For Taking input in any code

In [ ]:
```
input_string = input("Enter a list of numbers: ")
numbers = list(map(int, input_string.split()))
print(numbers)
```

```
Enter a list of numbers: 1 23 45 6 77 8
[1, 23, 45, 6, 77, 8]
```

Question 1

You are given a list of integers. Write a Python program that rearranges the list based on a specific pattern. The pattern is as follows: All even elements should come before all odd elements, and within even and odd elements, they should be sorted in ascending order.

Test case: Suppose you have the following list of integers:

Input:3 8 2 5 10 6 4 9

Output: 2 4 6 8 10 3 5 9

In [ ]:
```
def rearrange_list(lst):
    evens = sorted([x for x in lst if x % 2 == 0])
    odds = sorted([x for x in lst if x % 2 != 0])
    return evens + odds

# Test case
input_list = [3, 8, 2, 5, 10, 6, 4, 9]
print(rearrange_list(input_list))
```

```
[2, 4, 6, 8, 10, 3, 5, 9]
```

Question 2

Write a Python program to calculate the grade of a student based on the total marks scored in 5 subjects and determine if the student is eligible for a scholarship. The program should take the student's marks in 5 subjects as input.
Scholarship eligibility criteria:
The average score should be 85 or above for eligibility.
If the score is between 75 and 85, they should be on a waiting list.
If the score is below 75, the student is not eligible.

The program should include a function to calculate the average of the marks.
Students should also make use of a *pass* statement in one of the conditions.

Test case1: Input: 92, 88, 79, 85, 90 Output: Average: 86.8 Eligible for Scholarship	Test case2: Input: 80, 75, 78, 82, 70 Output: Average: 77.0 Waiting List	Test case3: Input: 60, 68, 72, 65, 70 Output: Average: 67.0 Not Eligible

In [ ]:
```python
def calculate_average(marks):
    return sum(marks) / len(marks)

def determine_scholarship(marks):
    average = calculate_average(marks)
    if average >= 85:
        return f"Average: {average}\nEligible for Scholarship"
    elif 75 <= average < 85:
        return f"Average: {average}\nWaiting List"
    else:
        return f"Average: {average}\nNot Eligible"

# Test cases
marks1 = [92, 88, 79, 85, 90]
marks2 = [80, 75, 78, 82, 70]
marks3 = [60, 68, 72, 65, 70]
print(determine_scholarship(marks1))
print(determine_scholarship(marks2))
print(determine_scholarship(marks3))
```

```
Average: 86.8
Eligible for Scholarship
Average: 77.0
Waiting List
Average: 67.0
Not Eligible
```

Question 3

Write a function login(username, password) that checks if both the username and password meet certain conditions (e.g., username must be alphanumeric, password must be at least 8 characters). Display appropriate error messages if the inputs don't meet the

criteria.
set inputs as, Username="user123" and password="password456"

Test case1: Enter Username: user123 Enter Password: password456 Output: Login Successful	Test case1: Enter Username: user123 Enter Password: pass456 Error: Password must be at least 8 characters long	Test case1: Enter Username: user_123 Enter Password: password456 Error: Username must be alphanumeric

In [ ]:
```python
def login(username, password):
    if not username.isalnum():
        return "Error: Username must be alphanumeric"
    if len(password) < 8:
        return "Error: Password must be at least 8 characters long"
```

```
    return "Login Successful"

# Test cases
print(login("user123", "password456"))
print(login("user123", "pass456"))
print(login("user_123", "password456"))
```

```
Login Successful
Error: Password must be at least 8 characters long
Error: Username must be alphanumeric
```

Question 4

Write a program that asks the user for a range of numbers and then counts how many of those numbers are even and how many are odd.

In [ ]:
```
def count_even_odd(start, end):
    evens = sum(1 for i in range(start, end + 1) if i % 2 == 0)
    odds = sum(1 for i in range(start, end + 1) if i % 2 != 0)
    return evens, odds

# Example usage
start, end = 1, 10
evens, odds = count_even_odd(start, end)
print(f"Evens: {evens}, Odds: {odds}")
```

```
Evens: 5, Odds: 5
```

Question 5

Refer Assignment 1 Question 31

Write a Python program that takes a list of dictionaries, where each dictionary represents a student with their name and scores in various subjects. The program should return a new dictionary where each key is a student's name, and the value is their average score across all subjects. (Using Function)

Sample List:

students = [{'name': 'Arjun', 'math': 85, 'science': 90, 'english': 78}, {'name': 'Balram', 'math': 92, 'science': 88, 'english': 84}, {'name': 'Damodar', 'math': 72, 'science': 75, 'english': 80}]

Expected Result:

{'Alice': 84.33, 'Bob': 88.0, 'Charlie': 75.67}

In [ ]:
```
def average_scores(students):
    averages = {}
    for student in students:
        name = student['name']
```

```
        scores = [score for subject, score in student.items() if subject != 'r
        averages[name] = round(sum(scores) / len(scores), 2)
    return averages

# Sample list
students = [
    {'name': 'Arjun', 'math': 85, 'science': 90, 'english': 78},
    {'name': 'Balram', 'math': 92, 'science': 88, 'english': 84},
    {'name': 'Damodar', 'math': 72, 'science': 75, 'english': 80}
]
print(average_scores(students))
```

```
{'Arjun': 84.33, 'Balram': 88.0, 'Damodar': 75.67}
```

Question 6

Write a program to generate calender of a month given the start_day and the number of days in that month.

In [ ]:
```
def generate_calendar(start_day, num_days):
    days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]
    calendar = ["   "] * start_day + [f"{i:2}" for i in range(1, num_days + 1)
    for i in range(0, len(calendar), 7):
        print(" ".join(calendar[i:i+7]))

# Example usage
generate_calendar(2, 30)  # Assuming the month starts on Wednesday and has 30
```

```
        1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30
```

Question 7

Write a recursive function to calculate the factorial of a number.
Sample Input: 5
Expected Output: 120

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

# Example usage
print(factorial(5))  # Output: 120
```

```
120
```

Question 8

Write a program to print the following pattern.

```
*
* *
* * *

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

In [ ]:
```
def print_pattern(n):
    for i in range(1, n + 1):
        print("* " * i)
    for i in range(n - 1, 0, -1):
        print("* " * i)

# Example usage
print_pattern(5)
```

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

Example

In [6]:
```
name = "Exam"

n = len(name)
for i in range(1, n+1):
  print(name[:i])
for i in range(n-1, 0, -1):
  print(name[:i])
```

```
E
Ex
Exa
Exam
Exa
Ex
E
```

Question 9

Write a python program to accept a coordinate point in a XY coordinate system and determine in which quadrant/axis the coordinate point lies.

Test Case 1:	**Test Case 2:**	**Test Case 3:**	**Test Case 4:**
Input: 7,8	Input: 0,0	Input: -7, 0	Input: 7,-8
Output:	Output:	Output:	Output:
First quadrant	Origin	X-axis	Fourth quadrant

In [ ]:
```python
def determine_quadrant(x, y):
    if x > 0 and y > 0:
        return "First quadrant"
    elif x < 0 and y > 0:
        return "Second quadrant"
    elif x < 0 and y < 0:
        return "Third quadrant"
    elif x > 0 and y < 0:
        return "Fourth quadrant"
    elif x == 0 and y != 0:
        return "Y-axis"
    elif y == 0 and x != 0:
        return "X-axis"
    else:
        return "Origin"

# Example usage
print(determine_quadrant(7, 8))  # Output: First quadrant
print(determine_quadrant(0, 0))  # Output: Origin
```

```
First quadrant
Origin
```

Question 10

Write a program to establish if a given integer num is a Curzon number. If (1+2^num) is exactly divisible by (1+2*num), then num is a Curzon number.

Example: Input: 5 Output: True

1 + 2 ** 5= 33

1 + 2 * 5 = 11

33 is a multiple of 11

Test Case 1:	**Test Case 2:**	**Test Case 3:**	**Test Case 4:**
Input: 5	Input: 10	Input: 6	Input: 4
Output:	Output:	Output:	Output:
True	False	True	False

In [ ]:
```python
def is_curzon(num):
    return (1 + 2 ** num) % (1 + 2 * num) == 0

# Example usage
print(is_curzon(5))  # Output: True
print(is_curzon(10))  # Output: False
```

Question 11

Write a program that accepts a sentence and calculates the number of uppercase letters and lowercase letters.

In [ ]:
```python
def count_letters(sentence):
    upper = sum(1 for c in sentence if c.isupper())
    lower = sum(1 for c in sentence if c.islower())
    return upper, lower

# Example usage
sentence = input("Sentence: ")
upper, lower = count_letters(sentence)
print(f"Uppercase letters: {upper}, Lowercase letters: {lower}")
```

```
Sentence: Through goes Hamilton
Uppercase letters: 2, Lowercase letters: 17
```

Question 12

Write a Python program to check whether an element exists within a tuple.

In [ ]:
```python
def element_exists(tup, element):
    return element in tup

# Example usage
tup = (1, 2, 3, 4, 5)
print(element_exists(tup, 3))  # Output: True
print(element_exists(tup, 6))  # Output: False
```

Question 13

Write a Python program to generate and print a list of the first and last 5 elements where the values are square numbers between 1 and 30 (both included).

In [ ]:
```python
def generate_square_numbers():
    squares = [i ** 2 for i in range(1, 31)]
    return squares[:5] + squares[-5:]

# Example usage
print(generate_square_numbers())
```

```
[1, 4, 9, 16, 25, 676, 729, 784, 841, 900]
```

Question 14

A triangle can be classified based on the lengths of its sides as equilateral, isosceles or scalene.
All 3 sides of an equilateral triangle have the same length.

An isosceles triangle has two sides that are the same length, and a third side that is a different length.
If all of the sides have different lengths then the triangle is scalene.

Write a program that reads the lengths of 3 sides of a triangle from the user.
Display a message indicating the type of the triangle and its area. (use Heron's formula)

In [ ]:
```
def classify_triangle(a, b, c):
    if a == b == c:
        triangle_type = "Equilateral"
    elif a == b or b == c or a == c:
        triangle_type = "Isosceles"
    else:
        triangle_type = "Scalene"

    s = (a + b + c) / 2
    area = (s * (s - a) * (s - b) * (s - c))**0.5
    return triangle_type, area

# Example usage
a, b, c = 3, 4, 5
triangle_type, area = classify_triangle(a, b, c)
print(f"Type: {triangle_type}, Area: {area}")
```

```
Type: Scalene, Area: 6.0
```

Question 15

Write a program to input a string and parse it. For every character in the string it should print one of the following:
Is a digit
Is a lowercase
Is an uppercase
Is something else
(a) Do the above program using if-elif-else
(b) Use match case statements

In [ ]:
```
def parse_string(s):
    for char in s:
        if char.isdigit():
            print(f"{char} is a digit")
        elif char.islower():
            print(f"{char} is a lowercase letter")
```

```
        elif char.isupper():
            print(f"{char} is an uppercase letter")
        else:
            print(f"{char} is something else")

# Example usage
parse_string("Hello123!")
```

```
H is an uppercase letter
e is a lowercase letter
l is a lowercase letter
l is a lowercase letter
o is a lowercase letter
@ is something else
1 is a digit
2 is a digit
3 is a digit
! is something else
```

In [ ]:
```
def parse_string(s):
    for char in s:
        match char:
            case _ if char.isdigit():
                print(f"{char} is a digit")
            case _ if char.islower():
                print(f"{char} is a lowercase letter")
            case _ if char.isupper():
                print(f"{char} is an uppercase letter")
            case _:
                print(f"{char} is something else")

# Example usage
parse_string("Hello123!")
```

Question 16

Write a Python program that takes a user's age as input and categorizes them into one of the following age groups:

- "Child" (age 0-12)
- "Teenager" (age 13-19)
- "Adult" (age 20-59)
- "Senior" (age 60 and above)

If the age is less than 0 or not a number, display an error message. Implement this using `if`, `elif`, and `else` statements. Also, include a `pass` statement in your code for a category where you don't want to take any action.

