Python Basic Question Bank

Question 1: Write a Python program to print "Hello, World!".

In [ ]:
```
print("Hello, World!")
```

```
Hello, World!
```

Question 2: Write a Python program that takes a user input and prints it.

In [ ]:
```
user = input("Please enter something: ")
print(f"You entered: {user}")
```

```
Please enter something: sanad singh
You entered: sanad singh
```

Question 3: Write a Python program to check if a number is positive, negative, or zero.

In [ ]:
```
number = float(input("Enter a number: "))
if number > 0:
    print("The number is positive.")
elif number < 0:
    print("The number is negative.")
else:
    print("The number is zero.")
```

```
Enter a number: 10
The number is positive.
```

In [ ]:

```
number = float(input("Enter a number: "))
if number > 0:
    print("The number is positive.")
elif number < 0:
    print("The number is negative.")
else:
    print("The number is zero.")
```

```
Enter a number: -30
The number is negative.
```

In [ ]:

```
number = float(input("Enter a number: "))
if number > 0:
    print("The number is positive.")
elif number < 0:
    print("The number is negative.")
else:
    print("The number is zero.")
```

```
Enter a number: 0
The number is zero.
```

Question 4: Write a Python program to find the largest of three numbers.

In [ ]:

```
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
num3 = float(input("Enter third number: "))

if (num1 >= num2) and (num1 >= num3):
    largest = num1
elif (num2 >= num1) and (num2 >= num3):
    largest = num2
else:
    largest = num3

print(f"The largest number is {largest}")
```

```
Enter first number: 10
Enter second number: 50
Enter third number: 14
The largest number is 50.0
```

Question 5: Write a Python program to calculate the factorial of a number.

In [ ]:
```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)

num = int(input("Enter a number: "))
print(f"The factorial of {num} is {factorial(num)}")
```

```
Enter a number: 5
The factorial of 5 is 120
```

Variables and Data Types

Question 6: Create variables of different data types: integer, float, string, and boolean. Print their values and types.

In [ ]:
```
integer = 10
float_var = 10.5
string= "Hello"
boolean= True

print(f"Integer value: {integer}, type: {type(integer)}")
print(f"Float value: {float_var}, type: {type(float_var)}")
print(f"String value: {string}, type: {type(string)}")
print(f"Boolean value: {boolean}, type: {type(boolean)}")
```

```
Integer value: 10, type: <class 'int'>
Float value: 10.5, type: <class 'float'>
String value: Hello, type: <class 'str'>
Boolean value: True, type: <class 'bool'>
```

Question 7: Write a Python program to swap the values of two variables.

In [ ]:
```
a = 5
b = 10
```

```
print(f"Before swap: a = {a}, b = {b}")

# Swapping
a, b = b, a
print(f"After swap: a = {a}, b = {b}")
```

```
Before swap: a = 5, b = 10
After swap: a = 10, b = 5
```

Question 8: Write a Python program to convert Celsius to Fahrenheit.

In [ ]:
```
celsius = float(input("Enter temperature in Celsius: "))
fahrenheit = (celsius * 9/5) + 32
print(f"{celsius}°C is equal to {fahrenheit}°F")
```

```
Enter temperature in Celsius: 2.2
2.2°C is equal to 35.96°F
```

Question 9: Write a Python program to concatenate two strings.

In [ ]:
```
string1 = "Hello"
string2 = "World"
concatenated_string = string1 + " " + string2
print(concatenated_string)
```

```
Hello World
```

Question 10: Write a Python program to check if a variable is of a specific data type.

In [ ]:
```
var = 10.5
if isinstance(var, float):
    print(f"{var} is a float")
else:
    print(f"{var} is not a float")
```

```
10.5 is a float
```

Basic Operators (Arithmetic, Comparison, Logical)

Question 11: Write a Python program to perform arithmetic operations: addition, subtraction, multiplication, and division.

```
In [ ]: a = 5
        b = 3

        print(f"Addition: {a} + {b} = {a + b}")
        print(f"Subtraction: {a} - {b} = {a - b}")
        print(f"Multiplication: {a} * {b} = {a * b}")
        print(f"Division: {a} / {b} = {a / b}")
```

```
Addition: 5 + 3 = 8
Subtraction: 5 - 3 = 2
Multiplication: 5 * 3 = 15
Division: 5 / 3 = 1.6666666666666667
```

Question 12: Write a Python program to demonstrate comparison operators: equal to, not equal to, greater than, less than.

```
In [ ]: a =int(input("Enter the Value: "))
        b =int(input("Enter the Value: "))

        print(f"{a} == {b}: {a == b}")
        print(f"{a} != {b}: {a != b}")
        print(f"{a} > {b}: {a > b}")
        print(f"{a} < {b}: {a < b}")
```

```
Enter the Value: 10
Enter the Value: 5
10 == 5: False
10 != 5: True
10 > 5: True
10 < 5: False
```

Question 13: Write a Python program to demonstrate logical operators: and, or, not.

```
In [ ]: a = True
        b = False

        print(f"True and False: {a and b}")
        print(f"True or False: {a or b}")
        print(f"not True: {not a}")
```

```
True and False: False
True or False: True
not True: False
```

Question 14: Write a Python program to calculate the square of a number.

```
In [ ]: num = float(input("Enter a number: "))
        square = num ** 2
        print(f"The square of {num} is {square}")
```

```
Enter a number: 100
The square of 100.0 is 10000.0
```

```
In [ ]: num = float(input("Enter a number: "))
        square = num ** 2
        print(f"The square of {num} is {square}")
```

```
Enter a number: 25
The square of 25.0 is 625.0
```

Question 15: Write a Python program to check if a number is even or odd.

```
In [ ]: num = int(input("Enter a number: "))
        if num % 2 == 0:
            print(f"{num} is even.")
        else:
            print(f"{num} is odd.")
```

```
Enter a number: 14
14 is even.
```

```
In [ ]: num = int(input("Enter a number: "))
        if num % 2 == 0:
            print(f"{num} is even.")
        else:
            print(f"{num} is odd.")
```

```
Enter a number: 27
27 is odd.
```

Question 16: Write a Python program to find the sum of the first n natural numbers.

```
In [ ]: n = int(input("Enter a number: "))
        sum_n = (n * (n + 1)) // 2
        print(f"The sum of the first {n} natural numbers is {sum_n}")
```

```
Enter a number: 5
The sum of the first 5 natural numbers is 15
```

In [ ]:
```
n = int(input("Enter a number: "))
sum_n = (n * (n + 1)) // 2
print(f"The sum of the first {n} natural numbers is {sum_n}")
```

```
Enter a number: 11
The sum of the first 11 natural numbers is 66
```

Question 17: Write a Python program to check if a year is a leap year.

In [ ]:
```
year = int(input("Enter a year: "))
if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
    print(f"{year} is a leap year.")
else:
    print(f"{year} is not a leap year.")
```

```
Enter a year: 4012
4012 is a leap year.
```

In [ ]:
```
year = int(input("Enter a year: "))
if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
    print(f"{year} is a leap year.")
else:
    print(f"{year} is not a leap year.")
```

```
Enter a year: 2023
2023 is not a leap year.
```

Question 18: Write a Python program to reverse a string.

In [ ]:
```
string = input("Enter a string: ")
reversed_string = string[::-1]
print(f"The reversed string is: {reversed_string}")
```

```
Enter a string: sanad singh
The reversed string is: hgnis danas
```

Question 19: Write a Python program to check if a string is a palindrome.

In [ ]:
```
string = input("Enter a string: ")
if string == string[::-1]:
    print(f"{string} is a palindrome.")
```

```
else:
    print(f"{string} is not a palindrome.")
```

```
Enter a string: samas
samas is a palindrome.
```

In [ ]:
```
string = input("Enter a string: ")
if string == string[::-1]:
    print(f"{string} is a palindrome.")
else:
    print(f"{string} is not a palindrome.")
```

```
Enter a string: sanad
sanad is not a palindrome.
```

Question 20: Write a Python program to sort a list of numbers in ascending order.

In [ ]:
```
numbers = [int(x) for x in input("Enter numbers separated by space: ").split()]
numbers.sort()
print(f"Sorted list: {numbers}")
```

```
Enter numbers separated by space: 1 2 3 4 5 6 7 8 9 10 9 8 7 6 5 4 3 2 1
Sorted list: [1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9, 10]
```

Conditional Statements

Question 21: Simple if Statement

Write a program that asks the user to input a number and prints whether the number is positive.

In [ ]:
```
number = float(input("Enter a number: "))
if number > 0:
    print("The number is positive.")
```

```
Enter a number: 2
The number is positive.
```

In [ ]:
```
number = float(input("Enter a number: "))
if number > 0:
    print("The number is positive.")
```

```
Enter a number: 3.2
The number is positive.
```

In [ ]:
```python
number = float(input("Enter a number: "))
if number > 0:
    print("The number is positive.")
```

```
Enter a number: -2
```

Question 22: if-else Statement

Write a program that asks the user to input a number and prints whether the number is positive or negative.

In [ ]:
```python
number = float(input("Enter a number: "))
if number > 0:
    print("The number is positive.")
else:
    print("The number is negative.")
```

```
Enter a number: 3
The number is positive.
```

In [ ]:
```python
number = float(input("Enter a number: "))
if number > 0:
    print("The number is positive.")
else:
    print("The number is negative.")
```

```
Enter a number: -2
The number is negative.
```

Question 23: if-elif-else Statement

Write a program that asks the user to input a number and prints whether the number is positive, negative, or zero.

In [ ]:
```python
number = float(input("Enter a number: "))
if number > 0:
    print("The number is positive.")
elif number < 0:
    print("The number is negative.")
else:
    print("The number is zero.")
```

```
Enter a number: 0
The number is zero.
```

Question 24: Nested if Statement

Write a program that asks the user to input a number and prints whether the number is positive and even, positive and odd, or negative.

In [ ]:
```
number = float(input("Enter a number: "))
if number > 0:
    if number % 2 == 0:
        print("The number is positive and even.")
    else:
        print("The number is positive and odd.")
else:
    print("The number is negative.")
```

```
Enter a number: 5
The number is positive and odd.
```

Question 25: for Loop

Write a program that prints all the numbers from 1 to 10 using a for loop.

In [ ]:
```
for i in range(1, 11):
    print(i)
```

```
1
2
3
4
5
6
7
8
9
10
```

Question 26: while Loop

Write a program that prints all the numbers from 1 to 10 using a while loop.

In [ ]:
```
i = 1
while i <= 10:
    print(i)
    i += 1
```

```
1
2
3
4
5
6
7
8
9
10
```

Question 27: Nested Loops

Write a program that prints a 5x5 grid of asterisks (*) using nested loops.

In [ ]:
```
for i in range(5):
    for j in range(5):
        print("*", end=" ")
    print()
```

```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

Question 28: break Statement

Write a program that asks the user to input numbers until they input 0. The program should print the sum of all the input numbers.

In [ ]:
```
total = 0
while True:
    number = float(input("Enter a number (0 to stop): "))
    if number == 0:
        break
    total += number
print(f"The sum of all the numbers is {total}.")
```

```
Enter a number (0 to stop): 2
Enter a number (0 to stop): 9
Enter a number (0 to stop): 4
Enter a number (0 to stop): 3
Enter a number (0 to stop): 0
The sum of all the numbers is 18.0.
```

Question 29: continue Statement

Write a program that prints all the numbers from 1 to 10 except 5 using a for loop and continue statement.

In [ ]:
```
for i in range(1, 11):
    if i == 5:
        continue
    print(i)
```

```
1
2
3
4
6
7
8
9
10
```

Question 30: pass Statement

Write a program that defines an empty function using the pass statement.

In [ ]:
```
def empty_function():
    pass

# Calling the empty function
empty_function()
```

Question 31: Combining Loops and Conditionals

Write a program that asks the user to input a number and prints all the even numbers from 1 to that number using a for loop.

In [ ]:
```
number = int(input("Enter a number: "))
for i in range(1, number + 1):
    if i % 2 == 0:
        print(i)
```

```
Enter a number: 6
2
4
6
```

In [ ]:
```
number = int(input("Enter a number: "))
for i in range(1, number + 1):
    if i % 2 == 0:
        print(i)
```

```
Enter a number: 16
2
4
6
8
10
12
14
16
```

Question 32: Factorial Calculation

Write a program that calculates the factorial of a number input by the user using a while loop.

In [ ]:
```
number = int(input("Enter a number: "))
factorial = 1
i = 1
while i <= number:
    factorial *= i
    i += 1
print(f"The factorial of {number} is {factorial}.")
```

```
Enter a number: 6
The factorial of 6 is 720.
```

Question 33: Sum of Digits

Write a program that calculates the sum of the digits of a number input by the user using a while loop.

In [ ]:
```
number = int(input("Enter a number: "))
sum_of_digits = 0
while number > 0:
    digit = number % 10
    sum_of_digits += digit
    number = number // 10
print(f"The sum of the digits is {sum_of_digits}.")
```

```
Enter a number: 6
The sum of the digits is 6.
```

In [ ]:
```
number = int(input("Enter a number: "))
sum_of_digits = 0
while number > 0:
    digit = number % 10
    sum_of_digits += digit
    number = number // 10
print(f"The sum of the digits is {sum_of_digits}.")
```

```
Enter a number: 9
The sum of the digits is 9.
```

Question 34: Prime Number Check

Write a program that checks if a number input by the user is a prime number using a for loop.

In [ ]:
```
number = int(input("Enter a number: "))
is_prime = True
if number <= 1:
    is_prime = False
else:
    for i in range(2, int(number ** 0.5) + 1):
        if number % i == 0:
            is_prime = False
            break
if is_prime:
    print(f"{number} is a prime number.")
```

```
else:
    print(f"{number} is not a prime number.")
```

```
Enter a number: 23
23 is a prime number.
```

In [ ]:

```
number = int(input("Enter a number: "))
is_prime = True
if number <= 1:
    is_prime = False
else:
    for i in range(2, int(number ** 0.5) + 1):
        if number % i == 0:
            is_prime = False
            break
if is_prime:
    print(f"{number} is a prime number.")
else:
    print(f"{number} is not a prime number.")
```

```
Enter a number: 24
24 is not a prime number.
```

Question 35: Fibonacci Sequence

Write a program that prints the first n Fibonacci numbers, where n is input by the user.

In [ ]:

```
n = int(input("Enter the number of Fibonacci numbers to print: "))
a, b = 0, 1
count = 0
while count < n:
    print(a)
    a, b = b, a + b
    count += 1
```

```
Enter the number of Fibonacci numbers to print: 9
0
1
1
2
3
5
8
13
21
```

Data Structures Assignments

Lists

Question 36: Creating and Accessing Lists

Create a list of the first 20 positive integers. Print the list.

In [ ]:
```
lst = list(range(1, 21))
print(lst)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```

Question 37: Accessing List Elements

Print the first, middle, and last elements of the list created in Question 36.

In [ ]:
```
print(f"First element: {lst[0]}")
print(f"Middle element: {lst[len(lst) // 2]}")
print(f"Last element: {lst[-1]}")
```

```
First element: 1
Middle element: 11
Last element: 20
```

Question 38: List Slicing

Print the first five elements, the last five elements, and the elements from index 5 to 15 of the list created in Question 36.

In [ ]:
```
print(f"First five elements: {lst[:5]}")
print(f"Last five elements: {lst[-5:]}")
print(f"Elements from index 5 to 15: {lst[5:16]}")
```

```
First five elements: [1, 2, 3, 4, 5]
Last five elements: [16, 17, 18, 19, 20]
Elements from index 5 to 15: [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
```

Question 39: List Comprehensions

Create a new list containing the squares of the first 10 positive integers using a list comprehension. Print the new list.

In [ ]:
```
squares = [x**2 for x in range(1, 11)]
print(squares)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Question 40: Filtering Lists

Create a new list containing only the even numbers from the list created in Question 36 using a list comprehension. Print the new list.

In [ ]:
```
evens = [x for x in lst if x % 2 == 0]
print(evens)
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

Question 41: List Methods

Create a list of random numbers and sort it in ascending and descending order. Remove the duplicates from the list and print the modified list.

In [ ]:
```
import random

random_numbers = [random.randint(1, 20) for _ in range(15)]
print(f"Original list: {random_numbers}")

sorted_numbers = sorted(random_numbers)
print(f"Sorted in ascending order: {sorted_numbers}")

sorted_numbers_desc = sorted(random_numbers, reverse=True)
```

```
print(f"Sorted in descending order: {sorted_numbers_desc}")

unique_numbers = list(set(random_numbers))
print(f"List with duplicates removed: {unique_numbers}")
```

```
Original list: [12, 7, 13, 18, 7, 16, 3, 7, 6, 19, 5, 18, 11, 6, 9]
Sorted in ascending order: [3, 5, 6, 6, 7, 7, 7, 9, 11, 12, 13, 16, 18, 18, 19]
Sorted in descending order: [19, 18, 18, 16, 13, 12, 11, 9, 7, 7, 7, 6, 6, 5, 3]
List with duplicates removed: [3, 5, 6, 7, 9, 11, 12, 13, 16, 18, 19]
```

Question 42: Nested Lists

Create a nested list representing a 3x3 matrix and print the matrix. Access and print the element at the second row and third column.

In [ ]:

```
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
print("Matrix:")
for row in matrix:
    print(row)
print(f"Element at second row and third column: {matrix[1][2]}")
```

```
Matrix:
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
Element at second row and third column: 6
```

Question 43: List of Dictionaries

Create a list of dictionaries where each dictionary represents a student with keys 'name' and 'score'. Sort the list of dictionaries by the 'score' in descending order and print the sorted list.

In [ ]:

```
students = [
    {'name': 'Sanad', 'score': 88},
    {'name': 'Raj', 'score': 72},
    {'name': 'Honey', 'score': 95},
    {'name': 'David', 'score': 65},
    {'name': 'Akash', 'score': 78}
]
sorted_students = sorted(students, key=lambda x: x['score'], reverse=True)
print("Sorted students by score in descending order:")
for student in sorted_students:
    print(student)
```

```
Sorted students by score in descending order:
{'name': 'Honey', 'score': 95}
{'name': 'Sanad', 'score': 88}
{'name': 'Akash', 'score': 78}
{'name': 'Raj', 'score': 72}
{'name': 'David', 'score': 65}
```

Question 44: Matrix Transposition

Write a function that takes a 3x3 matrix (nested list) as input and returns its transpose. Print the original and transposed matrices.

In [ ]:
```
def transpose_matrix(matrix):
    transposed = [[matrix[j][i] for j in range(len(matrix))] for i in range(len(matrix[0]))]
    return transposed

matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
transposed = transpose_matrix(matrix)
print("Original matrix:")
for row in matrix:
    print(row)
print("Transposed matrix:")
for row in transposed:
    print(row)
```

```
Original matrix:
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
Transposed matrix:
[1, 4, 7]
[2, 5, 8]
[3, 6, 9]
```

Question 45: Flattening a Nested List

Write a function that takes a nested list and flattens it into a single list. Print the original and flattened lists.

In [ ]:
```
def flatten_list(nested_list):
    flat_list = [item for sublist in nested_list for item in sublist]
    return flat_list

nested_list = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
flattened = flatten_list(nested_list)
print("Original nested list:")
print(nested_list)
print("Flattened list:")
print(flattened)
```

```
Original nested list:
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
Flattened list:
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Question 46: List Manipulation

Create a list of the first 10 positive integers. Remove the elements at indices 2, 4, and 6, and insert the element '99' at index 5. Print the modified list.

In [ ]:
```
lst = list(range(1, 11))
print(f"Original list: {lst}")
del lst[6]
del lst[4]
del lst[2]
lst.insert(5, 99)
print(f"Modified list: {lst}")
```

```
Original list: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Modified list: [1, 2, 4, 6, 8, 99, 9, 10]
```

Question 47: List Zipping

Create two lists of the same length. Use the `zip` function to combine these lists into a list of tuples and print the result.

In [ ]:
```python
list1 = [1, 2, 3, 4, 5]
list2 = ['a', 'b', 'c', 'd', 'e']
zipped = list(zip(list1, list2))
print(zipped)
```

```
[(1, 'a'), (2, 'b'), (3, 'c'), (4, 'd'), (5, 'e')]
```

Question 48: List Reversal

Write a function that takes a list and returns a new list with the elements in reverse order. Print the original and reversed lists.

In [ ]:
```python
def reverse_list(lst):
    return lst[::-1]

original_list = [1, 2, 3, 4, 5]
reversed_list = reverse_list(original_list)
print(f"Original list: {original_list}")
print(f"Reversed list: {reversed_list}")
```

```
Original list: [1, 2, 3, 4, 5]
Reversed list: [5, 4, 3, 2, 1]
```

Question 49: List Rotation

Write a function that rotates a list by n positions. Print the original and rotated lists.

In [ ]:
```python
def rotate_list(lst, n):
    return lst[n:] + lst[:n]

original_list = [1, 2, 3, 4, 5]
rotated_list = rotate_list(original_list, 2)
print(f"Original list: {original_list}")
print(f"Rotated list: {rotated_list}")
```

```
Original list: [1, 2, 3, 4, 5]
Rotated list: [3, 4, 5, 1, 2]
```

Question 50: List Intersection

Write a function that takes two lists and returns a new list containing only the elements that are present in both lists. Print the intersected list.

In [ ]:
```
def list_intersection(lst1, lst2):
    return [x for x in lst1 if x in lst2]

list1 = [1, 2, 3, 4, 5]
list2 = [3, 4, 5, 6, 7]
intersection = list_intersection(list1, list2)
print(f"List 1: {list1}")
print(f"List 2: {list2}")
print(f"Intersection: {intersection}")
```

```
List 1: [1, 2, 3, 4, 5]
List 2: [3, 4, 5, 6, 7]
Intersection: [3, 4, 5]
```

Tuples

Question 51: Creating and Accessing Tuples

Create a tuple with the first 10 positive integers. Print the tuple.

In [ ]:
```
tpl = tuple(range(1, 11))
print(tpl)
```

```
(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

Question 52: Accessing Tuple Elements

Print the first, middle, and last elements of the tuple created in Question 51.

In [ ]:
```
print(f"First element: {tpl[0]}")
print(f"Middle element: {tpl[len(tpl) // 2]}")
print(f"Last element: {tpl[-1]}")
```

```
First element: 1
Middle element: 6
Last element: 10
```

Question 53: Tuple Slicing

Print the first three elements, the last three elements, and the elements from index 2 to 5 of the tuple created in Question 51.

In [ ]:
```
print(f"First three elements: {tpl[:3]}")
print(f"Last three elements: {tpl[-3:]}")
print(f"Elements from index 2 to 5: {tpl[2:6]}")
```

```
First three elements: (1, 2, 3)
Last three elements: (8, 9, 10)
Elements from index 2 to 5: (3, 4, 5, 6)
```

Question 54: Nested Tuples

Create a nested tuple representing a 3x3 matrix and print the matrix. Access and print the element at the second row and third column.

