Assignment3

August 18, 2023

Q1. Which keyword is used to create a function? Create a function to return a list of odd numbers in the range of 1 to 25.

ANS: The def keyword is used to create, (or define) a function.

[12]:
```
def odd_no():
    l1=[]
    for i in range(1,25):
        if i%2!=0:
            l1.append(i)
    return l1
```

[13]:
```
odd_no()
```

[13]:
```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23]
```

[ ]:
```

```

Q2. Why *args and **kwargs is used in some functions? Create a function each for *args and **kwargs to demonstrate their use.

ANS: *ARGS: The special syntax* args in function definitions in Python is used to pass a variable number of arguments to a function. It is used to pass a non-keyworded, variable-length argument list.

The syntax is to use the symbol * to take in a variable number of arguments; by convention, it is often used with the word args. What *args allows you to do is take in more arguments than the number of formal arguments that you previously defined. With* args, any number of extra arguments can be tacked on to your current formal parameters (including zero extra arguments).

KWARGS: The special syntax kwargs in function definitions in Python is used to pass a keyworded, variable-length argument list. We use the name kwargs with the double star. The reason is that the double star allows us to pass through keyword arguments (and any number of them).

A keyword argument is where you provide a name to the variable as you pass it into the function. One can think of the kwargs as being a dictionary that maps each keyword to the value that we pass alongside it. That is why when we iterate over the kwargs there doesn't seem to be any order in which they were printed out.

[14]:
```
#a function for *args to demonstrate its use.

def fun(*args):
    sum=0
    for i in args:
        sum+=i
    print('sum:',sum)
```

[15]:
```
fun(12,2)
fun(2,55,7)
fun(5,6,8,7)
```

```
sum: 14
sum: 64
sum: 26
```

[16]:
```
#a function for **kwargs to demonstrate its use.

def makeSentence(**words):
    sentence=''
    for word in words.values():
        sentence+=word
    return sentence

print('Sentence:', makeSentence(a='Kwargs ',b='are ', c='awesome!'))
```

```
Sentence: Kwargs are awesome!
```

[ ]:
```

```

Q3. What is an iterator in python? Name the method used to initialise the iterator object and the method used for iteration. Use these methods to print the first five elements of the given list [2, 4, 6, 8, 10, 12, 14, 16, 18, 20].

ANS: An iterator is an object that allows you to iterate over collections of data, such as lists, tuples, dictionaries, and sets.

Instantiating the Iterator in Python

The iterator consists of two methods:

iter() and next().

Iter() method is used to initialize the iterator object so that the instance of this object can be used for iterating.

The next() method is used to iterate over the iterable objects. The next() method returns the next value in the iterable object.

[17]:
```
list=[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

count=0
for i in list:
    if count<5:
        print(i)
        count+=1
```

```
2
4
6
8
10
```

[ ]:

Q4. What is a generator function in python? Why yield keyword is used? Give an example of a generator function.

ANS: A generator function in Python is defined like a normal function, but whenever it needs to generate a value, it does so with the yield keyword rather than return. If the body of a def contains yield, the function automatically becomes a Python generator function.

The yield statement is used to produce a value, and the generator function can be resumed from where it left off the next time next() is called on the generator.

[18]:
```
#the fibonacci series program using generator yield

def fibonacci():
    x, y = 0, 1
    while True:
        yield x
        x, y = y, x + y

# Accept input from the user
n = int(input("Input the number of Fibonacci numbers you want to generate? "))

print("Number of first ",n,"Fibonacci numbers:")
fib = fibonacci()
for _ in range(n):
    print(next(fib),end=" ")
```

```
Input the number of Fibonacci numbers you want to generate?  20

Number of first  20 Fibonacci numbers:
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181
```

[ ]:

Q5. Create a generator function for prime numbers less than 1000. Use the next() method to print the first 20 prime numbers.

ANS:

[19]:
```
def primes():
    """Create a generator function for prime numbers less than 1000."""
    yield 2
    primes_list = [2]
    for i in range(3, 1000):
        is_prime = True
        for prime in primes_list:
            if i % prime == 0:
                is_prime = False
                break
        if is_prime:
            primes_list.append(i)
            yield i

prime_gen = primes()
# the next() method to print the first 20 prime numbers.
for i in range(20):
    print(next(prime_gen))
```

```
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
```

[ ]:
```

```

Q6. Write a python program to print the first 10 Fibonacci numbers using a while loop.

[20]:
```
# This program prints the first 10 Fibonacci numbers using a while loop.
def fibonacci(n):
    a,b=0,1
    count=0
    while count<n:
        print(a)
        a,b=b,a+b
        count+=1
```

[21]:
```
fibonacci(10)
```

```
0
1
1
2
3
5
8
13
21
34
```

[ ]:
```

```

Q7. Write a List Comprehension to iterate through the given string: 'pwskills'. Expected output: ['p', 'w', 's', 'k', 'i', 'l', 'l', 's']

[22]:
```
# List Comprehension to iterate through the given string: 'pwskills'.
string = 'pwskills'
output = [char for char in string]
print(output)
```

```
['p', 'w', 's', 'k', 'i', 'l', 'l', 's']
```

[ ]:
```

```

Q8. Write a python program to check whether a given number is Palindrome or not using a while loop.

[23]:
```
n = int(input("Please give a number : "))
def reverse(num):
    if num<10:
      return num
    else:
      return int(str(num%10) + str(reverse(num//10)))
def isPalindrome(num):
    if num == reverse(num):
        return 1
```

```
    return 0
if isPalindrome(n) == 1:
    print("Given number is a palindrome")
else:
    print("Given number is a not palindrome")
```

```
Please give a number :  258852
Given number is a palindrome
```

[ ]:

Q9. Write a code to print odd numbers from 1 to 100 using list comprehension. Note: Use a list comprehension to create a list from 1 to 100 and use another List comprehension to filter out odd numbers.

[24]:
```
odd_numbers = [n for n in range(1, 101) if n % 2 != 0]
for n in odd_numbers:
    print(n)
```

```
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41
43
45
47
49
51
53
55
```

```
57
59
61
63
65
67
69
71
73
75
77
79
81
83
85
87
89
91
93
95
97
99
```

[ ]: