10. Expected Output

2 4 6 8 10

Question 4: Create a while loop that keeps prompting the user for a number until they enter a negative number. Expected Input: 5, 10, -2 Expected Output: (No output, just prompt for input)

Question 5: Write a while loop that counts down from 10 to 1. Expected Output:

10 9 8 7 6 5 4 3 2 1

​

Question 6: Create a while loop that asks the user to guess a secret number (e.g., 7) and continues until the correct number is guessed. Expected Output: (Depends on user input)

Question 7: Write a while loop that calculates the factorial of a given number. Expected Input: 4 Expected Output: 24 (4! = 4 \* 3 \* 2 \* 1 = 24)

Question 8: Create a while loop that prints the Fibonacci series up to n. Expected Input: 10 Expected Output:

0 1 1 2 3 5 8

​

Question 9: Write a while loop that reads numbers from the user until they enter the number 0. Then, it prints the sum of all the entered numbers. Expected Input: 3, 5, 2, 0 Expected Output: 10 (3 + 5 + 2)

Question 10: Create a while loop that prints the square of numbers from 1 to 5. Expected Output:

1 4 9 16 25

#**SHIVA625#12**

#### For Loop

In Python, a for loop is used to iterate over a sequence (such as a list, tuple, string, or range) and execute a block of code for each item in the sequence. The loop continues until all items in the sequence have been processed. .

The basic syntax of a for loop in Python is as follows:

for item in sequence: # Code block to be executed for each item in the sequence

​

Example :

# Example: Printing numbers from 1 to 5 using a for loop for num in range(1, 6): print(num)

​

Output:

1 2 3 4 5

​

Explanation: In this example, we use a for loop to iterate over a range of numbers from 1 to 5 (excluding 6). The range function generates a sequence of numbers starting from the first argument (1) and up to, but not including, the second argument (6). For each value of num in the sequence, the loop executes the code block, which prints the value of num.

The loop iterates five times, printing the numbers 1 to 5, and then stops when all items in the sequence have been processed.

#### Questions to practice (Day - 12):

Write a for loop that prints all numbers from 1 to 5.

Input: None Expected Output:

1 2 3 4 5

​Use a for loop to calculate the sum of numbers from 1 to 10.

Input: None Expected Output:

55

​

Write a for loop to print each character in the string "Hello".

Input: None Expected Output:

H e l l o

​

Write a for loop to print the first 10 even numbers.

Input: None Expected Output:

0 2 4 6 8 10 12 14 16 18

​

Create a for loop that doubles each number in the sequence: 1, 2, 3, 4, 5.

Input: None Expected Output:

2 4 6 8 10

**#SHIVA625#**13

### String Methods

len(): Returns the length of the string.

string = "Hello, World!"

length = len(string)

print(length) # Output: 13

lower(): Converts all characters in the string to lowercase.

string = "Hello, World!"

lower\_case = string.lower()

print(lower\_case) # Output: hello, world!

upper(): Converts all characters in the string to uppercase.

string = "Hello, World!"

upper\_case = string.upper()

print(upper\_case) # Output: HELLO, WORLD!

strip(): Removes leading and trailing whitespaces from the string.

string = " Hello, World! "

stripped\_string = string.strip()

print(stripped\_string) # Output: "Hello, World!"

replace(): Replaces occurrences of a substring with another substring.

string = "Hello, World!"

new\_string = string.replace("Hello", "Hi")

print(new\_string) # Output: "Hi, World!"

split(): Splits the string into a list of substrings based on a delimiter.

string = "apple,orange,banana"

fruits = string.split(",")

print(fruits) # Output: ['apple', 'orange', 'banana']

startswith(): Checks if the string starts with a specific prefix.

string = "Hello, World!"

result = string.startswith("Hello")

print(result) # Output: True

endswith(): Checks if the string ends with a specific suffix.

string = "Hello, World!"

result = string.endswith("World!")

print(result) # Output: True

count(): Returns the number of occurrences of a substring in the string.

string = "Hello, World!"

count = string.count("l")

print(count) # Output: 3

find(): Returns the index of the first occurrence of a substring. If not found, returns -1.

string = "Hello, World!"

index = string.find("o")

print(index) # Output: 4

isdigit(): Checks if all characters in the string are digits.

string = "12345"

result = string.isdigit()

print(result) # Output: True

isalpha(): Checks if all characters in the string are alphabetic.

string = "Hello"

result = string.isalpha()

print(result) # Output: True

**Questions for practice (day-13)**

1. Write a Python program that takes a user input string and converts it to lowercase using the lower() method. Input: "Hello, World!" Expected Output: "hello, world!"
2. Given a string "Hello, World!", find its length using the len() method. Input: "Hello, World!" Expected Output: 13
3. Write a Python function that takes a string as input and converts it to uppercase using the upper() method. Input: "Hello, World!" Expected Output: "HELLO, WORLD!"
4. Given a string "Hello123", check if it contains only alphabetic characters using the isalpha() method. Input: "Hello123" Expected Output: False
5. Write a Python program that takes a user input string with leading and trailing whitespaces and removes them using the strip() method. Input: " Python Programming " Expected Output: "Python Programming"
6. Given a string "I love Python programming", split it into words using the split() method and print each word on a new line. Input: "I love Python programming" Expected Output:
7. I
8. love
9. Python
10. programming
11. Write a Python program that takes a user input string and replaces all occurrences of "Python" with "Java" using the replace() method. Input: "I love Python programming, Python is great." Expected Output: "I love Java programming, Java is great."
12. Given a string "Hello, World!", check if it starts with the word "Hello" using the startswith() method. Input: "Hello, World!" Expected Output: True
13. Write a Python function that takes a string as input and checks if it ends with the word "end" using the endswith() method. Input: "Hello, World!" Expected Output: False
14. Given a string "banana", count the occurrences of the letter "a" using the count() method. Input: "banana" Expected Output: 3
15. Write a Python program that takes a user input string and finds the index of the first occurrence of the letter "l" using the find() method. Input: "Hello, World!" Expected Output: 2
16. Given a string "12345", check if it consists of only digits using the isdigit() method. Input: "12345" Expected Output: True

#**SHIVA625#14**

### Nested Loops

Nested loops is nothing but a loop in the loop!

