This notebook contains a collection of beginner-friendly Python programs (Program 91–100), useful for logic building and interview preparation.

this notebook can be accessed via (https://github.com/amitvsuryavanshi04/SIC_programming_and_coding)

Program 91 Write a function that stutters a word as if someone is struggling to read it. The first two letters are repeated twice with an ellipsis ... and space after each, and then the word is pronounced with a question mark ?.

```
def stutter(word):
  if len(word) < 2:
    return "Word must be at least two characters long."
  stuttered_word = f"{word[:2]}... {word[:2]}... {word}?"
  return stuttered_word
# Test cases
print(stutter("incredible"))
print(stutter("enthusiastic"))
print(stutter("outstanding"))
```

```
in... in... incredible?
en... en... enthusiastic?
ou... ou... outstanding?
```

Program 92 Create a function that takes an angle in radians and returns the corresponding angle in degrees rounded to one decimal place.

```
import math

def radians_to_degrees(radians):
  degrees = radians * (180 / math.pi)
  return round(degrees, 1)
#test cases
print(radians_to_degrees(1))
print(radians_to_degrees(100))
print(radians_to_degrees(264))
```

```
57.3
5729.6
15126.1
```

Program 93 In this challenge, establish if a given integer num is a Curzon number. If 1 plus 2 elevated to num is exactly divisible by 1 plus 2 multiplied by num, then num is a Curzon number. Given a non-negative integer num, implement a function that returns True if num is a Curzon number, or False otherwise.

```
def is_curzon(num):
  numerator = 2 ** num + 1
  denominator = 2 * num + 1
  return numerator % denominator == 0
# Test cases
print(is_curzon(5))
print(is_curzon(10))
print(is_curzon(14))
```

```
True
False
True
```

Program 94 Given the side length x find the area of a hexagon.

```
import math
def area_of_hexagon(x):
  area = (3 * math.sqrt(3) * x ** 2) / 2
  return round(area, 1)
# Test cases
print(area_of_hexagon(1))
print(area_of_hexagon(2))
print(area_of_hexagon(4))
```

```
2.6
10.4
41.6
```

Program 95 create a function that returns a base-2 (binary) representation a base-10 (decimal) string number. To convert is simple((2) means base-2 and (10) means base-10)

```
def binary(decimal):
  binary_str = ""
  while decimal > 0:
    remainder = decimal % 2
    binary_str = str(remainder) + binary_str
    decimal = decimal // 2
  return binary_str if binary_str else "0"
print(binary(1))
print(binary(45))
print(binary(264))
```

```
1
101101
100001000
```

Program 96 Create a function that takes three arguments a,b,c and returns the sum of the numbers that are evenly divided by c from the range a,b inclusive.

```
def evenly_divisible(a, b, c):
  total = 0
  for num in range(a, b + 1):
    if num % c == 0:
      total += num
  return total
print(evenly_divisible(1, 10, 20))
print(evenly_divisible(1, 10, 2))
print(evenly_divisible(1, 10, 3))
```

```
0
30
18
```

Program 97 Create a function that returns true if a given inequality expression is correct and false otherwise.

```
def correct_signs(expression):
  try:
    return eval(expression)
  except:
    return False
```

```
print(correct_signs("3 < 7 < 11"))
print(correct_signs("13 > 44 > 33 > 1"))
print(correct_signs("1 < 2 < 6 < 9 > 3"))
```

```
True
False
True
```

Program 98 Create a function that replaces all the vowels in a string with a specified character.

```
def replace_vowels(string, char):
  vowels = "AEIOUaeiou" # List of vowels to be replaced
  for vowel in vowels:
    string = string.replace(vowel, char) # Replace each vowel with
  return string
print(replace_vowels("the aardvark", "#"))
print(replace_vowels("minnie mouse", "?"))
print(replace_vowels("shakespeare", "*"))
```

```
th# ##rdv#rk
m?nn?? m??s?
sh*k*sp**r*
```

Program 99 write a function that calculates the factorial of a number recursively.

```
#code
def factorial(n):
  if n == 0:
    return 1 # Base case: factorial of 0 is 1
  else:
    return n * factorial(n - 1) # Recursive case: n! = n * (n-1)!
print(factorial(5))
print(factorial(45))
print(factorial(1))
print(factorial(0))
```

```
120
119622220865480194561963161495657715064383733760000000000
1
```

```
1
```

Program 100 Hamming Distance is the number of characters that differ between two strings.

```
def hamming_distance(str1, str2):
# Check if the strings have the same length
  if len(str1) != len(str2):
    raise ValueError("Input strings must have the same length")
# Initialize a counter to keep track of differences
  distance = 0
# Iterate through the characters of both strings
  for i in range(len(str1)):
    if str1[i] != str2[i]:
      distance += 1 # Increment the counter for differences
  return distance
print(hamming_distance("abcde", "bcdef"))
print(hamming_distance("abcde", "abcde"))
print(hamming_distance("strong", "strung"))
```

```
5
0
1
```