In [ ]:
```
matrix = (
    (1, 2, 3),
    (4, 5, 6),
    (7, 8, 9)
)
print("Matrix:")
for row in matrix:
    print(row)
print(f"Element at second row and third column: {matrix[1][2]}")
```

```
Matrix:
(1, 2, 3)
(4, 5, 6)
(7, 8, 9)
Element at second row and third column: 6
```

Question 55: Tuple Concatenation

Concatenate two tuples: (1, 2, 3) and (4, 5, 6). Print the resulting tuple.

In [ ]:
```
tpl1 = (1, 2, 3)
tpl2 = (4, 5, 6)
concatenated = tpl1 + tpl2
print(concatenated)
```

```
(1, 2, 3, 4, 5, 6)
```

Question 56: Tuple Methods

Create a tuple with duplicate elements and count the occurrences of an element. Find the index of the first occurrence of an element in the tuple.

In [ ]:
```
tpl = (1, 2, 2, 3, 4, 4, 4, 5)
print(f"Occurrences of 4: {tpl.count(4)}")
print(f"Index of first occurrence of 2: {tpl.index(2)}")
```

```
Occurrences of 4: 3
Index of first occurrence of 2: 1
```

Question 57: Unpacking Tuples

Create a tuple with 5 elements and unpack it into 5 variables. Print the variables.

In [ ]:
```
tpl = (1, 2, 3, 4, 5)
a, b, c, d, e = tpl
print(a, b, c, d, e)
```

```
1 2 3 4 5
```

Question 58: Tuple Conversion

Convert a list of the first 5 positive integers to a tuple. Print the tuple.

In [ ]:
```
lst = [1, 2, 3, 4, 5]
tpl = tuple(lst)
print(tpl)
```

```
(1, 2, 3, 4, 5)
```

Question 59: Tuple of Tuples

Create a tuple containing 3 tuples, each with 3 elements. Print the tuple of tuples.

```
In [ ]: tpl_of_tpls = (
            (1, 2, 3),
            (4, 5, 6),
            (7, 8, 9)
        )
        print(tpl_of_tpls)
```

```
((1, 2, 3), (4, 5, 6), (7, 8, 9))
```

Question 60: Tuple and List

Create a tuple with the first 5 positive integers. Convert it to a list, append the number 6, and convert it back to a tuple. Print the resulting tuple.

```
In [ ]: tpl = (1, 2, 3, 4, 5)
        lst = list(tpl)
        lst.append(6)
        tpl = tuple(lst)
        print(tpl)
```

```
(1, 2, 3, 4, 5, 6)
```

Question 61: Tuple and String

Create a tuple with the characters of a string. Join the tuple elements into a single string. Print the string.

```
In [ ]: string = "hello"
        tpl = tuple(string)
        joined_string = ''.join(tpl)
        print(joined_string)
```

```
hello
```

Question 62: Tuple and Dictionary

Create a dictionary with tuple keys and integer values. Print the dictionary.

```
In [ ]: tpl_dict = {
            (1, 2): 3,
    (4, 5): 6,
    (7, 8): 9
}
print(tpl_dict)
```

```
{(1, 2): 3, (4, 5): 6, (7, 8): 9}
```

Question 63: Nested Tuple Iteration

Create a nested tuple and iterate over the elements, printing each element.

In [ ]:
```
nested_tpl = (
    (1, 2, 3),
    (4, 5, 6),
    (7, 8, 9)
)
for tpl in nested_tpl:
    for elem in tpl:
        print(elem)
```

```
1
2
3
4
5
6
7
8
9
```

Question 64: Tuple and Set

Create a tuple with duplicate elements. Convert it to a set to remove duplicates and print the resulting set.

In [ ]:
```
tpl = (1, 2, 2, 3, 4, 4, 4, 5)
unique_set = set(tpl)
print(unique_set)
```

```
{1, 2, 3, 4, 5}
```

Question 65: Tuple Functions

Write functions that take a tuple and return the minimum, maximum, and sum of the elements. Print the results for a sample tuple.

In [ ]:
```
def min_in_tuple(tpl):
    return min(tpl)

def max_in_tuple(tpl):
    return max(tpl)

def sum_of_tuple(tpl):
    return sum(tpl)

sample_tpl = (1, 2, 3, 4, 5)
print(f"Minimum: {min_in_tuple(sample_tpl)}")
print(f"Maximum: {max_in_tuple(sample_tpl)}")
print(f"Sum: {sum_of_tuple(sample_tpl)}")
```

```
Minimum: 1
Maximum: 5
Sum: 15
```

Sets

Question 66: Creating and Accessing Sets

Create a set with the first 10 positive integers. Print the set.

In [ ]:
```
s = set(range(1, 11))
print(s)
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

Question 67: Adding and Removing Elements

Add the number 11 to the set created in Assignment 1. Then remove the number 1 from the set. Print the modified set.

In [ ]:
```
s.add(11)
s.remove(1)
print(s)
```

```
{2, 3, 4, 5, 6, 7, 8, 9, 10, 11}
```

Question 68: Set Operations

Create two sets: one with the first 5 positive integers and another with the first 5 even integers. Perform and print the results of union, intersection, difference, and symmetric difference operations on these sets.

In [ ]:
```
set1 = set(range(1, 6))
set2 = set(range(2, 11, 2))
print(f"Union: {set1 | set2}")
print(f"Intersection: {set1 & set2}")
print(f"Difference (set1 - set2): {set1 - set2}")
print(f"Symmetric Difference: {set1 ^ set2}")
```

```
Union: {1, 2, 3, 4, 5, 6, 8, 10}
Intersection: {2, 4}
Difference (set1 - set2): {1, 3, 5}
Symmetric Difference: {1, 3, 5, 6, 8, 10}
```

Question 69: Set Comprehensions

Create a new set containing the squares of the first 10 positive integers using a set comprehension. Print the new set.

In [ ]:
```
squares = {x**2 for x in range(1, 11)}
print(squares)
```

```
{64, 1, 4, 36, 100, 9, 16, 49, 81, 25}
```

Question 70: Filtering Sets

Create a new set containing only the even numbers from the set created in Assignment 1 using a set comprehension. Print the new set.

In [ ]:
```
evens = {x for x in s if x % 2 == 0}
print(evens)
```

```
{2, 4, 6, 8, 10}
```

Question 71: Set Methods

Create a set with duplicate elements and remove the duplicates using set methods. Print the modified set.

