Assignment 04: 03 Feb 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.
Q2. Why "args and **kwargs are used in some functions? Create a function each for "args and **kwargs to demonstrate their use.
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].
Q4. What is a generator function in python? Why is the yield keyword used? Give an example of a generator function.
Q5. Create a generator function for prime numbers less than 1000. Use the next() method to print the first 20 prime numbers.
Q6. Write a python program to print the first 10 Fibonacci numbers using a while loop.
Q7. Write a List Comprehension to iterate through the given string: 'pwskills'.
Expected output: ['p', 'w', 's', 'k', 'i', 'l', 's']
Q8. Write a python program to check whether a given number is Palindrome or not using a while loop.
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.

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.

```
# create function & return list
def odd(start,end):
    for start in range (start,end,1):
        if start%2!=0 :
            print(start)
start,end = 1,20
odd(i,end)
```

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

Ans:
In Python, we can pass a variable number of arguments to a function using special symbols. There are two special symbols:

1. *args (Non Keyword Arguments)
2. **kwargs (Keyword Arguments)

We **use *args and **kwargs as an argument** when we are unsure about the number of arguments to pass in the functions.

S.No.	*args	*kwargs
Difference		
1	*args should come before **kwargs	**kwargs should come after *args
2	It passes a variable number of non-keyworded arguments and on which operation of the tuple can be performed.	It passes a variable number of keyword arguments to the dictionary to function on which operation of a dictionary can be performed.
3	This is used to pass n number of positional arguments to the function body.	This is used to pass n numbers of keyword arguments to the function body.
4	Single asterisk (*) representation, i.e function will accept only positional arguments.	Double asterisk (**) representation i.e this function will accept only keyword arguments.
5	The data type of value going inside this function is tuple.	The data type of value going inside this function is a dictionary.
Similarities		
1	Both make functions flexible.	
2	Both are special python keywords which allow functions to take variable length arguments.	
3	The words “args” and “kwargs” are only a convention, you can use any name of your choice	

Eg 1: **Using *args**

```
def adder(*num):
    sum = 0
    for n in num:
        sum = sum + n
    print("Sum:",sum)
adder(3,5)
adder(4,5,6,7)
adder(1,2,3,5,6)
```

Eg 2: **Using **kwargs**

```
def intro(**data):
    print("\nData type of argument:",type(data))
    for key, value in data.items():
        print("{} is {}".format(key,value))
intro(Firstname="Sita", Lastname="Sharma", Age=22, Phone=1234567890)
intro(Firstname="John", Lastname="Wood", Email="johnwood@nomail.com", Country="Wakanda",
Age=25, Phone=9876543210)
```

Output

```
Data type of argument: <class 'dict'>
Firstname is Sita
Lastname is Sharma
Age is 22
Phone is 1234567890
```

Data type of argument: <class 'dict'>
Firstname is John
Lastname is Wood
Email is johnwood@nomail.com
Country is Wakanda
Age is 25
Phone is 9876543210

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:
iterable and iterator are different. The main difference between them is, iterable cannot save the state of the iteration, whereas in iterators the state of the current iteration gets saved.

- **Iterable** is an object that one can iterate over. It generates an Iterator when passed to iter() method.
- **Iterator** is an object, which is used to iterate over an iterable object using the __next__() method which returns the next item of the object.

Note: Every iterator is also an iterable, but not every iterable is an iterator in Python.
Iterator in Python
Iterator in Python is an object that is used to iterate over iterable objects like lists, tuples, dicts, and sets. The iterator object is initialised using the iter() method. It uses the next() method for iteration.

1. __iter__(): The iter() method is called for the initialization of an iterator. This returns an iterator object
2. __next__(): The next method returns the next value for the iterable. When we use a for loop to traverse any iterable object, internally it uses the iter() method to get an iterator object, which further uses the next() method to iterate over. This method raises a StopIteration to signal the end of the iteration.

```
# iterator method to print first five element
list = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
iterator = iter(list)
for i in range(0,5,1):
  print(next(iterator))
```

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

Ans:
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.
yield keyword is used **to create a generator function**.

```
# Example of generator function
def generator(n):
    value = 0
    while value < n:
        yield value
```

```
        value += 1
for value in generator(3):
    print(value)
```

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

Ans:

```
# print prime no. from using generator () & next ()
from math import sqrt
def is_prime(n):
    if (n <= 1):
        return False
    if (n == 2):
        return True
    if (n % 2 == 0):
        return False
    i = 3
    while i <= sqrt(n):
        if n % i == 0:
            return False
        i = i + 2
    return True
def prime_generator():
    n = 1
    while True:
        n += 1
        if is_prime(n):
            yield n
generator = prime_generator()
for i in range(1000):
    print(next(generator))
```

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

Ans:

```
# Fibonacci Series in python
n=int(input("Enter frequency of fibonacci series: "))
a,b=0,1
counter=0
while counter<n:
    print (a)
    c=a+b
    a=b
    b=c
    counter=counter+1
```

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

Ans:

```
# iterate string from list comprehension in python
List = []
for character in 'pwskill':
        List.append(character)
print(List)
```

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

Ans:

```
# palindrome number in python
num=int(input("Enter a number: "))
temp=num
rev=0
while(num>0):
    rem=num%10
    rev=rev*10+rem
    num=num//10
if(temp==rev):
    print(temp ,"is a palindrome number")
else:
    print(temp ,"is not a palindrome number")
```

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.

Ans:

```
# print odd no. from list comprehension in python
# step 1
n=int(input("Enter no. of elements: "))
list=[]
for i in range(0,n):
    print("Enter element",i)
    a=int(input())
    list.append(a)
print ("list1= ",list)
#step 2
list1 = [10, 21, 4, 45, 66, 93]
only_odd = [num for num in list1 if num % 2 == 1]
print("Odd list=",only_odd)
```