**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **25 may 2020** | **Name:** | **Shreya poojary** |
| **Course:** | **python** | **USN:** | **4al16ec074** |
| **Topic:** | **For loop** | **Semester & Section:** | **8-B** |
| **Github Repository:** | **Shreya-test** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session**  **C:\Users\Nelson\Desktop\p1.PNG**  **C:\Users\Nelson\Desktop\p2.PNG** |
| **For Loops**  Loop is iterations. Do things again. Traversing through, Going to each value.  They don’t come again as what you might think in English..  Loops take time to understand so it is a matter of time...  For loop   1. the\_count = [1, 2, 3, 4, 5] 2. fruits = ['apples', 'oranges', 'pears', 'apricots'] 3. change = [1, 'rupee', 2, 'paisa', 3, 'mudra'] 5. # traverse through the list we created 6. for xxx inthe\_count: 7. print(f"This is count {xxx}") 9. # traverse throught a string list we created 10. for fruit in fruits: 11. print(f"A fruit of type: {fruit}") 13. # also we can go through mixed lists too 14. for i in change: 15. print(f"I got {i}") 17. # we can also build lists, first start with an empty one 18. elements = [] 20. # then use the range function to do 0 to 5 counts 21. for i in range(0, 6): 22. print(f"Adding {i} to the list.") 23. # append is a function that lists understand 24. elements.append(i) 26. # now we can print them out too 27. for i in elements: 28. print(f"Element was: {i}")   This one is for\_each\_xxx   1. elements = [] 2. # then use the range function to do 0 to 5 counts 3. forxxx\_eachinthe\_count: 4. print(f"Adding {xxx\_each} to the list.") 5. # append is a function that lists understand 6. elements.append(xxx\_each)   Traverse and double the value   1. # this first kind of for-loop goes through a list 2. foreachvalueinthe\_count: 3. print(f"This is count {eachvalue\*2}")   Other languages dont have "in" the one in "for xxx in listofelements" hence they have to do the it the length way!   1. for i in range(0, len(the\_count)): 2. print (the\_count[i],i) 3. element=[] 4. for i in range(6, 0, -1): 5. print(f"Adding {i} to the list.") 6. # append is a function that lists understand 7. elements.append(i)   List and changing things for printing.   1. the\_count = [10, 20, 30, 40, 50] 3. for xxx inthe\_count: 4. print(xxx-1) 5. for i in range(0, 3): 6. print(i)   List with loop and append command.   1. # we can also build lists, first start with an empty one 2. elements = [] 4. # then use the range function to do 0 to 5 counts 5. for p in range(0, 6): 6. print(f"Adding {p} to the list.") 7. # append is a function that lists understand 8. elements.append(p) 9. xxx =range(0, 6) 10. print (xxx) 11. list(xxx)   Loop in a loop, dream in a dream, are we in simulation inside a simulation?   1. for i in range(4): 2. for j in range(4): 3. if j > i: 4. break 5. print((i, j))   It goes through list and double all values but print it and not change it.   1. # this first kind of for-loop goes through a list 2. foreachvalueinthe\_count: 3. print(f"This is count {eachvalue\*2}")   simple loop again, I keep bringing simple code to make sure you are not lost!   1. for i in range(0, len(the\_count)): 2. print (the\_count[i],i)   Loop in a loop, dream in a dream... is this real or this is your code in dream?   1. for i in range(4): 2. for j in range(4): 3. # if j > i: 4. # break 5. print((i, j))   Or in a loop...   1. sum = 0 2. for i in range(10): 4. # % is the modulo operator 5. if i % 3 == 0 or i % 5 == 0: 6. sum =sum + i 7. print ("i",i) 8. print ("sum",sum)   Step size changed...   1. for i in range(5,-5,-1): 2. if i >=-2: 3. print ('Non-negative') 4. else: 5. print('Negative')   Printing number triangle   1. def line(n): 2. triangle = '' 3. for i in range(1, n+1): 4. triangle = triangle + (str(i)) 5. print(triangle) 6. i+=1   Pyramid world:   1. defhalf\_pyramid(rows): 2. print('Half pyramid...\n') 3. for i in range(rows): 4. print('\*' \* (i+1)) 6. deffull\_pyramid(rows): 7. print('\nFull pyramid...\n') 8. for i in range(rows): 9. print(' '\*(rows-i-1) + '\*'\*(2\*i+1)) 11. definverted\_pyramid(rows): 12. print('\nInverted pyramid...\n') 13. for i in reversed(range(rows)): 14. print(' '\*(rows-i-1) + '\*'\*(2\*i+1)) 16. half\_pyramid(5) 17. full\_pyramid(5) 18. inverted\_pyramid(5)   Prime Number and nested loops:  # First is to search prime numbers -  #for this we need to use a code to see if number is divisible by itself and no other   1. defis\_prime(number): 2. # if number is equal to or less than 1, return False 3. if number <= 1: 4. returnFalse 6. for x in range(2, number): 7. # if number is divisble by x, return False 8. ifnot number % x: 9. returnFalse 10. returnTrue 12. output = [] 13. listofn = [262, 102, 23, 164, 96] 14. numtotest = 30 16. # this the main loop - it goes through each number 17. # once we get access to each then add things and also for prime number we create another function and check 19. defget\_divisors(listofn, numtotest): 21. for i inlistofn: 22. print (i) 23. sumofn = 0 24. #This loop converts the elements like 262 into string and then sums them up 25. fornumin list(str(i)): 26. sumofn = sumofn + int(num) 27. # this is the sum of the digits of the number 28. # if checking for both conditions 29. if (numtotest % sumofn ==0) and (is\_prime(sumofn) == True): 30. #adding to the empty list 31. output.append(i) 32. return output |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |