17thsept_assignment

September 1, 2024

1 FOR LOOP:

[1]:
```
#Q1.Write a Python program to print numbers from 1 to 10 using a for loop.
for i in range(1,11):
    print(i)
```

```
1
2
3
4
5
6
7
8
9
10
```

[ ]:
```
#Q2.Explain the difference between a for loop and a while loop in Python.

FOR LOOP- It is used to perform a loop sequentially multiple times according to␣
 ↪our command and no. of iteration is known in advance
WHILE LooP- It is used to perform loop but only till the condition comes true.␣
 ↪and here no. of iteration is not known
```

[4]:
```
#Q3.Write a Python program to calculate the sum of all numbers from 1 to 100␣
 ↪using a for loop.
sum_=0
for i in range(1,101):
    sum_+=i
sum_
```

[4]:
```
5050
```

[21]:
```
#Q4.How do you iterate through a list using a for loop in Python?
# first we assign a variable(iterator or loop variable) and using 'in' to make␣
 ↪it-
# -run through the loop according to given condition
```

```
LIST = [1,2,3,4,5]

for variable in LIST:
    print(variable)
```

```
1
2
3
4
5
```

[9]:
```
#Q5.Write a Python program to find the product of all elements in a list using␣
 ↪a for loop.
a=[1,2,3,4]
product_=1
for i in a:
    product_*= i
product_
```

[9]:
```
24
```

[10]:
```
#Q6.Create a Python program that prints all even numbers from 1 to 20 using a␣
 ↪for loop.
for i in range(2,21,2):
    print(i)
```

```
2
4
6
8
10
12
14
16
18
20
```

[20]:
```
#Q7.Write a Python program that calculates the factorial of a number using a␣
 ↪for loop.
fact_=1
n=int(input('enter the number'))
for i in range (1,n+1):
    fact_*=i
fact_
```

```
enter the number 5
```

[20]:
```
120
```

[27]:
```
#Q8.How can you iterate through the characters of a string using a for loop in␣
 ↪Python?
string = "Hello, World!"
for VARIABLE in string:
    print(VARIABLE)
# first we assign a variable(iterator or loop variable) and using 'in' to make␣
 ↪it-
# -run through the loop according to given condition
```

```
H
e
l
l
o
,

W
o
r
l
d
!
```

[11]:
```
#Q9.Write a Python program to find the largest number in a list using a for␣
 ↪loop.
l=[3,4,6,2,1]
l1=sorted(l)
for i in l1:
    if len(l1)>0:
        i=l1[-1]
print(l1)
print(i)
```

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

[4]:
```
#Q10.Create a Python program that prints the Fibonacci sequence up to a␣
 ↪specified limit using a for loop.
```

```
def find_fib(n):
    a,b=0,1
    for i in range(n):
```

None

2 MAP QUESTIONS:

[ ]:
```
#Q1.Explain the purpose of the `map()` function in Python and provide an␣
 ↪example of how it can be used to apply a function to each element of an␣
 ↪iterable.
```

[21]:
```
#Q2.Write a Python program that uses the `map()` function to square each␣
 ↪element of a list of numbers.
l=[1,2,3,4]
def square_fx(i):
       return i*i
square_=list(map(square_fx,l))
square_
```

[21]: [1, 4, 9, 16]

[ ]:
```
#Q3.How does the `map()` function differ from a list comprehension in Python,␣
 ↪and when would you choose one over the other?
```

[20]:
```
#Q4.Create a Python program that uses the `map()` function to convert a list of␣
 ↪names to uppercase.
l=['ayush','bruno']
def upper(i):
    return i.upper()
req_list=list(map(upper,l))
req_list
```

[20]: ['AYUSH', 'BRUNO']

[24]:
```
#Q5.Write a Python program that uses the `map()` function to calculate the␣
 ↪length of each word in a list of strings.
l=['ayush','bruno']
def length_(i):
    return len(i)
req_result= list(map(length_,l))
req_result
```

[24]: [5, 5]

[29]:
```
#Q6.How can you use the `map()` function to apply a custom function to elements␣
 ↪of multiple lists simultaneously in Python?
l=['ayush','bruno','food']
l1=['1122','22','11']
l2=['wake','up','sid']
def length_(i,x,y):
    return len(i),len(x),len(y)
req_result= list(map(length_,l,l1,l2))
req_result
# ask this
```

[29]:
```
[(5, 4, 4), (5, 2, 2), (4, 2, 3)]
```

[36]:
```
#Q7.Create a Python program that uses `map()` to convert a list of temperatures␣
 ↪from Celsius to Fahrenheit
l=[27,30,45]
def c_to_f(i):
    return i* 9/5 +32
req_list=list(map(c_to_f,l))
req_list
```

[36]:
```
[80.6, 86.0, 113.0]
```

[42]:
```
#Q8.Write a Python program that uses the `map()` function to round each element␣
 ↪of a list of floating-point numbers to the nearest integer.
l=[9/4,5/2,7/6]
def round_off(i):
    return round(i)
req_list=list(map(round_off,l))
req_list
```

[42]:
```
[2, 2, 1]
```

3 REDUCE QUESTIONS

[ ]:
```
# Q1.What is the `reduce()` function in Python, and what module should you␣
 ↪import to use it? Provide an example of its basic usage.
```

[54]:
```
#Q2.Write a Python program that uses the `reduce()` function to find the␣
 ↪product of all elements in a list.
from functools import reduce
def product(a,b):
    return a*b
l=[1,2,3,4,5,6]
req_number=reduce(product,l)
req_number
```

[54]: 720

[52]:
```
#Q3.Create a Python program that uses `reduce()` to find the maximum element in␣
 ↪a list of numbers.
from functools import reduce
def find_max(x,y):
    return x if x>y else y
l=[1,2,3,4,5,6]
req_number=reduce(find_max,l)
req_number
```

[52]: 6

[70]:
```
#Q4.How can you use the `reduce()` function to concatenate a list of strings␣
 ↪into a single string?
def add_list(a,b):
    return a+b
l=['a','y','u','s','h']
req_list=reduce(add_list,l)
req_list
```

[70]: 'ayush'

[73]:
```
# Q5.Write a Python program that calculates the factorial of a number using the␣
 ↪`reduce()` function.
def fact_(a,b):
    return a*b
n=5
fact_5=reduce(fact_,range(1,n+1))
fact_5
```

[73]: 120

[3]:
```
#Q6.Create a Python program that uses `reduce()` to find the GCD (Greatest␣
 ↪Common Divisor) of a list of numbers.
from functools import reduce
import math
def gcd_(x,y):
    return math.gcd(x,y)
l=[16,48,56,8]
req_number=reduce(gcd_,l)
req_number
```

[3]: 8

[5]:
```
#Q7.Write a Python program that uses the `reduce()` function to find the sum of␣
 ↪the digits of a given number.
from functools import reduce
def count_digit(x,y):
    return int(x)+int(y)

n=3767
a=str(n)    #cz integers are not iterables
req_number=reduce(count_digit,a)
req_number
```

[5]: 23

4 FILTER QUESTIONS

[ ]:
```
# Q1.Explain the purpose of the `filter()` function in Python and provide an␣
 ↪example of how it can be used to filter elements from an iterable.
```

[9]:
```
# Q2.Write a Python program that uses the `filter()` function to select even␣
 ↪numbers from a list of integers.
l=[1,2,3,4,5,6]
even_no=list(filter(lambda a:a%2==0,l))
even_no
```

[9]: [2, 4, 6]

[11]:
```
#Q3.Create a Python program that uses the `filter()` function to select names␣
 ↪that start with a specific letter from a list of strings.
l1=['ayush','analyst','scientist']
req_list=list(filter(lambda a:a.startswith('s'),l1))
req_list
```

[11]: ['scientist']

[ ]:
```
#Q4.Write a Python program that uses the `filter()` function to select prime␣
 ↪numbers from a list of integers.
```

[12]:
```
#Q5.How can you use the `filter()` function to remove None values from a list␣
 ↪in Python?
l = [1,None,2,None,3,None]

req_values = list(filter(lambda x: x is not None, l))

print("Filtered values:", req_values)
```

```
Filtered values: [1, 2, 3]
```

[14]:
```
#Q6.Create a Python program that uses `filter()` to select words longer than a␣
 ↪certain length from a list of strings.
l=['pw','pwskills','datascience','data-analyst']
req_word=list(filter(lambda x: len(x)>8,l)) #we have to give a certain length,␣
 ↪here it is 8.
req_word
```

[14]:
```
['datascience', 'data-analyst']
```

[17]:
```
# Q7.Write a Python program that uses the `filter()` function to select␣
 ↪elements greater than a specified threshold from a list of values.
L=[22,34,27,41,45]
req_number=list(filter(lambda x: x>25,L)) #here 25 is the threshold value that␣
 ↪we have given
req_number
```

[17]:
```
[34, 27, 41, 45]
```

5 Recurssion questions

[ ]:
```
# Q1.Explain the concept of recursion in Python. How does it differ from␣
 ↪iteration?
```

[2]:
```
# Q2.Write a Python program to calculate the factorial of a number using␣
 ↪recursion
def fact_(n):
        if n==0:
            return 1
        else:
            return n * fact_(n-1)
n=int(input('enter any number you want to find factoril of'))
print('factorial of',n,'is',fact_(n))
```

```
enter any number you want to find factoril of 5
factorial of 5 is 120
```

[5]:
```
#Q3.Create a recursive Python function to find the nth Fibonacci number.
def fib_number(n):
    if n<=0:
        return 0
    elif n==1:
        return 1
    else:
```

```
        return fib_number(n-1)+fib_number(n-2)
n=int(input('enter the position'))
print('Number at',n,'position is',fib_number(n))
```

```
enter the position 8
```

```
Number at 8 position is 21
```

[9]:
```
#Q4.Write a recursive Python function to calculate the sum of all elements in a␣
 ↪list.
def find_sum(LIST):
    if len(LIST)==0:
            return 0
    else:
        return LIST[0]+find_sum(LIST[1:])
l=[1,2,3,4,5]
print('Sum of all elements in given list is:',find_sum(l))
```

```
Sum of all elements in given list is: 15
```

[ ]:
```
#Q5.How can you prevent a recursive function from running indefinitely, causing␣
 ↪a stack overflow error?
```

[10]:
```
#Q6.Create a recursive Python function to find the greatest common divisor␣
 ↪(GCD) of two numbers using the Euclidean algorithm.
def find_gcd(a,b):
    if b==0:
        return a
    else:
        return find_gcd(b,a%b)
n1=int(input('enter the first number'))
n2=int(input('enter the second number'))
print('The greatest common divisor of', n1,'and',n2,'is',find_gcd(n1,n2))
```

```
enter the first number 16
enter the second number 32
```

```
The greatest common divisor of 16 and 32 is 16
```

[11]:
```
#Q7.Write a recursive Python function to reverse a string.
def reverse(a):
    if len(a)==0:
        return a
    else:
        return reverse(a[1:]) + a[0]

s1 = input("Enter a string: ")
```

```
req_str = reverse(s1)
print("Reversed string:", req_str)
```

```
Enter a string:  ayush
```

```
Reversed string: hsuya
```

[3]:
```
#Q8.Create a recursive Python function to calculate the power of a number (x^n).
def find_power(a,n):
    if n==0:
        return 1
    elif n==1:
        return a
    else:
        return a*find_power(a,n-1)
n1=int(input('enter the number'))
n2=int(input('enter the power'))
req_number=find_power(n1,n2)
req_number
```

```
enter the number 5
enter the power 3
```

[3]:
```
125
```

[13]:
```
#Q9.Write a recursive Python function to find all permutations of a given␣
 ↪string.
from itertools import permutations

def find_permutations(a):
    if len(a)==0:
        return a
    else:
        return [''.join(i) for i in permutations(a)]

s1 = "pws"
req_permutations=list(find_permutations(s1))
print(req_permutations)
```

```
['pws', 'psw', 'wps', 'wsp', 'spw', 'swp']
```

[7]:
```
#Q10.Write a recursive Python function to check if a string is a palindrome.
def reverse(a):
     if a=="":
        return""
     elif len(a)==0:
        return a
```

```
     else:
        return reverse(a[1:]) + a[0]

def check_palindrome(a):
    if len(a)==0:
        return a
    elif a==reverse(a):
            return True
    else:
        return False
a='Able was I ere I saw Elba'
req_result=check_palindrome(a.lower())
req_result
```

[7]: True

```
[14]: #Q11.Create a recursive Python function to generate all possible combinations␣
       ↪of a list of elements.
      from itertools import permutations

      def find_permutations(a):
          if len(a)==0:
              return a
          else:
              return [''.join(i) for i in permutations(a)]

      L1=['2','4','6']
      req_permutations=list(find_permutations(L1))
      print(req_permutations)
```

```
['246', '264', '426', '462', '624', '642']
```

6 Lambda Functions and Higher-Order Functions

```
[ ]: # Q1.What are lambda functions in Python, and when are they typically used?
```

```
[3]: # Q2.Write a Python program that uses lambda functions to sort a list of tuples␣
      ↪based on the second element.
     l1 = [(2,4),(3,1),(7,8),(5,6)]

     req_tuple=sorted(l1 , key=lambda x: x[1])

     print(req_tuple)
```

```
[(3, 1), (2, 4), (5, 6), (7, 8)]
```

[1]:
```
#Q3.Explain the concept of higher-order functions in Python, and provide an␣
 ↪example
# Example 1: Using a higher-order function as an argument

# Define a higher-order function 'apply_operation'
def apply_operation(operation, x, y):
    return operation(x, y)

# Define some simple operations as functions
def add(x, y):
    return x + y

def subtract(x, y):
    return x - y

def multiply(x, y):
    return x * y

def divide(x, y):
    if y == 0:
        return "Cannot divide by zero"
    return x / y

# Use 'apply_operation' to perform different operations
result1 = apply_operation(add, 5, 3)          # Adds 5 and 3
result2 = apply_operation(subtract, 8, 2)     # Subtracts 2 from 8
result3 = apply_operation(multiply, 4, 6)     # Multiplies 4 and 6
result4 = apply_operation(divide, 10, 2)      # Divides 10 by 2

print("Result 1:", result1)
print("Result 2:", result2)
print("Result 3:", result3)
print("Result 4:", result4)

# Example 2: Using a higher-order function as a return value

# Define a higher-order function 'get_multiplier'
def get_multiplier(factor):
    def multiplier(x):
        return x * factor
    return multiplier

# Create a multiplier function for a specific factor
double = get_multiplier(2)
triple = get_multiplier(3)

# Use the multiplier functions
```

```
result5 = double(5)    # Doubles 5
result6 = triple(7)    # Triples 7

print("Result 5:", result5)
print("Result 6:", result6)
```

```
Result 1: 8
Result 2: 6
Result 3: 24
Result 4: 5.0
Result 5: 10
Result 6: 21
```