Interactive Playground for Problem Solving and Python Programming

By MR. P Jason

Assistant Professor, Department of Information Technology

1. VOTER’S AGE VALIDATION

AIM:

The aim of this task is to create a Python program that validates whether a person is eligible to vote based on their age. The typical voting age is 18 years and older. The program will take the person's birth year as input, calculate their age, and determine if they are eligible to vote.

ALGORITHM:

- Step-1. Input the Birth Year: Prompt the user to enter their birth year.
- Step-2. Calculate the Current Year: Get the current year from the system.
- Step-3. Calculate the Age: Subtract the birth year from the current year to determine the person's age.
- Step-4. Check Eligibility:
- Step-5.If the age is 18 or older, the person is eligible to vote.
- Step-6.If the age is less than 18, the person is not eligible to vote.
- Step-7. Display the Result: Print a message indicating whether the person is eligible to vote or not.

PROGRAM:

In [31]:
```python
from datetime import datetime

def is_eligible_to_vote(birth_year):
    current_year = datetime.now().year
    age = current_year - birth_year

    if age >= 18:
        return True, age
    else:
        return False, age

def main():
    try:
        birth_year = int(input("Enter your birth year: "))
        eligible, age = is_eligible_to_vote(birth_year)

        if eligible:
            print(f"You are {age} years old. You are eligible to vote.")
        else:
            print(f"You are {age} years old. You are not eligible to vote.")
    except ValueError:
        print("Invalid input. Please enter a valid year.")

if __name__ == "__main__":
    main()
```

```
You are 18 years old. You are eligible to vote.
```

2. SIMPLE SORTING

AIM:

The aim of this program is to sort a list of numbers using the bubble sort algorithm.

ALGORITHM:

- Step-1. Given a list of numbers.
- Step-2. Repeat the following steps for each element in the list:
- Step-3. Iterate through the list from the beginning to the end.
- Step-4. Compare each pair of adjacent elements.
- Step-5. If the current element is greater than the next element, swap them.
- Step-6. Repeat step 2 until no swaps are needed (the list is sorted).

PROGRAM:

In [32]:
```python
def bubble_sort(my_list):
    n = len(my_list)
```

```
    for i in range(n):
        for j in range(0, n - i - 1):
            if my_list[j] > my_list[j + 1]:
                my_list[j], my_list[j + 1] = my_list[j + 1], my_list[j]

# Example usage:
# Prompt the user to enter elements separated by spaces
user_input = input("Enter the elements in the list, separated by spaces: ")

# Split the input string into a list of strings, then convert each to an int
my_list = list(map(int, user_input.split()))

# Sort the list using bubble sort
bubble_sort(my_list)

# Print the sorted list
print("Sorted array is:", my_list)
```

```
Sorted array is: [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

3. BIANRY SEARCH

AIM:

The aim of this program is to do binary search in a list of numbres.

ALGORITHM:

- Step-1. Given a sorted array and a target value to search.
- Step-2. Set two pointers, low and high, initially pointing to the first and last elements of the array, respectively.
- Step-3. While low is less than or equal to high, do:
- Step-4. Calculate the middle index as (low + high) // 2.
- Step-5. If the middle element is equal to the target, return its index.
- Step-6. If the middle element is greater than the target, update high to mid -1.
- Step-7. If the middle element is less than the target, update low to mid + 1.
- Step-8. If the target is not found in the array, return -1.

PROGRAM:

In [33]:
```
def binary_search(my_list, target):
    low = 0
    high = len(my_list) - 1

    while low <= high:
        mid = (low + high) // 2
        if my_list[mid] == target:
```

```
            return mid
        elif my_list[mid] < target:
            low = mid + 1
        else:
            high = mid - 1

    return -1

# Example usage:
# Prompt the user to enter elements separated by spaces
user_input = input("Enter the elements in the list, separated by spaces: ")

# Split the input string into a list of strings, then convert each to an int
my_list = list(map(int, user_input.split()))

# Sort the list before performing binary search
my_list.sort()

print("The Sorted List is: ", my_list)

# Prompt the user to enter the target element
target = int(input("Enter the target element: "))

# Perform binary search
result = binary_search(my_list, target)

if result != -1:
    print("Element", target, "is present at index", result)
else:
    print("Element", target, "is not present in the list.")
```

```
The Sorted List is:  [1, 2, 3, 4, 5, 6, 7, 8, 9]
Element 4 is present at index 3
```

4. LINEAR SEARCH

AIM:

The aim of this program is to search for a target value in a list using the linear search algorithm.

ALGORITHM:

- Step-1. Given a list and a target value to search.
- Step-2. Iterate through each element in the list:
- Step-3. If the current element is equal to the target, return its index.
- Step-4. If the target is not found in the list, return -1.

PROGRAM:

In [23]:
```
def linear_search(my_list, target):
    for index in range(len(my_list)):
        if my_list[index] == target:
            return index
    return -1