Expected Output:
If the user inputs `8`, the output will be `"Child"`.
If the user inputs `25`, the output will be `"Adult"`.
If the user inputs `-5`, the output will be `"Error: Age cannot be negative."`.
If the user inputs an invalid number like `abc`, the output will be `"Error: Please enter a valid number for age.`

In [ ]:
```python
def categorize_age(age):
    if age < 0:
        return "Error: Age cannot be negative."
    elif age <= 12:
        return "Child"
    elif age <= 19:
        return "Teenager"
    elif age <= 59:
        return "Adult"
    elif age >= 60:
        return "Senior"
    else:
        return "Error: Please enter a valid number for age."

# Example usage
print(categorize_age(8))  # Output: Child
print(categorize_age(25))  # Output: Adult
print(categorize_age(-5))  # Output: Error: Age cannot be negative.
```

Question 17

Write a program to read a 5 digit number and then display the number in the following formats. Eg., if the user entered 12345, the result should be

12345	1
2345	12

345 123

45 1234

5 12345

In [ ]:
```python
# Function to display the required format
def display_number_format(number):
    number_str = str(number)  # Convert the number to a string for easy slicin
    n = len(number_str)  # Get the length of the number (should be 5)

    # Generate the output format
    for i in range(n):
        left_part = number_str[i:]       # Left column: slice the number from
        right_part = number_str[:i + 1] # Right column: slice the number from
        print(f"{left_part:<5} {right_part}")  # Print in formatted columns

# Input: 5-digit number
number = int(input("Enter a 5-digit number: "))
display_number_format(number)
```

```
Enter a 5-digit number: 12345
12345 1
2345  12
345   123
45    1234
5     12345
```

Question 18

Write a Python program that prints all numbers from 1 to 20, except the multiples of 5. Use the continue statement to skip printing multiples of 5.

In [ ]:
```python
def print_numbers_except_multiples_of_5():
    for i in range(1, 21):
        if i % 5 == 0:
            continue
        print(i, end=" ")

# Example usage
print_numbers_except_multiples_of_5()
```

```
1 2 3 4 6 7 8 9 11 12 13 14 16 17 18 19
```

Question 19

Write a Python program to check if a student has passed or failed the exams with 5 subjects. A student passes if they score at least 40 in all subjects. Additionally, if the average score is 75 or more, print "Distinction."

In [ ]:
```python
def check_pass_fail(marks):
    if all(mark >= 40 for mark in marks):
```

```
        if sum(marks) / len(marks) >= 75:
            return "Distinction"
        return "Pass"
    return "Fail"

# Example usage
marks = [5, 55, 65, 75, 85]
print(check_pass_fail(marks))
```

```
Fail
```

Question 20

> The town has 500,000 residents now. For the past ten years, the population has been growing at a steady pace of 12% annually. write a program to find out the population at the end of each year over the past ten years.

In [ ]:
```
def population_growth(initial_population, growth_rate, years):
    population = initial_population
    for year in range(1, years + 1):
        population += population * growth_rate / 100
        print(f"Year {year}: {int(population)}")

# Example usage
population_growth(500000, 12, 10)
```

```
Year 1: 560000
Year 2: 627200
Year 3: 702464
Year 4: 786759
Year 5: 881170
Year 6: 986911
Year 7: 1105340
Year 8: 1237981
Year 9: 1386539
Year 10: 1552924
```

Question 21

Write a Python program that utilizes lambda functions and the map function to manipulate a list of numbers provided by the user. Additionally, use the filter function to identify certain numbers from the list.

In [ ]:
```
def manipulate_list(numbers):
    squared_numbers = list(map(lambda x: x ** 2, numbers))
    even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
    return squared_numbers, even_numbers

# Example usage
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
squared, evens = manipulate_list(numbers)
print(f"Squared Numbers: {squared}")
```

```
print(f"Even Numbers: {evens}")
```

```
Squared Numbers: [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
Even Numbers: [2, 4, 6, 8, 10]
```

Question 22

WAP to generate 10 random Unique number from 0-15

In [ ]:
```
import random

def generate_unique_random_numbers():
    return random.sample(range(16), 10)

# Example usage
print(generate_unique_random_numbers())
```

```
[5, 9, 7, 6, 4, 13, 15, 12, 8, 3]
```

Question 23

Write a program that prompts users to enter the numbers. The process will continue till the user enters -1. Finally, the program prints the count of prime and composite numbers entered.

