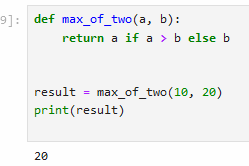
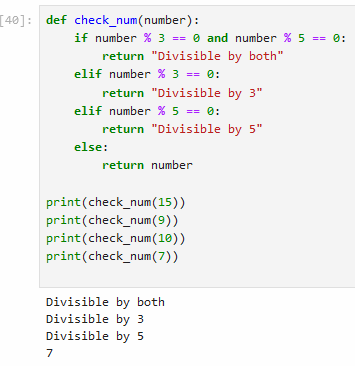
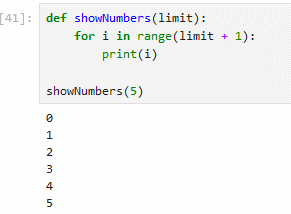
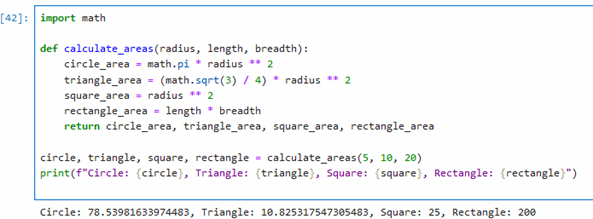
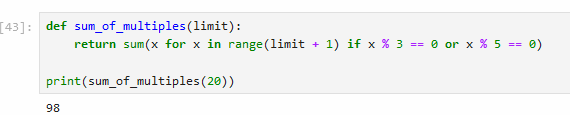
Name : Abdul Rehman Section :B

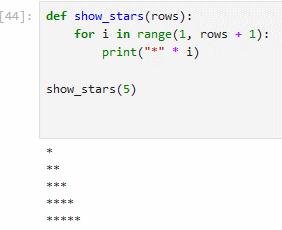
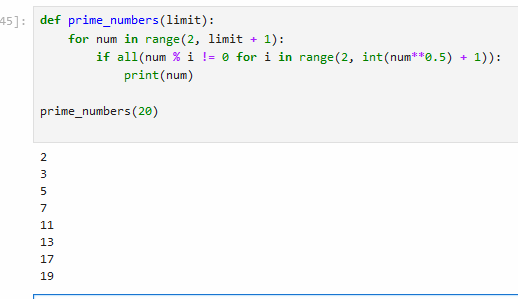
Course : AI&ES Roll No : CT-22052

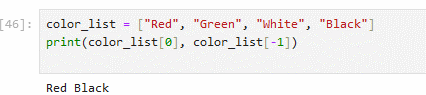
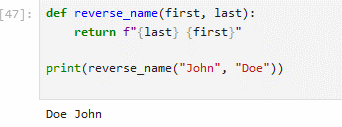
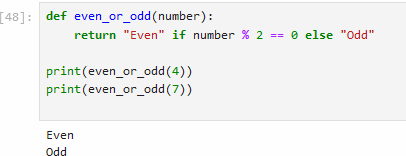
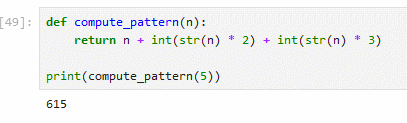
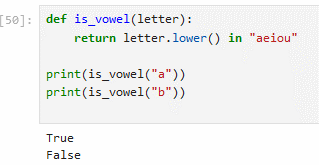
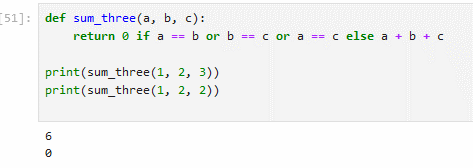
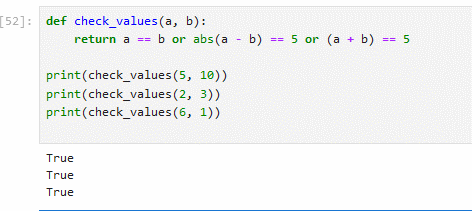
---------------------------------------------------------------------------------------------------------------------  
  
1: **Expression:** A combination of values, variables, and operators that results in a single value.  
  
2: **Syntax Error:** An error in the structure or grammar of the code. Python can't understand the code due to incorrect punctuation, missing keywords, etc.  
  
3: This expression results in the string "\*\*\*\*\*\*\*\*\*\*"  
  
4: **Variable:** A named container that stores a value. The value can be changed during program execution.  
 e.g : name=”Abdul Rehman”  
  
5:

* int (integers)
* float (floating-point numbers)
* str (strings)
* bool (Boolean values: True or False)
* None (represents the absence of a value)

6: To define multi-line strings and to include single or double quotes within the string.  
  
7:   
name[1] returns "o" (the second character in the string).  
name[-2] returns "t" (the second-to-last character).  
name[1:-1] returns "ohn Smit" (all characters except the first and last).  
len(name) returns 10 (the number of characters in the string).  
  
8: **name.title()**: Returns "John Smith" (capitalizes the first letter of each word).  
  
9: **name.strip()**: Removes any leading or trailing whitespace (spaces or tabs) from the string.  
  
10: **name.find("Smith")**: Returns 5 (the starting index of the word "Smith" in the string).  
  
11: **name.replace("j", "k")**: Changes the string to "kohn smith" (replaces all occurrences of "j" with "k").  
  
12: if "John" in name:  
  
13:   
**Integers (int)**: Whole numbers (e.g., 10, -5, 0).  
**Floating-point numbers (float)**: Numbers with decimals (e.g., 3.14, -2.5).  
**Complex numbers**: Numbers with an imaginary part (e.g., 2 + 3j).  
  
14: 10 / 3 = 3.3333333333333335 (regular division, returns a float) while 10 // 3 = 3 (floor division, returns the integer part of the result).  
  
15 : 1000 (10 power 3).  
  
16: After this, x will be 3 (equivalent to x = x + 2)  
  
17: s  
 - round(3.14159) returns 3.  
 - round(3.14159, 2) returns 3.14 (rounds to 2 decimal places).  
  
18: **float(1)** = 1.0 (converts the integer 1 to a floating-point number)  
  
19: **10 == "10"**: False (compares a number to a string, which are different data types)  
  
20: **"bag" > "apple"**: True (string comparison is based on alphabetical order)  
  
21: **not(True or False)**: False (evaluates to not(True), which is False)  
  
22: **range(1, 10, 2)**: Returns a sequence of numbers: 1, 3, 5, 7, 9 (starts at 1, increments by 2, stops before 10)  
  
23:   
  
24:  
  
25:  
  
26:  


27:  


28:  
  
  
29:  


30:  
  
  
  
31:  
  
  
32:  
  
33:  
  
34:  
  
35:  
  
36:  
  
37:  