Examples :

Nested Loop to Print a Rectangular Pattern:

# Example: Print a 3x3 rectangular pattern using nested loops

for i in range(3):

for j in range(3):

print("\*", end=" ")

print() # Move to the next line after each row

Output:

\* \* \*

\* \* \*

\* \* \*

Nested Loop to Print a Triangle Pattern:

# Example: Print a right-angled triangle pattern using nested loops

n = 5

for i in range(1, n + 1):

for j in range(i):

print("\*", end=" ")

print() # Move to the next line after each row

Output:

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

### Questions to practice (Day - 14):

.Write a nested loop that prints a square pattern of stars (asterisks). Expected Input: None Expected Output:

1. \*\*\*\*\*
2. \*\*\*\*\*
3. \*\*\*\*\*
4. \*\*\*\*\*
5. \*\*\*\*\*
6. Create a nested loop to print the multiplication table from 1 to 5 (up to 10 times each). Expected Input: None Expected Output:
7. 1 2 3 4 5
8. 2 4 6 8 10
9. 3 6 9 12 15
10. 4 8 12 16 20
11. 5 10 15 20 25
12. Write a nested loop to print a right-angled triangle of numbers in ascending order. Expected Input: None Expected Output:
13. 1
14. 1 2
15. 1 2 3
16. 1 2 3 4
17. 1 2 3 4 5
18. Create a nested loop that prints a hollow square pattern of stars (asterisks). Expected Input: None Expected Output:
19. \*\*\*\*\*
20. \* \*
21. \* \*
22. \* \*
23. \*\*\*\*\*
24. Write a nested loop to calculate and print the sum of numbers from 1 to 5 using nested iteration. Expected Input: None Expected Output: 15
25. Create a nested loop to print the reverse of the multiplication table from 1 to 5 (up to 10 times each). Expected Input: None Expected Output:
26. 5 4 3 2 1
27. 10 8 6 4 2
28. 15 12 9 6 3
29. 20 16 12 8 4
30. 25 20 15 10 5
31. Write a nested loop that prints a right-angled triangle of stars (asterisks). Expected Input: None Expected Output:
32. \*
33. \*\*
34. \*\*\*
35. \*\*\*\*
36. \*\*\*\*\*
37. Create a nested loop to find and print prime numbers from 2 to 20. Expected Input: None Expected Output: 2 3 5 7 11 13 17 19
38. Write a nested loop that prints a diamond pattern of stars (asterisks). Expected Input: None Expected Output:
39. \*
40. \*\*\*
41. \*\*\*\*\*
42. \*\*\*
43. \*
44. Create a nested loop to find and print the factorial of numbers from 1 to 5.

Expected Input: None Expected Output: 1 2 6 24 120

**#SHIVA625#15**

#### Loop Control Statements

: break, continue, and pass.

break Statement: The break statement is used to terminate a loop prematurely when a certain condition is met.

Python

Copy

# Example: Using 'break' to stop the loop when the value reaches 3 for num in range(1, 6): if num == 3: break print(num)

​

Output:

1 2

​

Explanation: In this example, the loop runs from 1 to 5, but when num becomes 3, the break statement is encountered, and the loop is terminated immediately.

continue Statement: The continue statement is used to skip the current iteration of a loop and move to the next iteration.

# Example: Using 'continue' to skip printing even numbers for num in range(1, 6): if num % 2 == 0: continue print(num)

​

Output:

1 3 5

​

Explanation: In this example, the loop runs from 1 to 5, but when an even number is encountered (2 and 4), the continue statement is used, skipping the print statement, and the loop moves to the next iteration

pass Statement: The pass statement is used as a placeholder when you don't want to execute any code inside a loop or a conditional block.

# Example: Using 'pass' to do nothing inside the loop for num in range(1, 6): pass

​Explanation: In this example, the loop runs from 1 to 5, but the pass statement does nothing inside the loop.

#### Questions to practice (Day - 15):

Write a Python program to print all numbers from 1 to 10, but stop the loop immediately when reaching 5 using the break statement.

Expected Output:

1 2 3 4

​

Given a list of numbers [1, 2, 3, 4, 5], use a for loop to print the elements one by one. However, if the element is 3, skip it using the continue statement.

Expected Output:

1 2 4 5

​

Write a Python function that takes a string as input and checks if it contains the letter 'o'. If it does, print "Found 'o'" and use the break statement to stop searching.

Input: "Hello, World!" Expected Output:

Found 'o'

​

Given a list of numbers [1, 2, 3, 4, 5], use a for loop to double each element and print the result. However, if the element is 4, use the continue statement to skip it.

Expected Output:

2 4 6 10

​Write a Python program to print all numbers from 1 to 20 using a while loop. However, stop the loop when reaching 15 using the break statement.

Expected Output: 1 2 3 ... (up to 15)

**#SHIVA625#16**

### Comparing Strings in Python:

Equality and Inequality Comparison (==, !=):

# Example: Equality and inequality comparison

string1 = "hello"

string2 = "Hello"

if string1 == string2:

print("Both strings are equal.")

else:

print("The strings are not equal.")

if string1 != string2:

print("The strings are not equal.")

else:

print("Both strings are equal.")

Output:

The strings are not equal.

The strings are not equal.

Example 2:

# Example: Lexicographical comparison

string1 = "apple"

string2 = "banana"

if string1 < string2:

print("string1 comes before string2.")

else:

print("string1 comes after or is equal to string2.")

if string1 <= string2:

print("string1 comes before or is equal to string2.")

else:

print("string1 comes after string2.")

if string1 > string2:

print("string1 comes after string2.")

else:

print("string1 comes before or is equal to string2.")

if string1 >= string2:

print("string1 comes after or is equal to string2.")

else:

print("string1 comes before string2.")

Output:

string1 comes before string2.

string1 comes before or is equal to string2.

string1 comes before or is equal to string2.

string1 comes before string2.

### Questions to practice (Day - 16):

1. Given two strings "Python" and "python", check if they are equal using the equality operator (==).

Input: "Python", "python" Expected Output: False

1. Write a Python function that takes a string as input and checks if it is an empty string using the equality operator (==).

Input: "" Expected Output: True

**#SHIVA625#17**

### Lists

### List is used to store the collectin of items

### It is mutable it means modify the items and delete the items

### Items are separated by the (,) comma and it is denoted by the square bracket([])

Examples :

Creating Lists: You can create a list by enclosing elements in square brackets.

# Example: Creating a list of numbers

numbers = [1, 2, 3, 4, 5]

# Creating a list of strings

fruits = ["apple", "banana", "orange"]

# Creating a mixed-type list

mixed\_list = [1, "apple", True, 3.14]

Accessing Elements:

# Example: Accessing elements in a list

fruits = ["apple", "banana", "orange"]

print(fruits[0]) # Output: "apple"

print(fruits[2]) # Output: "orange"

Modifying Elements:

# Example: Modifying elements in a list

fruits = ["apple", "banana", "orange"]

fruits[1] = "grape"

print(fruits) # Output: ['apple', 'grape', 'orange']

List Operations:

# Example: List operations

list1 = [1, 2, 3]

list2 = [4, 5, 6]

# Concatenation

result = list1 + list2

print(result) # Output: [1, 2, 3, 4, 5, 6]

# Repetition

repeated\_list = list1 \* 3

print(repeated\_list) # Output: [1, 2, 3, 1, 2, 3, 1, 2, 3]

List Methods:

# Example: List methods

fruits = ["apple", "banana", "orange"]

# Adding elements

fruits.append("grape")

print(fruits) # Output: ['apple', 'banana', 'orange', 'grape']

# Removing elements

fruits.remove("banana")

print(fruits) # Output: ['apple', 'orange', 'grape']

# Sorting elements

fruits.sort()

print(fruits) # Output: ['apple', 'grape', 'orange']

Lists are widely used in Python for storing and manipulating data. They are flexible, efficient, and an essential tool for various programming tasks. Understanding lists and their methods will help you work with collections of data effectively in Python.

### Questions to practice (Day - 17):

1. Write a Python program that takes a list of numbers [1, 2, 3, 4, 5] and prints each number on a new line.

Input: [1, 2, 3, 4, 5] Expected Output:

1

2

3

4

5

1. Given a list of strings ["apple", "banana", "orange"], concatenate all the strings together with a space in between and print the result.

Input: ["apple", "banana", "orange"] Expected Output: "apple banana orange"

1. Write a Python function that takes a list of numbers as input and returns the sum of all the numbers.

Input: [1, 2, 3, 4, 5] Expected Output: 15

1. Given a list of numbers [10, 20, 30, 40, 50], find the maximum number using the max() function and print the result.

Input: [10, 20, 30, 40, 50] Expected Output: 50

1. Write a Python program that takes a list of names ["Alice", "Bob", "Charlie"] and checks if a given name (e.g., "Alice") is present in the list. Print "Name found" if the name is in the list; otherwise, print "Name not found".

Input: Names = ["Alice", "Bob", "Charlie"], Name = "Alice" Expected Output: "Name found"

**#SHIVA625#18**

**Tuples**

Idi kuda list laaga oka data structure eyy! kaani tuples are immutable ante okasari deenni create chesnaka elements ni nuv add or remove cheyyalev ani. Elements anni parentheses () lo esi pedtam seperated by a comma.

Example chusthe:

Creating Tuples:

# Example: Creating a tuple

fruits = ("apple", "banana", "orange")

coordinates = (3.14, 2.71)

Accessing Elements:

# Example: Accessing elements in a tuple

fruits = ("apple", "banana", "orange")

print(fruits[0]) # Output: "apple"

print(fruits[2]) # Output: "orange"

Tuple Packing and Unpacking:

# Example: Tuple packing and unpacking

person = ("John", 30, "New York")

name, age, city = person

print(name) # Output: "John"

print(age) # Output: 30

print(city) # Output: "New York"

Tuple Functions: Tuples support various built-in functions like len(), min(), and max().

# Example: Tuple functions

numbers = (5, 2, 8, 1, 7)

# Length of the tuple

length = len(numbers)

print(length) # Output: 5

# Maximum and minimum elements in the tuple

maximum = max(numbers)

minimum = min(numbers)

print(maximum, minimum) # Output: 8 1

Tuple Concatenation and Repetition:

# Example: Tuple concatenation and repetition

tuple1 = (1, 2, 3)

tuple2 = (4, 5, 6)

# Concatenation

result = tuple1 + tuple2

print(result) # Output: (1, 2, 3, 4, 5, 6)

# Repetition

repeated\_tuple = tuple1 \* 3

print(repeated\_tuple) # Output: (1, 2, 3, 1, 2, 3, 1, 2, 3)

Tuples are handy when you need to store data that should remain unchanged throughout your program's execution. Their immutability makes them suitable for certain use cases, providing a safe and efficient way to group related data.

**Questions to practice (Day - 18):**

1. Create a tuple containing three elements: 'apple', 5, and True. Expected Input: No input required. Expected Output: ('apple', 5, True)
2. Access the second element from the given tuple: ('cat', 'dog', 'bird', 'fish'). Expected Input: No input required. Expected Output: 'dog'
3. Concatenate two tuples: (1, 2, 3) and ('a', 'b', 'c'). Expected Input: No input required. Expected Output: (1, 2, 3, 'a', 'b', 'c')
4. Find the length of the tuple: (10, 20, 30, 40, 50). Expected Input: No input required. Expected Output: 5
5. Check if the element 25 exists in the tuple: (10, 20, 30, 40, 50). Expected Input: No input required. Expected Output: False
6. Create a new tuple with elements from the given tuple (3, 6, 9) repeated 3 times. Expected Input: No input required. Expected Output: (3, 6, 9, 3, 6, 9, 3, 6, 9)
7. Perform packing and unpacking on a tuple containing the names of three fruits: 'apple', 'banana', and 'orange'. Use unpacking to assign each fruit to three variables: fruit1, fruit2, and fruit3.

Expected Input: No input required. Expected Output:

* fruit1 = 'apple'
* fruit2 = 'banana'
* fruit3 = 'orange'

**#SHIVA625#19**

**Dictionaries**

Dictionaries are used to store the key value pairs. Dictionaries are mutable, it means we can modify the values . Dictionaries are defined using curly braces { }, and each key-value pair is separated by a colon :.

Examples:

Creating Dictionaries: You can create a dictionary by enclosing key-value pairs in curly braces.

# Example: Creating a dictionary of student information

student = {

"name": "Dodagatta Nihar",

"age": 25,

"major": "Computer Science",

"gpa": 3.8

}

Accessing Values: You can access values in a dictionary using keys. Dictionary keys are unique.

# Example: Accessing values in a dictionary

student = {

"name": "Dodagatta Nihar",

"age": 25,

"major": "Computer Science",

"gpa": 3.8

}

print(student["name"]) # Output: "Dodagatta Nihar"

print(student["gpa"]) # Output: 3.8

**Questions to practice (Day - 19):**

1. Question: Create an empty dictionary. Expected Output: {}
2. Question: Create a dictionary to store the age of two people, "John" and "Alice." John is 25 years old, and Alice is 30 years old. Expected Output: {"John": 25, "Alice": 30}
3. Question: Access the value associated with the key "city" from the given dictionary. Expected Input: Dictionary: {"name": "Alice", "city": "New York", "age": 30} Expected Output: "New York"
4. Question: Create a dictionary to store the contact information of a person. The person's name is "Bob," and their email is "[bob@example.com](mailto:bob@example.com)." Expected Output: {"name": "Bob", "email": "[bob@example.com](mailto:bob@example.com)"}
5. Question: Access the value associated with the key "score" from the given dictionary. Expected Input: Dictionary: {"name": "John", "age": 22, "score": 85} Expected Output: 85
6. Question: Create a dictionary to represent a rectangle. The rectangle has a width of 10 and a height of 5. Expected Output: {"width": 10, "height": 5}
7. Question: Access the value associated with the key "phone" from the given dictionary. If the key does not exist, return "Not available." Expected Input: Dictionary: {"name": "Eve", "age": 27} Expected Output: "Not available"

**#SHIVA625#20**

**Working with Dictionaries**

Modifying and Adding Key-Value Pairs:

# Example: Modifying and adding key-value pairs

student = {

"name": "Dodagatta Nihar",

"age": 25,

"major": "Computer Science",

"gpa": 3.8

}

# Modifying value

student["gpa"] = 4.0

# Adding new key-value pair

student["university"] = "XYZ University"

print(student)

# Output: {'name': 'Dodagatta Nihar', 'age': 25, 'major': 'Computer Science', 'gpa': 4.0, 'university': 'XYZ University'}

Dictionary Methods: Python provides various built-in methods to perform common operations on dictionaries.

# Example: Dictionary methods

student = {

"name": "Dodagatta Nihar",

"age": 25,

"major": "Computer Science",

"gpa": 3.8

}

# Get keys and values

keys = student.keys()

values = student.values()

print(keys) # Output: dict\_keys(['name', 'age', 'major', 'gpa'])

print(values) # Output: dict\_values(['Dodagatta Nihar', 25, 'Computer Science', 3.8])

# Check if a key exists in the dictionary

if "major" in student:

print("Major:", student["major"]) # Output: Major: Computer Science

# Remove a key-value pair from the dictionary

removed\_value = student.pop("age")

print("Removed Value:", removed\_value) # Output: Removed Value: 25

print(student) # Output: {'name': 'Dodagatta Nihar', 'major': 'Computer Science', 'gpa': 3.8}

**Questions to practice (Day - 20):**

1. Question: Given the initial dictionary employee as {"name": "Alice", "age": 30}, modify the value of the key "age" to 35. Expected Input: employee = {"name": "Alice", "age": 30} Expected Output: {"name": "Alice", "age": 35}
2. Question: Add a new key-value pair to the dictionary fruits, where the key is "orange" and the value is 3. Expected Input: fruits = {"apple": 5, "banana": 7} Expected Output: {"apple": 5, "banana": 7, "orange": 3}
3. Question: Given the dictionary inventory, remove the key "sugar" and its associated value from the dictionary. Expected Input: inventory = {"apple": 10, "banana": 15, "sugar": 2} Expected Output: {"apple": 10, "banana": 15}
4. Question: Create a function called add\_stock that takes a dictionary stock and an item item\_name (string) as input and adds 1 to the value of the corresponding key in the dictionary. Return the modified dictionary. Expected Input: stock = {"apple": 10, "banana": 15}, item\_name = "banana" Expected Output: {"apple": 10, "banana": 16}
5. Question: Given the dictionary scores, check if the key "Alice" exists. If it exists, print the associated value; otherwise, print "Key not found." Expected Input: scores = {"Bob": 85, "Charlie": 90, "Alice": 78} Expected Output: 78
6. Question: Write a function called count\_vowels that takes a string text as input and returns a dictionary containing the count of each vowel (a, e, i, o, u) in the text. Ignore case sensitivity. Expected Input: text = "Hello, World!" Expected Output: {"a": 0, "e": 1, "i": 0, "o": 2, "u": 0}
7. Question: Given the dictionary student\_grades, find the highest grade and its corresponding student name. The dictionary contains student names as keys and their grades as values. Expected Input: student\_grades = {"Alice": 85, "Bob": 90, "Charlie": 78} Expected Output: ("Bob", 90)

**#SHIVA625#21**

### Sets

* Set is nothing but a collection unordered unique elements
* A Python set is the collection of the unordered items. Each element in the set must be unique, immutable, and the sets remove the duplicate elements. Sets are mutable which means we can modify it after its creation.
* Unlike other collections in Python, there is no index attached to the elements of the set, i.e., we cannot directly access any element of the set by the index. However, we can print them all together, or we can get the list of elements by looping through the set.

## **Creating a set**

* The set can be created by enclosing the comma-separated immutable items with the curly braces {}. Python also provides the set() method, which can be used to create the set by the passed sequence.

In Python, sets are defined using curly braces {} or by using the set() constructor.

