

PW - ASSIGNMENT

DATE – 3rd Feb

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.

Answer – 1: ‘def’ keyword is used to create a function.

def odd\_no(n):  
 l = []  
 for i in range(1,n):  
 if i % 2 == 1:  
 l.append(i)  
 return l  
  
input1 = odd\_no(25)  
print(input1)

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

Answer – 2:

\*args is used in functions to pass any number of variables to the function so it does not show any error. It does not provide a limit to number of parameters on which a function is working.

\*\*kwargs is used to return a dictionary that is output in key value pairs on any input received in form of key value pair.

def test(\*args):  
 return args  
  
a = test(1,2,3,4, ["hello", "goodmorning"], (2,3,1))  
print(a)

def test(\*\*kwargs):  
 return kwargs  
  
a = test(b = "Good", c = 2, d = 4, e = "hello")  
print(a)

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].

Answer – 3: An iterator is an object that contains a countable number of values. An iterator is an object that can be iterated upon, meaning that you can traverse through all the values. Technically, in Python, an iterator is an object which implements the iterator protocol.

‘next’ and ‘iter’ are methods used to initialize the iterator object and used for iteration.

a = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]  
iterator = iter(a)  
print(next(iterator))  
print(next(iterator))  
print(next(iterator))  
print(next(iterator))  
print(next(iterator))

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

Answer – 4:

In Python, a generator is a function that returns an iterator that produces a sequence of values when iterated over. Generators are useful when we want to produce a large sequence of values, but we don't want to store all of them in memory at once.

In simpler words, the yield keyword will convert an expression that is specified along with it to a generator object and return it to the caller. Hence, if you want to get the values stored inside the generator object, you need to iterate over it.

def fib(n):  
 a,b = 0,1  
 for i in range(n):  
 yield a  
 a,b = b, a+b  
  
for i in fib(10):  
 print(i)