# Example usage:
# Prompt the user to enter elements separated by spaces
user_input = input("Enter the elements in the list, separated by spaces: ")

# Split the input string into a list of strings, then convert each to an int
my_list = list(map(int, user_input.split()))

# Sort the list before performing binary search
my_list.sort()

print("The Sorted List is: ", my_list)

# Prompt the user to enter the target element
target = int(input("Enter the target element: "))

result = linear_search(my_list, target)

if result != -1:
    print("Element", target, "is present at index", result)
else:
    print("Element", target, "is not present in the list.")
```

```
The Sorted List is [1, 7, 22, 22, 44, 300]
Element 22 is present at index 2
```

5. SWAPPING TWO NUMBERS

AIM:

The aim of this program is to exchange the values of two variables.

ALGORITHM:

- Step-1. Start
- Step-2. Initialize two variables, a and b, with some values.
- Step-3. Print the values of a and b before swapping.
- Step-4. Exchange the values of a and b using a temporary variable.
- Step-5. Print the values of a and b after swapping.
- Step-6. Stop

In [34]:
```
# Exchange the values of two variables
a = int(input("Enter the value of a"))
b = int(input("Enter the value of b"))
```

```
print("Before swapping:")
print("a =", a)
print("b =", b)

# Using a temporary variable
temp = a
a = b
b = temp

print("\nAfter swapping:")
print("a =", a)
print("b =", b)
```

```
Before swapping:
a = 12
b = 6

After swapping:
a = 6
b = 12
```

6. TOWER OF HANOI

AIM:

To solve the Towers of Hanoi problem for a given number of disks, moving them from one peg to another, obeying the rules of the game.

ALGORITHM:

- Step-1. Define three pegs: source (where the disks start), auxiliary (a spare peg), and destination (where the disks should end up).
- Step-2. Recursively move the top n-1 disks from the source peg to the auxiliary peg, using the destination peg as a temporary location.
- Step-3. Move the largest disk from the source peg to the destination peg.
- Step-4. Recursively move the n-1 disks from the auxiliary peg to the destination peg, using the source peg as a temporary location.

In [35]:
```
def towers_of_hanoi(n, source, auxiliary, destination):
    if n == 1:
        print("Move disk 1 from", source, "to", destination)
        return
    towers_of_hanoi(n-1, source, destination, auxiliary)
    print("Move disk", n, "from", source, "to", destination)
    towers_of_hanoi(n-1, auxiliary, source, destination)

# Example usage:
n = int(input("Enter Number of disks"))
```

```
source_peg = "A"
auxiliary_peg = "B"
destination_peg = "C"
print("Steps to solve the Towers of Hanoi with", n, "disks:")
towers_of_hanoi(n, source_peg, auxiliary_peg, destination_peg)
```

```
Steps to solve the Towers of Hanoi with 3 disks:
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 3 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C
```

7. SIMPLE SQUARE ROOT, EXPONENTIATION

AIM:

To calculate the square root and exponentiation of a number using Python.

ALGORITHM:

- Step-1. Square Root: Use the math.sqrt() function from the math module
- Step-2. to calculate the square root of a number.
- Step-3. Exponentiation: Use the ** operator to perform exponentiation in Python.
- Step-4. For example, a ** b calculates a^b.

In [36]:
```
import math

# Function to calculate square root
def calculate_square_root(number):
    square_root = math.sqrt(number)
    return square_root

# Function to perform exponentiation
def calculate_exponentiation(base, exponent):
    result = base ** exponent
    return result

# Example usage for square root
number = int(input("Enter Number: "))
square_root = calculate_square_root(number)
print("Square root of", number, ":", square_root)

# Example usage for exponentiation
base = int(input("Enter base number: "))
exponent = int(input("Enter exponent number: "))
```

```
result = calculate_exponentiation(base, exponent)
print(base, "raised to the power of", exponent, ":", result)
```

```
Square root of 2 : 1.4142135623730951
2 raised to the power of 2 : 4
```

8. PYRAMIND PATTERN

AIM:

The aim of this program is to print a pyramid pattern using iterative loops and conditionals.

ALGORITHM:

- Step-1. Start
- Step-2. Prompt the user to input the number of rows for the pyramid pattern.
- Step-3. Generate and print the pyramid pattern using iterative loops and conditionals.
- Step-4. Stop

In [37]:
```
# Print a pyramid pattern of asterisks
rows = int(input("Enter the number of rows for the pyramid pattern: "))

for i in range(1, rows + 1):
    print(" " * (rows - i) + "* " * i)
```

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

Conclusion

This interactive playground enhances logical problem-solving abilities with Python Programming.

By MR. P Jason

Assistant Professor, Department of Information Technology