Creating a Set:

# Using curly braces

my\_set = {1, 2, 3, 4, 5}

# Using the set() constructor

another\_set = set([5, 6, 7, 8, 9])

Adding Elements to a Set:

my\_set = {1, 2, 3}

my\_set.add(4)

my\_set.add(5)

print(my\_set) # Output: {1, 2, 3, 4, 5}

Removing Elements from a Set:

my\_set = {1, 2, 3, 4, 5}

my\_set.remove(3)

print(my\_set) # Output: {1, 2, 4, 5}

# Remove an element without raising an error if the element is not present

my\_set.discard(10)

# Pop removes and returns an arbitrary element from the set

popped\_element = my\_set.pop()

print(popped\_element, my\_set) # Output (random order): 1, {2, 4, 5}

### Questions to practice (Day - 21):

**Question 1: Creating a set** Expected Input: None Expected Output: An empty set

**Question 2: Adding elements to a set** Expected Input: Set: {1, 2, 3}, Element to add: 4 Expected Output: Set: {1, 2, 3, 4}

**Question 3: Removing an element from a set** Expected Input: Set: {1, 2, 3, 4}, Element to remove: 3 Expected Output: Set: {1, 2, 4}

**Question 4: Removing a non-existent element from a set** Expected Input: Set: {1, 2, 3, 4}, Element to remove: 5 Expected Output: Set: {1, 2, 3, 4}

**Question 5: Creating a set from a list** Expected Input: List: [5, 10, 15, 20] Expected Output: Set: {5, 10, 15, 20}

**Question 6: Adding multiple elements to a set** Expected Input: Set: {1, 2, 3}, Elements to add: {4, 5} Expected Output: Set: {1, 2, 3, 4, 5}

**Question 7: Removing elements using discard()** Expected Input: Set: {10, 20, 30, 40}, Elements to remove: {20, 50} Expected Output: Set: {10, 30, 40}

#**SHIVA625#22**

### Working with Sets

Set Operations:

set1 = {1, 2, 3, 4}

set2 = {3, 4, 5, 6}

# Union of two sets

union\_set = set1.union(set2)

print(union\_set) # Output: {1, 2, 3, 4, 5, 6}

# Intersection of two sets

intersection\_set = set1.intersection(set2)

print(intersection\_set) # Output: {3, 4}

# Difference between two sets

difference\_set = set1.difference(set2)

print(difference\_set) # Output: {1, 2}

# Symmetric Difference (elements in either set, but not in both)

symmetric\_difference\_set = set1.symmetric\_difference(set2)

print(symmetric\_difference\_set) # Output: {1, 2, 5, 6}

Membership Test:

my\_set = {1, 2, 3, 4, 5}

print(3 in my\_set) # Output: True

print(6 not in my\_set) # Output: True

### Questions to practice (Day - 22):

**Question 1: Symmetric Difference between two sets** Expected Input: Set A: {1, 2, 3}, Set B: {3, 4, 5} Expected Output: {1, 2, 4, 5}

**Question 2: Checking membership in a set** Expected Input: Set: {1, 2, 3}, Element to check: 2 Expected Output: True

**Question 3: Union of two sets** Expected Input: Set A: {1, 2, 3}, Set B: {3, 4, 5} Expected Output: {1, 2, 3, 4, 5}

**Question 4: Intersection of two sets** Expected Input: Set A: {1, 2, 3}, Set B: {3, 4, 5} Expected Output: {3}

**Question 5: Difference between two sets** Expected Input: Set A: {1, 2, 3, 4}, Set B: {3, 4, 5} Expected Output: {1, 2}

**Question 6: Removing duplicate elements from a list using a set** Expected Input: List: [1, 2, 2, 3, 4, 4, 5] Expected Output: [1, 2, 3, 4, 5]

**Question 7: Finding the length of a set** Expected Input: Set: {1, 2, 3, 4, 5} Expected Output: 5

**Question 8: Checking if one set is a subset of another** Expected Input: Set A: {1, 2, 3}, Set B: {1, 2, 3, 4, 5} Expected Output: True

**Question 9: Checking if two sets are disjoint** Expected Input: Set A: {1, 2, 3}, Set B: {4, 5, 6} Expected Output: True

**Question 10: Removing all elements from a set** Expected Input: Set: {1, 2, 3, 4, 5} Expected Output: An empty set

**#SHIVA625#23**

#### Functions Part - 1

Functions are used to create a reusable code and it is used to proper organization of code

**Creating a function**: To define a function, you use the def keyword, followed by the function name, a pair of parentheses (), and a colon :. The code block inside the function is indented and contains the instructions that the function will execute when called.

def greet(): print("Hello, World!") # Calling the function greet() # Output: Hello, World!

​

Function Parameters: You can define parameters inside the parentheses to pass data into the function. Parameters act as placeholders for the actual values that you provide when calling the function.

Python

Copy

def greet\_user(name): print(f"Hello, {name}!") # Calling the function with an argument greet\_user("Alice") # Output: Hello, Alice!

​

Return Statement: Functions can return values using the return keyword. When a function returns a value, you can capture it and use it elsewhere in your code.

Python

Copy

def add\_numbers(a, b): return a + b # Calling the function and storing the result in a variable result = add\_numbers(3, 5) print(result) # Output: 8

​

#### Questions to practice (Day - 23):

**Problem 1: Simple function without parameters** Write a function called greet that takes no parameters and returns the string "Hello, World!"

Expected Output:

message = greet() print(message) # Output: Hello, World!

​

**Problem 2: Function with one parameter** Write a function called greet\_user that takes a name parameter and returns a greeting message with the name.

Expected Output:

message = greet\_user("Alice") print(message) # Output: Hello, Alice!

​

**Problem 3: Function with two parameters and mathematical operation** Write a function called add\_numbers that takes two parameters a and b, and returns their sum.

Expected Output:

result = add\_numbers(5, 10) print(result) # Output: 15

​

**Problem 4: Function with a conditional statement** Write a function called check\_even\_odd that takes a number parameter and returns "Even" if the number is even, and "Odd" if the number is odd.

Expected Output:

result = check\_even\_odd(7) print(result) # Output: Odd

​

**Problem 5: Function with default parameter value** Write a function called greet\_user that takes an optional name parameter with a default value of "Guest". The function should return a greeting message with the provided name or "Guest" if no name is provided.

Expected Output:

message = greet\_user() print(message) # Output: Hello, Guest!

​

**Problem 6: Function with a variable number of arguments (variadic function)** Write a function called sum\_numbers that takes a variable number of arguments and returns the sum of all the arguments.

