Beginner's Guide to Python: Must-Do Programs

Get Started - Beginner-Friendly Programs

Think Like a Programmer - Output Prediction Challenges

Pattern Mastery - Fun with Series and Patterns

Advanced Logic - Combining Algorithms and Code

1: Get Started - Beginner-Friendly Programs

1.1 Calculate the Sum of Digits

Write a program to calculate the sum of digits of a given three-digit integer.

Example:

```
a = 426
# Output: 4 + 2 + 6 = 12
```

Note:

- Do not use `if-else`, `for`, or `while` statements.

Clue:

- `c = a % 10` results in `6`
- `a = a // 10` results in `42`

1.2 Reverse a Three-Digit Number

Write a program to reverse a given three-digit integer.

Example:

```
a = 426
# Output: 624
```

Note:

- Do not use `if-else`, `for`, or `while` statements.

Clue:

- `c = (c * 10) + (a % 10)` results in `6` (initial value of `c` is `0`)

1.3 Check if a Number is a Palindrome

Write a program to check whether a given three-digit integer is a palindrome.

Example:

```
a = 626
# Output: Yes, because reverse of 626 is also 626
```

Note:

- Do not use `for` or `while` loops.

1.4 Predict the Output of Mathematical Expressions

What is the output of the following expressions?

```
10 % 3  # Output: 1
3 % 10  # Output: 3
25 / 2  # Output: 12.5
```

1.5 Determine Promotion Status Based on Marks

Write a program to determine if a student is promoted based on marks in three subjects (S1, S2, S3).

Example 1:

```
S1 = 40, S2 = 50, S3 = 70
# Output: Promoted
```

Example 2:

```
S1 = 25, S2 = 34, S3 = 70
# Output: Not Promoted
```

Example 3:

```
S1 = 24, S2 = 34, S3 = 20
# Output: Not Promoted
```

Notes:

1. Print “Promoted” only if all three subjects have marks greater than or equal to 35.
2. Avoid using logical operators like `and` or `or`.
3. Avoid writing multiple print statements which prints "Not promoted"

2: Think Like a Programmer - Output Prediction Challenges

2.1 Predict the Output of a Loop

Given the following code, predict the output and determine how many iterations the loop will run:

```
i = 0
n = 7
for i in range(1, n):
    print(i)
print(i)
```

2.2 Loop Through a Range of Values

What will be the output of the code below, and how many iterations does the loop execute?

```
i = 0
n = 7
for i in range(0, n):
    print(i)
print(i)
```

2.3 While Loop with Condition

In the following while loop, predict the output and count how many times the loop will iterate:

```
i = 0
n = 7
while(i <= n):
    print(i)
    i += 1
print(i)
```

2.4 Iteration in a While Loop

Predict the final value of `i` after the loop ends:

```
i = 0
n = 7
while(i < n):
    print(i)
    i += 1
print(i)
```

2.5 For Loop with a Range

What will be printed by the following code? How many times does the loop run?

```
i = 0
n = 5
for i in range(5, n):
    print(i)
print(i)
```

2.6 Looping with Negative Start

How does starting the range from a negative number affect the loop's output?

```
i = 0
n = 5
for i in range(-3, n):
    print(i)
print(i)
```

2.7 While Loop with Increment

Predict the values printed during the loop execution:

```
i = 0
n = 5
while(i < n):
    print(i)
    i += 1
print(i)
```

2.8 Breaking the Loop

In this code, predict the output and understand how the `break` statement affects the loop:

```
i = 0
n = 10
for i in range(1, n):
    print(i)
    if (i == 5):
        break
print(i)
```

2.9 Loop with a Step Value

Given the following loop, predict the output and see how the step value affects the iterations:

```
i = 0
n = 10
for i in range(1, n+1, 2):
    print(i)
print(i)
```

2.10 While Loop with a Conditional Increment

Predict the output when specific conditions change the increment behavior inside the loop:

```
i = 1
n = 10
while(i <= n):
    print(i)
    if (i % 5 == 0):
        i += 4
    i += 1
print(i)
```

2.11 Nested Loops

In this code, predict the output for each iteration of the inner and outer loops:

```
i = 1
j = 1
while(i <= 5):
    print(i)
    j = 1
    while(j <= 5):
        print(j)
        j += 1
    print("")
    i += 1
print(i, j)
```

2.12 Nested Loops with Conditional Logic

In this problem, predict the output and see how conditional statements affect both loops:

```
i = 1
j = 1
while(i <= 5):
    print(i)
    j = 1
    while(j <= 5):
        print(j)
        if (j % 2 == 0):
            i += 2
        j += 1
    print("")
    i += 1
print(i, j)
```

3: Pattern Mastery - Fun with Series and Patterns

3.1 Factorial Calculation

Write a program to find the factorial value of a given number `n`.

3.2 Prime Number Check

Write a program to check whether a given number `n` is prime or not.

3.3 Print First `n` Prime Numbers

Write a program to print the first `n` prime numbers.

3.4 Sum of First `n` Prime Numbers

Write a program to calculate the sum of the first `n` prime numbers.

3.5 Factorial Series

Write a program to print the factorial values of numbers from `1` to `n`.

3.6 Sum of Factorial Series

Write a program to calculate the sum of the factorial values from `1!` to `n!`.

3.7 Special Series Calculation

Write a program to calculate the following series up to `n` terms:

```
2!/1! + 3!/2! + 4!/3! + ... + (n+1)!/n!
```

3.8 Pyramid Pattern

Write a program to print the following pyramid pattern:

```
1 2 3 4 5 4 3 2 1
```

3.9 Inverted Triangle Pattern

Write a program to print the following inverted triangle pattern:

```
12345
1234
123
12
1
```

3.10 Right-Aligned Triangle Pattern

Write a program to print the following right-aligned triangle pattern:

```
12345
 1234
  123
   12
    1
```

3.12 Left-Aligned Pyramid Pattern

Write a program to print the following left-aligned pyramid pattern:

```
    1
    12
    123
    1234
    12345
```

3.13 Center-Aligned Pyramid Pattern

Write a program to print the following center-aligned pyramid pattern:

```
    1
   12
  123
 1234
12345
```

3.14 Diamond Pattern

Write a program to print the following diamond pattern:

```
123454321
1234 4321
123   321
12     21
1       1
```

3.15 Symmetrical Diamond Pattern

Write a program to print the following symmetrical diamond pattern:

```
1       1
12     21
123   321
1234 4321
123454321
```

3.16 Alphabet Diamond Pattern

Write a program to print the following diamond pattern using alphabets:

```
    A
  A B C
A B C D E
  A B C
    A
```

3.17 Symmetrical Alphabet Diamond Pattern

Write a program to print the following symmetrical alphabet pattern:

```
ABCBA
AB BA
A   A
AB BA
ABCBA
```

4. Advanced Logic - Combining Algorithms and Code

This section presents a series of programming puzzles designed to test and enhance your understanding of arrays and algorithms in Python. Let's dive in!

4.1. Duplicate Annihilation

Write a Python program that replaces all duplicate elements within a single-dimensional integer array with the value '0'.

Example:

- **Input Array:** `[1, 2, 2, 3, 4, 4, 4, 5]`
- **Expected Output:** `[1, 2, 0, 3, 4, 0, 0, 5]`

4.2. Unique Element Squad

Create a Python program that eliminates all duplicate elements from a single-dimensional integer array, resulting in an array containing only unique values.

Example:

- **Input Array:** `[1, 2, 2, 3, 4, 4, 4, 5]`
- **Expected Output:** `[1, 2, 3, 4, 5]`

4.3. Ascending Order Achiever

Develop a Python program to sort 'n' integer values within a single-dimensional array into ascending order.

Example:

- **Input Array:** `[5, 2, 8, 1, 9]`
- **Expected Output:** `[1, 2, 5, 8, 9]`

4.4. Matrix Addition Master

Write a Python program to perform matrix addition on two matrices, 'a' and 'b', both of size m x n. The resulting sum should be stored in a matrix 'c'. Use two-dimensional arrays to represent the matrices.

Example:

- **Input Matrix 'a':**

```
[1, 2, 3]
[4, 5, 6]
```

- **Input Matrix 'b':**

```
[7, 8, 9]
[10, 11, 12]
```

- **Expected Output Matrix 'c':**

```
[8, 10, 12]
[14, 16, 18]
```

4.5. Matrix Multiplication Maestro

Construct a Python program to perform matrix multiplication on two matrices, 'a' and 'b', both of size m x n. Store the product in a matrix 'c'. Represent the matrices using two-dimensional arrays.

Example:

- **Input Matrix 'a':**

```
[1, 2]
[3, 4]
```

- **Input Matrix 'b':**

```
[5, 6]
[7, 8]
```

- **Expected Output Matrix 'c':**

```
[19, 22]
[43, 50]
```

4.6. Magic Square Conjurer

Craft a Python program that generates a magic square for any given odd-sized matrix. For insights and guidance, refer to this helpful video: https://www.youtube.com/watch?v=XEVNYisghbg

Example:

- **Input Matrix Size:** 3x3
- **Expected Output:**

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

(Each row, column, and diagonal sums to 15)