In [4]:
```
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            return False
    return True

def count_prime_composite():
    primes = composites = 0
    while True:
        num = int(input("Enter a number (-1 to stop): "))
        if num == -1:
            break
        if is_prime(num):
            primes += 1
        else:
            composites += 1
    return primes, composites

# Example usage
primes, composites = count_prime_composite()
print(f"Primes: {primes}, Composites: {composites}")
```

```
Enter a number (-1 to stop): r
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-4-a7b308b4c3ca> in <cell line: 22>()
     20
     21 # Example usage
---> 22 primes, composites = count_prime_composite()
     23 print(f"Primes: {primes}, Composites: {composites}")

<ipython-input-4-a7b308b4c3ca> in count_prime_composite()
     10     primes = composites = 0
     11     while True:
---> 12         num = int(input("Enter a number (-1 to stop): "))
     13         if num == -1:
     14             break

ValueError: invalid literal for int() with base 10: 'r'
```

Question 24

Write a Python program to remove all non-alphanumeric characters (i.e., characters that are neither letters nor digits) from a given string using regular expressions.
Input: " Hello ! This is a test string. "
Output: "HelloThisisateststring"
Note: All spaces are also considered as Non-alphanumeric character.

In [ ]:
```python
def remove_non_alphanumeric(s):
    result = ""
    for char in s:
        if char.isalnum():
            result += char
    return result

# Example usage
input_string = "Hello! This is a test string."
output_string = remove_non_alphanumeric(input_string)
print(output_string)  # Output: HelloThisisateststring
```

```
HelloThisisateststring
```

Question 25

> Write a Python program to filter a list of integers using Lambda.
> Input: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
> Output:
> 1. Even numbers: [2, 4, 6, 8, 10]
> 2. Odd numbers from the said list: [1, 3, 5, 7, 9]

In [ ]:
```python
def filter_list(numbers):
    evens = list(filter(lambda x: x % 2 == 0, numbers))
    odds = list(filter(lambda x: x % 2 != 0, numbers))
```

```
    return evens, odds

# Example usage
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
evens, odds = filter_list(numbers)
print(f"Even Numbers: {evens}")
print(f"Odd Numbers: {odds}")
```

```
Even Numbers: [2, 4, 6, 8, 10]
Odd Numbers: [1, 3, 5, 7, 9]
```

Question 26

Write a function that converts a decimal number to binary number

In [ ]:
```
def decimal_to_binary(n):
    if n == 0:
        return "0"
    binary = ""
    while n > 0:
        binary = str(n % 2) + binary
        n = n // 2
    return binary

# Example usage
print(decimal_to_binary(10))  # Output: 1010
print(decimal_to_binary(0))   # Output: 0
print(decimal_to_binary(255)) # Output: 11111111
```

```
1010
0
11111111
```

Question 27

Write a Python program to find palindromes in a given list of strings using Lambda.
Orginal list of strings:

['php', 'w3r', 'Python', 'abcd', 'Java', 'aaa']

In [ ]:
```
def find_palindromes(strings):
    return list(filter(lambda x: x == x[::-1], strings))

# Example usage
strings = ['php', 'w3r', 'Python', 'abcd', 'Java', 'aaa']
print(find_palindromes(strings))
```

```
['php', 'aaa']
```

Question 28

Find all prime numbers between 1 and 50 .

In [ ]:
```python
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            return False
    return True

def find_primes():
    primes = []
    for num in range(2, 51):
        if is_prime(num):
            primes.append(num)
    return primes

# Example usage
print(find_primes())
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```

Question 29

Write a Python function student_grades that takes a dictionary where the keys are student names and the values are lists of their scores in different subjects. The function should return a dictionary where each student's name is a key, and the value is their average grade, rounded to two decimal places.

```
grades = {
   'Sat': [85, 90, 78],
   'Chid': [92, 88, 84],
   'Anand': [72, 75, 80]
}
print(student_grades(grades))
# Output: {'Sad': 84.33, 'Chid': 88.0, 'Anand': 75.67}
```

In [ ]:
```python
def student_grades(grades):
    return {student: round(sum(scores) / len(scores), 2) for student, scores i

# Example usage
grades = {
    'Sat': [85, 90, 78],
    'Chid': [92, 88, 84],
    'Anand': [72, 75, 80]
}
print(student_grades(grades))
```

Question 30

Write a Python function square_numbers that takes a list of numbers as input and returns a list of their squares using a lambda function and the map() function.
print(square_numbers([1, 2, 3, 4]))
# Output: [1, 4, 9, 16]

In [ ]:
```
def square_numbers(numbers):
    return list(map(lambda x: x ** 2, numbers))

# Example usage
print(square_numbers([1, 2, 3, 4]))
```

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

Question 31

Write a program to sum the series 1/2 + 2/3 + ……n/(n+1) .

In [ ]:
```
def sum_series(n):
    return sum(i / (i + 1) for i in range(1, n + 1))

# Example usage
print(sum_series(5))  # Output: 3.5500000000000003
```

```
3.5500000000000003
```

Question 32

Create a Python program that iterates over a list of numbers and prints only the even numbers, skipping the odd numbers using the continue statement.
Sample List: [1, 2, 3, 4, 5, 6]
Expected Output: 2, 4, 6

In [ ]:
```
def print_even_numbers(numbers):
    for num in numbers:
        if num % 2 != 0:
            continue
        print(num, end=" ")

# Example usage
numbers = [1, 2, 3, 4, 5, 6]
print_even_numbers(numbers)
```

```
2 4 6
```

Question 33

Write a Python program that takes an integer n as input and prints a hollow square pattern of size n x n using asterisks (*) and spaces. The pattern should have asterisks on the outer boundary, forming a square, while the inner cells should contain spaces.

Test case:

Enter the size of the hollow square: 8

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

In [3]:
```python
def hollow_square(n):
    for i in range(n):
        for j in range(n):
            if i == 0 or i == n -1  or j == 0 or j == n-1:
                print("*", end=" ")
            else:
                print(" ", end=" ")
        print()

# Example usage
hollow_square(8)
```

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

Question 34

You are given a list of dictionaries representing employee data. Each dictionary contains the following keys:

"name": a string representing the employee's name.

"age": an integer representing the employee's age.

"salary": a float representing the employee's monthly salary.

"department": a string representing the employee's department.

Write a Python program that performs the following tasks:

i) Create a function **average_salary_by_department** that takes the list of employee records and returns a dictionary where the keys are department names. The values are the average salaries of employees in each department. Calculate the average salary rounded to two decimal places.

ii) Create a function **department_with_highest_salary** that takes the list of employee records and returns the department name with the highest average salary.

iii) Create a function **youngest_employee_in_department** that takes the list of employee records and a department name as input. The function should return the name of the youngest employee in the specified department.

In [ ]:
```python
def average_salary_by_department(employees):
    department_salaries = {}
    for employee in employees:
        department = employee['department']
        salary = employee['salary']
        if department not in department_salaries:
            department_salaries[department] = []
        department_salaries[department].append(salary)
    return {dept: round(sum(salaries) / len(salaries), 2) for dept, salaries i

def department_with_highest_salary(employees):
    avg_salaries = average_salary_by_department(employees)
    return max(avg_salaries, key=avg_salaries.get)

def youngest_employee_in_department(employees, department):
    dept_employees = [emp for emp in employees if emp['department'] == departm
    youngest = min(dept_employees, key=lambda x: x['age'])
    return youngest['name']

# Example usage
employees = [
    {'name': 'Alice', 'age': 30, 'salary': 5000, 'department': 'HR'},
    {'name': 'Bob', 'age': 25, 'salary': 6000, 'department': 'IT'},
    {'name': 'Charlie', 'age': 35, 'salary': 7000, 'department': 'IT'},
    {'name': 'David', 'age': 28, 'salary': 5500, 'department': 'HR'}
]
```

```
print(average_salary_by_department(employees))
print(department_with_highest_salary(employees))
print(youngest_employee_in_department(employees, 'IT'))
```