Expected Output:

result = sum\_numbers(2, 4, 6, 8, 10) print(result) # Output: 30

**Problem 7: Nested Functions** Write a function called square that takes a number x as input and returns the square of that number using a nested function called multiply to perform the calculation.

Expected Output:

result = square(5) print(result) # Output: 25

#SHIVA625#24

#### Functions Part – 2

Default Parameters:

def greet\_user(name="Guest"): print(f"Hello, {name}!") greet\_user() # Output: Hello, Guest! greet\_user("Alice") # Output: Hello, Alice!

​

Calling Functions Inside Functions:

def greet(name): return f"Hello, {name}!" def greet\_and\_emphasize(name): greeting = greet(name) return greeting.upper() + "!!!" result = greet\_and\_emphasize("Alice") print(result) # Output: HELLO, ALICE!!!

​

**Scope of variable:**

* We can declare a global variable in the we can use any where in the program
* We can declare a variable inside the function we use that particular function only

global\_variable = "I'm global" def function\_with\_local\_variable(): local\_variable = "I'm local" print(global\_variable) # Output: I'm global print(local\_variable) # Output: I'm local function\_with\_local\_variable() print(global\_variable) # Output: I'm global # print(local\_variable) # Raises NameError: name 'local\_variable' is not defined

​

#### Questions to practice (Day - 24):

**Question 1: Simple function with a default parameter** Write a function called greet\_user that takes an optional name parameter with a default value of "Guest". The function should return a greeting message with the provided name or "Guest" if no name is provided.

Expected Output:

greet\_user() # Output: Hello, Guest! greet\_user("Alice") # Output: Hello, Alice!

​

**Question 2: Function with multiple parameters** Write a function called calculate\_sum that takes three parameters a, b, and c, and returns the sum of the three numbers.

Expected Output:

result = calculate\_sum(5, 10, 15) print(result) # Output: 30

​

**Question 3: Function calling another function** Write a function called square that takes a number x as input and returns the square of that number. Then, write a function called square\_and\_double that takes a number x, calls the square function, and returns twice the square value.

Expected Output:

result = square\_and\_double(5) print(result) # Output: 50

​

**Question 4: Nested functions and variable scopes** Write a function called outer\_function that has a local variable outer\_variable with the value "I'm outer". Inside the outer\_function, define another function called inner\_function that has a local variable inner\_variable with the value "I'm inner". The inner\_function should print both the outer\_variable and inner\_variable. Then, call the outer\_function and print the outer\_variable outside the function.

Expected Output:

I'm outer I'm inner I'm outer

​

**Question 5: Function returning multiple values** Write a function called divide\_and\_remainder that takes two numbers a and b as input and returns their division result and remainder.

Expected Output:

result = divide\_and\_remainder(15, 4) print(result) # Output: (3, 3)

​

**Question 6: Recursive function** Write a recursive function called factorial that takes a positive integer n as input and returns its factorial.

Expected Output:

result = factorial(5) print(result) # Output: 120

**#SHIVA625#25**

#### Function Arguments

Positional Arguments: Positional arguments are the most basic type of arguments. They are matched to function parameters based on their order of appearance.

def greet(name, age): print(f"Hello, {name}! You are {age} years old.") # Calling the function with positional arguments greet("Alice", 25) # Output: Hello, Alice! You are 25 years old.

​

Keyword Arguments: Keyword arguments are specified with the parameter name followed by the value, separated by an equal sign. These arguments allow you to pass the values to the function in any order.

def greet(name, age): print(f"Hello, {name}! You are {age} years old.") # Calling the function with keyword arguments in a different order greet(age=25, name="Bob") # Output: Hello, Bob! You are 25 years old.

​

Default Values: You can provide default values for function parameters. If a value is not passed for that parameter when the function is called, the default value will be used.

def greet(name, age=30): print(f"Hello, {name}! You are {age} years old.") # Calling the function without specifying 'age' argument greet("Alice") # Output: Hello, Alice! You are 30 years old. # Calling the function with 'age' argument greet("Bob", 25) # Output: Hello, Bob! You are 25 years old.

​

#### Questions to practice (Day - 25):

**Question 1: Positional Arguments** Write a function called add that takes two positional arguments a and b, and returns their sum.

Expected Output:

result = add(5, 10) print(result) # Output: 15

​

**Question 2: Keyword Arguments** Write a function called greet that takes two keyword arguments name and age, and prints a greeting message with the provided values.

Expected Output:

greet(name="Alice", age=25) # Output: Hello, Alice! You are 25 years old.

​

**Question 3: Default Values** Write a function called multiply that takes two arguments a and b, with b having a default value of 2. The function should return the product of a and b.

Expected Output:

result = multiply(5) print(result) # Output: 10

​

**Question 4: Mixing Positional and Keyword Arguments** Write a function called print\_info that takes three positional arguments name, age, and country, and prints the information in a formatted message.

Expected Output:

print\_info("Alice", 25, "USA") # Output: Name: Alice, Age: 25, Country: USA

#**SHIVA625#26**

#### Built-in Functions

max() and min() - Returns the maximum and minimum values from a sequence.

numbers = [10, 5, 8, 15, 3] max\_value = max(numbers) min\_value = min(numbers) print(max\_value) # Output: 15 print(min\_value) # Output: 3

​

sum() - Returns the sum of elements in a sequence.

numbers = [1, 2, 3, 4, 5] total\_sum = sum(numbers) print(total\_sum) # Output: 15

​

range() - Generates a sequence of numbers within a specified range.

# Generate numbers from 0 to 4 my\_range = range(5) print(list(my\_range)) # Output: [0, 1, 2, 3, 4]

​

sorted() - Returns a new sorted list from the elements in an iterable.

numbers = [3, 1, 4, 1, 5, 9, 2, 6] sorted\_numbers = sorted(numbers) print(sorted\_numbers) # Output: [1, 1, 2, 3, 4, 5, 6, 9]

​

These are just a few examples of the built-in functions available in Python. There are many more built-in functions that can be utilized for various purposes, and you can find more details in the Python documentation.

#### Questions to practice (Day - 26):

**Question 1: Using max() and min()** Write a Python program that takes a list of numbers as input and prints the maximum and minimum values from that list.