```
In [ ]: s = {1, 2, 2, 3, 4, 4, 5}
        unique_s = set(s)
        print(unique_s)
```

```
{1, 2, 3, 4, 5}
```

Question 72: Subsets and Supersets

Create two sets: one with the first 5 positive integers and another with the first 3 positive integers. Check if the second set is a subset of the first set and if the first set is a superset of the second set. Print the results.

```
In [ ]: set1 = set(range(1, 6))
        set2 = set(range(1, 4))
        print(f"Is set2 a subset of set1? {set2.issubset(set1)}")
        print(f"Is set1 a superset of set2? {set1.issuperset(set2)}")
```

```
Is set2 a subset of set1? True
Is set1 a superset of set2? True
```

Question 73: Frozenset

Create a frozenset with the first 5 positive integers. Print the frozenset.

```
In [ ]: fs = frozenset(range(1, 6))
        print(fs)
```

```
frozenset({1, 2, 3, 4, 5})
```

Question 74: Set and List Conversion

Create a set with the first 5 positive integers. Convert it to a list, append the number 6, and convert it back to a set. Print the resulting set.

```
In [ ]: s = set(range(1, 6))
        lst = list(s)
        lst.append(6)
        s = set(lst)
        print(s)
```

```
{1, 2, 3, 4, 5, 6}
```

Question 75: Set and Dictionary

Create a dictionary with set keys and integer values. Print the dictionary.

In [ ]:
```
d = {
    frozenset({1, 2}): 3,
    frozenset({3, 4}): 7,
    frozenset({5, 6}): 11
}
print(d)
```

```
{frozenset({1, 2}): 3, frozenset({3, 4}): 7, frozenset({5, 6}): 11}
```

Question 76: Iterating Over Sets

Create a set and iterate over the elements, printing each element.

In [ ]:
```
s = set(range(1, 6))
for elem in s:
    print(elem)
```

```
1
2
3
4
5
```

Question 77: Removing Elements from Sets

Create a set and remove elements from it until it is empty. Print the set after each removal.

In [ ]:
```
s = set(range(1, 6))
while s:
    s.pop()
    print(s)
```

```
{2, 3, 4, 5}
{3, 4, 5}
{4, 5}
{5}
set()
```

Question 78: Set Symmetric Difference Update

Create two sets and update the first set with the symmetric difference of the two sets. Print the modified first set.

In [ ]:
```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
set1.symmetric_difference_update(set2)
print(set1)
```

```
{1, 2, 4, 5}
```

Question 79: Set Membership Testing

Create a set and test if certain elements are present in the set. Print the results.

In [ ]:
```
s = set(range(1, 6))
print(3 in s)
print(6 in s)
```

```
True
False
```

Question 80: Set of Tuples

Create a set containing tuples, where each tuple contains two elements. Print the set.

In [ ]:
```
s = { (1, 2), (3, 4), (5, 6) }
print(s)
```

```
{(1, 2), (3, 4), (5, 6)}
```

Dictionaries

Question 81: Creating and Accessing Dictionaries

Create a dictionary with the first 10 positive integers as keys and their squares as values. Print the dictionary.

In [ ]:
```python
d = {i: i**2 for i in range(1, 11)}
print(d)
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
```

Question 82: Accessing Dictionary Elements

Print the value of the key 5 and the keys of the dictionary created in Question 81.

In [ ]:
```python
print(f"Value of key 5: {d[5]}")
print(f"Keys: {list(d.keys())}")
```

```
Value of key 5: 25
Keys: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Question 83: Dictionary Methods

Add a new key-value pair (11, 121) to the dictionary created in Question 81 and then remove the key-value pair with key 1. Print the modified dictionary.

In [ ]:
```python
d[11] = 121
d.pop(1)
print(d)
```

```
{2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100, 11: 121}
```

Question 84: Iterating Over Dictionaries

Iterate over the dictionary created in Question 81 and print each key-value pair.

In [ ]:
```python
for key, value in d.items():
    print(f"{key}: {value}")
```

```
2: 4
3: 9
4: 16
5: 25
6: 36
7: 49
8: 64
9: 81
10: 100
11: 121
```

Question 85: Dictionary Comprehensions

Create a new dictionary containing the cubes of the first 10 positive integers using a dictionary comprehension. Print the new dictionary.

In [ ]:
```
cubes = {i: i**3 for i in range(1, 11)}
print(cubes)
```

```
{1: 1, 2: 8, 3: 27, 4: 64, 5: 125, 6: 216, 7: 343, 8: 512, 9: 729, 10: 1000}
```

Question 86: Merging Dictionaries

Create two dictionaries: one with keys as the first 5 positive integers and values as their squares, and another with keys as the next 5 positive integers and values as their squares. Merge these dictionaries into a single dictionary and print it.

In [ ]:
```
d1 = {i: i**2 for i in range(1, 6)}
d2 = {i: i**2 for i in range(6, 11)}
d1.update(d2)
print(d1)
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
```

Question 87: Nested Dictionaries

Create a nested dictionary representing a student with keys 'name', 'age', 'grades', where 'grades' is another dictionary with keys 'math', 'science', and 'english'. Print the nested dictionary.

In [ ]:
```
student = {
    'name': 'sanad singh',
    'age': 24,
    'grades': {
        'math': 83,
        'science': 79,
        'english': 88
    }
}
print(student)
```

```
{'name': 'sanad singh', 'age': 24, 'grades': {'math': 83, 'science': 79, 'english': 88}}
```

Question 88: Dictionary of Lists

Create a dictionary where the keys are the first 5 positive integers and the values are lists containing the first 5 multiples of the key. Print the dictionary.

In [ ]:
```
multiples = {i: [i * j for j in range(1, 6)] for i in range(1, 6)}
print(multiples)
```

```
{1: [1, 2, 3, 4, 5], 2: [2, 4, 6, 8, 10], 3: [3, 6, 9, 12, 15], 4: [4, 8, 12, 16, 20], 5: [5, 10, 15, 20, 25]}
```

Question 89: Dictionary of Tuples

Create a dictionary where the keys are the first 5 positive integers and the values are tuples containing the key and its square. Print the dictionary.

In [ ]:
```
tuple_dict = {i: (i, i**2) for i in range(1, 6)}
print(tuple_dict)
```

```
{1: (1, 1), 2: (2, 4), 3: (3, 9), 4: (4, 16), 5: (5, 25)}
```

Question 90: Dictionary and List Conversion

Create a dictionary with the first 5 positive integers as keys and their squares as values. Convert the dictionary to a list of tuples and print it.

In [ ]:
```
d = {i: i**2 for i in range(1, 6)}
list_of_tuples = list(d.items())
print(list_of_tuples)
```

```
[(1, 1), (2, 4), (3, 9), (4, 16), (5, 25)]
```

Question 91: Dictionary Filtering

Create a dictionary with the first 10 positive integers as keys and their squares as values. Create a new dictionary containing only the key-value pairs where the key is even. Print the new dictionary.

In [ ]:
```
d = {i: i**2 for i in range(1, 11)}
even_dict = {k: v for k, v in d.items() if k % 2 == 0}
print(even_dict)
```

```
{2: 4, 4: 16, 6: 36, 8: 64, 10: 100}
```

Question 92: Dictionary Key and Value Transformation

Create a dictionary with the first 5 positive integers as keys and their squares as values. Create a new dictionary with keys and values swapped. Print the new dictionary.

In [ ]:
```
d = {i: i**2 for i in range(1, 6)}
swapped_dict = {v: k for k, v in d.items()}
print(swapped_dict)
```

```
{1: 1, 4: 2, 9: 3, 16: 4, 25: 5}
```

Question 93: Default Dictionary

Create a default dictionary where each key has a default value of an empty list. Add some elements to the lists and print the dictionary.

In [ ]:
```
from collections import defaultdict

default_dict = defaultdict(list)
default_dict['a'].append(1)
default_dict['a'].append(2)
default_dict['b'].append(3)
print(default_dict)
```

```
defaultdict(<class 'list'>, {'a': [1, 2], 'b': [3]})
```

Question 94: Counting with Dictionaries

Write a function that takes a string and returns a dictionary with the count of each character in the string. Print the dictionary.

In [ ]:
```
def count_chars(s):
    count_dict = {}
    for char in s:
        count_dict[char] = count_dict.get(char, 0) + 1
    return count_dict

string = "hello world"
print(count_chars(string))
```

```
{'h': 1, 'e': 1, 'l': 3, 'o': 2, ' ': 1, 'w': 1, 'r': 1, 'd': 1}
```

Question 95: Dictionary and JSON

Create a dictionary representing a book with keys 'title', 'author', 'year', and 'genre'. Convert the dictionary to a JSON string and print it.

In [121…
```
import json

book = {
    'title': 'To Kill Mr Fury',
    'author': 'xyz',
    'year': 2050,
    'genre': 'Fiction'
}
book_json = json.dumps(book)
print(book_json)
```

```
{"title": "To Kill Mr Fury", "author": "xyz", "year": 2050, "genre": "Fiction"}
```

Functions

Defining Functions

Question 96: Simple Function

Define a function that takes a single integer as input and returns its square. Test the function with different inputs.

In [122…

```python
def square(x):
    return x ** 2

# Test
print(square(2))  # 4
print(square(5))  # 25
```

```
4
25
```

Question 97: Multiple Arguments

Define a function that takes two integers as input and returns their sum. Test the function with different inputs.

In [123…

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

# Test
print(add(2, 3))  # 5
print(add(10, 20))  # 30
```

```
5
30
```

Question 98: Default Arguments

Define a function that takes two integers as input and returns their sum. The second integer should have a default value of 5. Test the function with different inputs.

In [124…

```python
def add(x, y=5):
    return x + y

# Test
print(add(2))  # 7
print(add(10, 20))  # 30
```

```
7
30
```

Question 99: Keyword Arguments

Define a function that takes three named arguments: first_name, last_name, and age, and returns a formatted string. Test the function with different inputs.

In [126…

```python
def format_person(first_name, last_name, age):
    return f"{first_name} {last_name} is {age} years old."
print(format_person(first_name="Sanad", last_name="Singh", age=30))
print(format_person(age=24, first_name="Shiv", last_name="Singh"))
```

```
Sanad Singh is 30 years old.
Shiv Singh is 24 years old.
```

Question 100: Variable-length Arguments

Define a function that takes a variable number of integer arguments and returns their product. Test the function with different inputs.

In [127…

```python
def multiply(*args):
    product = 1
    for num in args:
        product *= num
    return product

# Test
print(multiply(2, 3, 4))  # 24
print(multiply(1, 2, 3, 4, 5))  # 120
```

```
24
120
```

Question 101: Nested Functions

Define a function that contains another function inside it. The outer function should take two integers as input and return the result of the inner function, which multiplies the two integers. Test the function with different inputs.

In [128…

```python
def outer_function(x, y):
    def inner_function(a, b):
        return a * b
    return inner_function(x, y)

print(outer_function(2, 3))  # 6
print(outer_function(4, 5))  # 20
```

```
6
20
```

Question 102: Returning Multiple Values

Define a function that takes a single integer as input and returns the integer squared, cubed, and raised to the power of four. Test the function with different inputs.

In [129…

```
def powers(x):
    return x ** 2, x ** 3, x ** 4

print(powers(2))  # (4, 8, 16)
print(powers(3))  # (9, 27, 81)
```

```
(4, 8, 16)
(9, 27, 81)
```

Question 103: Recursive Function

Define a recursive function that calculates the factorial of a given number. Test the function with different inputs.

In [130…

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)

print(factorial(5))  # 120
print(factorial(6))  # 720
```

```
120
720
```

Question 104: Lambda Function

Define a lambda function that takes two integers as input and returns their sum. Test the lambda function with different inputs.

In [131…

```
add = lambda x, y: x + y
```

```
print(add(2, 3))  # 5
print(add(10, 20))  # 30
```

```
5
30
```

Question 105: Map Function

Use the map function to apply a lambda function that squares each number in a list of integers. Test with different lists.

In [132...

```
numbers = [1, 2, 3, 4, 5]
squared_numbers = list(map(lambda x: x ** 2, numbers))
print(squared_numbers)  # [1, 4, 9, 16, 25]
```

```
[1, 4, 9, 16, 25]
```

Question 106: Filter Function

Use the filter function to filter out all odd numbers from a list of integers. Test with different lists.

In [133...

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
print(even_numbers)  # [2, 4, 6, 8, 10]
```

```
[2, 4, 6, 8, 10]
```

Question 107: Function Decorator

Define a decorator function that prints 'Executing function...' before executing a function and 'Function executed.' after executing it. Apply this decorator to a function that takes a list of integers and returns their sum. Test the decorated function with different lists.

In [134...

```
def my_decorator(func):
    def wrapper(*args, **kwargs):
        print('Executing function...')
        result = func(*args, **kwargs)
        print('Function executed.')
        return result
    return wrapper

@my_decorator
```

```
def sum_list(lst):
    return sum(lst)
print(sum_list([1, 2, 3, 4, 5]))  # 15
```

```
Executing function...
Function executed.
15
```

Question 108: Function with (* args and **kwargs)

Define a function that takes variable-length arguments and keyword arguments and prints them. Test the function with different inputs.

In [135…

```
def print_args_kwargs(*args, **kwargs):
    print('args:', args)
    print('kwargs:', kwargs)

# Test
print_args_kwargs(1, 2, 3, a='apple', b='banana')
print_args_kwargs('hello', 'world', x=10, y=20)
```

```
args: (1, 2, 3)
kwargs: {'a': 'apple', 'b': 'banana'}
args: ('hello', 'world')
kwargs: {'x': 10, 'y': 20}
```

Question 109: Higher-Order Function

Define a higher-order function that takes a function and a list of integers as arguments, and applies the function to each integer in the list. Test with different functions and lists.

In [136…

```
def apply_function(func, lst):
    return [func(x) for x in lst]

# Test
print(apply_function(lambda x: x ** 2, [1, 2, 3, 4, 5]))  # [1, 4, 9, 16, 25]
print(apply_function(lambda x: x + 1, [1, 2, 3, 4, 5]))  # [2, 3, 4, 5, 6]
```

```
[1, 4, 9, 16, 25]
[2, 3, 4, 5, 6]
```

Question 110: Function Documentation

Define a function with a docstring that explains what the function does, its parameters, and its return value. Print the function's docstring.

In [137...
```
def example_function(x, y):
    """
    This function takes two integers and returns their sum.

    Parameters:
    x: The first integer.
    y: The second integer.

    Returns:
    int: The sum of the two integers.
    """
    return x + y

# Print docstring
print(example_function.__doc__)
```

```
    This function takes two integers and returns their sum.

    Parameters:
    x (int): The first integer.
    y (int): The second integer.

    Returns:
    int: The sum of the two integers.
```

Functions Examples

Example 1: Temperature Conversion

In [138...
```
def convert_temperature(temp,unit):
    """This function converts temperature between Celsius and Fahrenheit"""
    if unit=='C':
        return temp * 9/5 + 32  ## Celsius To Fahrenheit
    elif unit=="F":
        return (temp-32)*5/9 ## Fahrenheit to celsius
    else:
        return None
```

```
print(convert_temperature(25,'C'))
print(convert_temperature(77,'F'))
```

```
77.0
25.0
```

Example 2: Password Strength Checker

In [139…

```
def is_strong_password(password):
    """This function checks if the password is strong or not"""
    if len(password)<8:
        return False
    if not any(char.isdigit() for char in password):
        return False
    if not any(char.islower() for char in password):
        return False
    if not any(char.isupper() for char in password):
        return False
    if not any(char in '!@#$%^&*()_+' for char in password):
        return False
    return True

## calling the function
print(is_strong_password("WeakPwd"))
print(is_strong_password("Str0ngPwd!"))
```

```
False
True
```

Example 3: Calculate the Total Cost Of Items In a Shopping Cart

In [140…

```
def calculate_total_cost(cart):
    total_cost=0
    for item in cart:
        total_cost+=item['price']* item['quantity']

    return total_cost

cart=[
    {'name':'Apple','price':0.5,'quantity':4},
    {'name':'Banana','price':0.3,'quantity':6},
    {'name':'Orange','price':0.7,'quantity':3}

]
```

```
total_cost=calculate_total_cost(cart)
print(total_cost)
```

```
5.8999999999999995
```

Example 4: Check IF a String Is Palindrome

In [141…

```
def is_palindrome(s):
    s=s.lower().replace(" ","")
    return s==s[::-1]

print(is_palindrome("A man a plan a canal Panama"))
print(is_palindrome("Hello"))
```

```
True
False
```

Example 5: Calculate the factorials of a number using recursion

In [142…

```
def factorial(n):
    if n==0:
        return 1
    else:
        return n * factorial(n-1)

print(factorial(6))
```

```
720
```

Example 6: Validate Email Address

In [144…

```
import re

# Email validation function
def is_valid_email(email):
    """This function checks if the email is valid."""
    pattern = r'^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$'
    return re.match(pattern, email) is not None

# Calling the function
print(is_valid_email("test@example.com"))  # Output: True
print(is_valid_email("invalid-email"))  # Output: False
```

```
True
False
```

In [ ]:

```

```