```
{'HR': 5250.0, 'IT': 6500.0}
IT
Bob
```

Question 35

Write a Python program that processes a list of student grades. The goal is to filter out students who have passed (grades above a certain threshold), apply a curve to the passing grades using a lambda function, and then calculate the average of the curved grades using recursion.

Example:

1. Grades= 55,78,90,45,67,82,88
2. Min Mark pass=60
3. Grades after filter pass= [78, 90, 67, 82, 88]
4. Using the map function with a lambda that adds 5 to each grade:

   [78 + 5 = 83, 90 + 5 = 95, 67 + 5 = 72, 82 + 5 = 87, 88 + 5 = 93]

   Resulting in: [83, 95, 72, 87, 93]

5. Calculating Average Using Recursion: The recursive function calculates the average of the curved grades. To calculate this recursively:
   First call with grades as [83, 95, 72, 87, 93]:
   Sum is (83 + (4 * recursive_average([95,72,87,93])))/5
   Second call with grades as [95,72,87,93]:
   Sum is (95 + (3 * recursive_average([72,87,93])))/4
   Third call with grades as [72,87,93]:
   Sum is (72 + (2 * recursive_average([87,93])))/3
   Fourth call with grades as [87,93]:
   Sum is (87 + (1 * recursive_average([93])))/2
   Fifth call with grades as [93]
   Returns just 93.

Now we can backtrack to compute each level:
Fourth call returns (87 + (1 (93)))/2 = (87 + (1 (93)))/2 = (180)/2 = 90
Third call returns (72 + (2 (90)))/3 = (72 +180)/3 = *252/3=84
Second call returns (95 + (3 (84)))/4 = (95+252)/4=*347/4=86.75
First call returns: (83+(4(86.75)))/5=(83+347)/5=430/5=*86
Thus the final output when you run this code will be: 86

In [ ]:
```python
def filter_and_curve_grades(grades, min_pass):
    passed_grades = list(filter(lambda x: x >= min_pass, grades))
    curved_grades = list(map(lambda x: x + 5, passed_grades))
    return curved_grades

def recursive_average(grades):
    if len(grades) == 1:
        return grades[0]
    return (grades[0] + (len(grades) - 1) * recursive_average(grades[1:])) / l

# Example usage
grades = [55, 78, 90, 45, 67, 82, 88]
min_pass = 60
curved_grades = filter_and_curve_grades(grades, min_pass)
average = recursive_average(curved_grades)
print(f"Curved Grades: {curved_grades}")
print(f"Average: {average}")
```

```
Curved Grades: [83, 95, 72, 87, 93]
Average: 86.0
```

Question 36

Write a program to check whether two circles intersect each other:
Check for all 5 cases:

(a) Circles intersect each other
(b) Circles do not intersect

(c) Circles touch at a point from outside
(d) Circles touch at a point from inside
(e) One circle lies inside another

The user should give the values for r1, (x1,y1), r2, (x2, y2)

In [ ]:
```python
def check_circle_intersection(r1, x1, y1, r2, x2, y2):
    distance = ((x2 - x1) ** 2 + (y2 - y1) ** 2) ** 0.5

    if distance == r1 + r2:
        return "Circles touch at a point from outside"
    elif distance == abs(r1 - r2):
        return "Circles touch at a point from inside"
    elif distance < abs(r1 - r2):
        return "One circle lies inside another"
    elif distance < r1 + r2:
        return "Circles intersect each other"
    else:
        return "Circles do not intersect"

# Example usage
r1, x1, y1 = 5, 0, 0
r2, x2, y2 = 3, 4, 0
```

```
print(check_circle_intersection(r1, x1, y1, r2, x2, y2))
```

```
Circles intersect each other
```

Question 37

Write a function to find the mean, median and mode of a list. (Input will be entered by user)

Test Case 1:	**Test Case 2:**	**Test Case 3:**
Input: 4 5 3 2 1 2	Input: 4 5 3 2 1	Input: 4 5 3 2 1 4 5
Output:	Output:	Output:
Mean is 2.8333	Mean is 3	Mean is 3.42857
Median is 2.5	Median is 3	Median is 4
Mode is 2	Mode is <No Mode>	Mode is 4,5

In [ ]:
```
from statistics import mean, median, mode

def calculate_statistics(numbers):
    try:
        mean_value = mean(numbers)
        median_value = median(numbers)
        mode_value = mode(numbers)
    except StatisticsError:
        mode_value = "<No Mode>"

    return mean_value, median_value, mode_value

# Example usage
numbers = [4, 5, 3, 2, 1, 2]
mean_value, median_value, mode_value = calculate_statistics(numbers)
print(f"Mean is {mean_value}")
print(f"Median is {median_value}")
print(f"Mode is {mode_value}")
```

```
Mean is 2.8333333333333335
Median is 2.5
Mode is 2
```