Expected Output:

numbers = [10, 5, 8, 15, 3] print(max\_value) # Output: 15 print(min\_value) # Output: 3

​

**Question 2: Using sum()** Write a Python program that takes a list of numbers as input and prints the sum of all the elements in that list.

Expected Output:

numbers = [1, 2, 3, 4, 5] print(total\_sum) # Output: 15

​

**Question 3: Using range()** Write a Python program that generates a list of even numbers from 2 to 10 using the range() function.

Expected Output:

# Generate numbers from 2 to 10 with a step of 2 print(list(my\_range)) # Output: [2, 4, 6, 8, 10]

​

**Question 4: Using sorted()** Write a Python program that takes a list of numbers as input, sorts the list in ascending order using the sorted() function, and prints the sorted list.

Expected Output:

numbers = [3, 1, 4, 1, 5, 9, 2, 6] print(sorted\_numbers) # Output: [1, 1, 2, 3, 4, 5, 6, 9]

​

**Question 5: Using len()** Write a Python program that takes a string as input and prints the length of the string using the len() function.

Expected Output:

text = "Hello, World!" print(length) # Output: 13

​

**Question 6: Using abs()** Write a Python program that takes a number as input and prints its absolute value using the abs() function.

Expected Output:

number = -10 print(abs\_value) # Output: 10

​

**Question 7: Using all() and any()** Write a Python program that takes a list of boolean values as input and checks if all values are True using the all() function. Then, check if at least one value is True using the any() function.

Expected Output:

bool\_list = [True, True, False, True] print(all\_true) # Output: False print(any\_true) # Output: True

**#SHIVA625#27**

#### Pattern Printing

Right Triangle Star Pattern

Printing star patterns is a common exercise to practice looping in Python. Here are some simple star patterns you can print:

Right Triangle Star Pattern: Print a right-angled triangle of stars.

def right\_triangle(rows): for i in range(1, rows + 1): print("\*" \* i) right\_triangle(5) # Output: # \* # \*\* # \*\*\* # \*\*\*\* # \*\*\*\*\*

​

Pyramid Star Pattern: Print a pyramid of stars.

def pyramid(rows): for i in range(1, rows + 1): spaces = " " \* (rows - i) stars = "\*" \* (2 \* i - 1) print(spaces + stars) pyramid(5) # Output: # \* # \*\*\* # \*\*\*\*\* # \*\*\*\*\*\*\* # \*\*\*\*\*\*\*\*\*

​

Hollow Square Star Pattern: Print a hollow square of stars.

def hollow\_square(rows): for i in range(rows): if i == 0 or i == rows - 1: print("\*" \* rows) else: print("\*" + " " \* (rows - 2) + "\*") hollow\_square(5) # Output: # \*\*\*\*\* # \* \* # \* \* # \* \* # \*\*\*\*\*

​

These are just a few examples of star patterns you can create using loops in Python. You can modify the number of rows or add more patterns by adjusting the loop ranges and the number of stars and spaces printed in each row.

#### Questions to practice (Day - 27):

**Question 1: Inverted Right Triangle Star Pattern** Expected Input: Rows: 5 Expected Output:

\*\*\*\*\* \*\*\*\* \*\*\* \*\* \*

​

**Question 2: Half Pyramid Star Pattern** Expected Input: Rows: 5 Expected Output:

\* \*\* \*\*\* \*\*\*\* \*\*\*\*\*

​

**Question 3: Hollow Right Triangle Star Pattern** Expected Input: Rows: 5 Expected Output:

\* \*\* \* \* \* \* \*\*\*\*\*

​

**Question 4: Full Pyramid Star Pattern** Expected Input: Rows: 5 Expected Output:

\* \*\*\* \*\*\*\*\* \*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*

​

**Question 5: Rhombus Star Pattern** Expected Input: Rows: 5 Expected Output:

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*

\*

#**SHIVA625#28**

**PRACTICE**

1. Create a variable name and assign your name to it. Print a greeting message using the variable.
2. Convert the string "42" to an integer and assign it to a variable num. Print the value and type of num.
3. Define a tuple with three elements: "cat", "dog", and "rabbit". Print the third element.
4. Swap the values of two variables, a and b, without using a temporary variable.
5. Write a program that takes your age as input and prints a message: "You are [age] years old."
6. Calculate the product of two numbers entered by the user and print the result.
7. Format the variables item = "book" and price = 25.99 into a sentence.
8. Write a function that takes an integer as input and returns "Positive", "Negative", or "Zero".
9. Create a program that checks if a user-input number is even or odd.
10. Write a program that determines if a year entered by the user is a leap year.
11. Print the numbers from 1 to 5 using a for loop.
12. Write a while loop that calculates the sum of numbers from 1 to 20.
13. Print each character of the string "Hello" using a loop.
14. Create a list of your favorite fruits and print the second fruit on the list.
15. Add the number 7 to a set. Add the number 7 again and observe the set's behavior.
16. Write a function that takes a list as input and returns a new list without duplicate elements.
17. Create a tuple containing your birth year, birth month, and birth day.
18. Define a function calculate\_average that takes a list of numbers as input and returns their average.
19. Write a function power that takes two arguments, base and exponent, and calculates base raised to the power of exponent.
20. Define a function that takes any number of strings as arguments and returns them concatenated.
21. Check if the word "apple" is present in the string "I like apples and oranges".
22. Count the occurrences of the letter "i" in the string "Mississippi".
23. Reverse the string "Python" using slicing.
24. Create a dictionary representing a car with keys "make", "model", and "year". Print the car's model.
25. Add a new car to the dictionary, modify the year of an existing car, and retrieve the make of a specific car

**#SHIVA625#29**

**PRACTICE**

* Create a variable name and assign your name to it. Print a message using the variable, such as: "Hello, [name]!"
* Convert the string "37" to an integer and assign it to a variable num. Print the value and type of num.
* Define a tuple with four elements: "apple", "banana", "cherry", and "date". Print the second element.
* Swap the values of two variables, x and y, without using a temporary variable.
* Write a program that takes your age as input and prints a sentence like: "You are [age] years old."
* Calculate the product of two numbers entered by the user and print the result.
* Format the variables item = "hat" and price = 19.99 into a sentence.
* Write a function that takes an integer as input and returns whether it's a prime number or not.
* Create a program that checks if a user-input number is positive, negative, or zero.
* Write a program that prints the Fibonacci sequence up to a specified number of terms.
* Print the multiplication table of 7 using nested loops.
* Print each character of the string "Python" on a new line using nested loops.
* Create a list of your favorite colors and print each color in a sentence: "I like [color]."
* Add the number 5 to a set. Add the number 5 again and observe the set's behavior.
* Write a function that takes a list as input and returns a new list with unique elements only.
* Create a tuple containing your birth year, birth month, and birth day. Print each element separately.
* Define a function calculate\_average that takes a list of numbers as input and returns their average.
* Write a function factorial that calculates the factorial of a given number using recursion.
* Define a function that takes any number of strings as arguments and returns them concatenated.
* Check if the word "apple" is present in the string "I like apples and oranges."
* Count the occurrences of the letter "t" in the string "Testing the text for t's."
* Reverse the string "Python" using a loop.
* Create a dictionary representing a book with keys "title", "author", and "year". Print the author's name.
* Add a new book to the dictionary, modify the title of an existing book, and retrieve the year of a specific book.
* Print a pattern of right-angled triangles using nested loops:

\*

\*\*

\*\*\*

\*\*\*\* \*\*\*\*\*

**#SHIVA625#30**

* What is the purpose of a variable in Python?

a) To store data

b) To perform calculations

c) To create loops

d) To define functions

1.Which of the following data types is immutable in Python?

a) List

b) Set

c) Dictionary

d) Tuple

2.How do you convert the string "50" to an integer in Python?

a) int("50")

b) convert\_to\_int("50")

c) str\_to\_int("50")

d) integer("50")

3.Which loop is best suited when the number of iterations is known in advance? a) for loop

b) while loop

c) Both are equally suitable

d) It depends on the situation

4.What is the output of the following code snippet?

x = 5 x += 3 print(x)

​

a) 3 b) 5 c) 8 d) 15

5.How do you take user input in Python?

a) input() b) user\_input() c) get\_input() d) read\_input()

5.Which conditional statement is used to check multiple conditions?

a) if b) else c) elif d) then

6.What does the range(3, 10) function generate?

a) A sequence from 0 to 9 b) A sequence from 3 to 10 c) A sequence from 3 to 9 d) A sequence from 3 to 11

7. What is the output of print("Hello", end=" ") followed by print("World")?

a) Hello World b) Hello c) World d) Hello\\nWorld

8.Which of the following is used to define a function in Python?

a) func b) def c) define d) function

Which data type is best suited to store a collection of unique values?

a) List b) Set c) Dictionary d) Tuple

How do you check if a certain value exists in a list?

a) value in list b) list.contains(value) c) list.includes(value) d) list.exists(value)

What will the following code print?

for i in range(5): print(i \* "\*")

​

a) Prints the numbers from 0 to 4 b) Prints a pattern of asterisks c) Prints a pattern of numbers d) Generates an error

What does the len() function return for a dictionary?

a) The number of keys in the dictionary b) The number of values in the dictionary c) The sum of the lengths of all keys and values d) The total number of items in the dictionary

What is the purpose of a return statement in a function?

a) It terminates the function b) It outputs a value from the function c) It defines a loop d) It defines a condition

How do you reverse a string "Python"?

a) "Python"[::-1] b) reverse("Python") c) reverse\_string("Python")

d) "Python".reverse()

Which of the following is a valid way to add a key-value pair to a dictionary?

a) dict.add(key, value) b) dict[key] = value c) dict.insert(key, value)

d) dict.append(key, value)

What is the output of print(3 \* 5)?

a) 8 b) 15 c) 53 d) 15

What does the following code snippet do?

num = 10 while num > 0: print(num) num -= 2

​

a) Prints even numbers from 10 to 0 b) Prints odd numbers from 10 to 0 c) Prints numbers from 10 to 0 in steps of 2 d) Generates an error

Write nested loops to print the following pattern:

\* \*\* \*\*\* \*\*\*\*

​

a) This pattern cannot be achieved using nested loops. b) It will print a different pattern. c) The correct nested loops are included below this question. d) It depends on the input provided to the loops.

Which of the following is NOT a valid variable name in Python?

a) my\_var b) variable\_name c) 123\_variable d) \_underscore

What does the input() function do in Python?

a) Displays output on the screen b) Takes user input from the keyboard c) Converts data types d) Performs mathematical calculations

What will the following code snippet print?

for i in range(3): for j in range(i+1): print("\*", end="") print()

​

a) Prints a pattern of asterisks: \*, \*\*, \*\*\* b) Prints a pattern of asterisks: \*\*\*, \*\*, \* c) Prints a pattern of asterisks: \*, \*, \* d) Generates an error

Which of the following is an example of a mutable data type?

a) String b) Integer c) Float d) List

What is the purpose of the elif statement in Python?

a) It defines a loop b) It handles exceptions c) It follows the else statement d) It provides an alternative condition to check

What will the following code snippet output

sentence = "Python programming is fun" words = sentence.split() print(len(words))

​

How do you access the value associated with the key "age" in a dictionary person?

a) `person("age")

b)person["age"]

c)person.get("age")

d)person.value("age")`

What does the break statement do in a loop?

a) It continues to the next iteration of the loop b) It exits the loop immediately c) It skips the current iteration and moves to the next one d) It raises an error

What is the result of 10 / 3 in Python?

a) 3.3333 b) 3.0 c) 3 d) 3.333

Write nested loops to print the following pattern:

\* \*\*\* \*\*\*\*\* \*\*\*\*\*\*\*

SHIVA625

OOP

**Assignment on Classes and Objects! Complete the following code.**

Python

Copy

class Student:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

self.grades = []

def add\_grades(self, grade):

"""

Add a grade to the list of grades for the student.

Args:

grade (float): The grade to be added to the list.

"""

def calculate\_average(self):

"""

Calculate the average of all grades in the list.

Returns:

float: The average grade.

"""

def display\_student\_info(self):

"""

Display student information, including name, age, and average grade.

"""

# Example usage of the Student class:

student1 = Student("Dodagatta Nihar", 19)

student1.add\_grades(90)

student1.add\_grades(85)

student1.add\_grades(92)

student1.display\_student\_info()

### Thank You! S*hiva.625*