03_feb_assig

February 8, 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.

The keyword used to create a function in Python is def.

[1]:
```
def  odd_numbers():
    odd_numbers = []
    for i in range(1, 26):
        if i % 2 != 0:
            odd_numbers.append(i)
    return odd_numbers
```

[2]:
```
odd_numbers()
```

[2]:
```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25]
```

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

–> *args is used when we need to give numerous arguments in the function, while **kwargs is used we need to numerous arguments in the form of key - value pairs and we get return value in the form of dictionary

[3]:
```
# passing numerous arguments inside func

def test5(*args):
    return args
```

[4]:
```
test5(1,5,"xyz")
```

[4]:
```
(1, 5, 'xyz')
```

[5]:
```
# passing numerous arguments inside func

def test8(**kargs):
    return kargs
```

[6]:
```
test8(a=5,b="xyz")
```

[6]: {'a': 5, 'b': 'xyz'}

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].

–>An iterator in Python is an object that can be iterated (looped) upon. It implements two methods, **iter**() and **next**(), allowing the object to be used in a for loop or with the next() function.

[7]:
```
mylist = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

# Creating an iterator object
my_iter = iter(mylist)

# Iterating through the first five elements
for i in range(5):
    print(next(my_iter))
```

```
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.

A generator function is a special kind of function in Python that returns a generator iterator, instead of a list, tuple, or other collection. The generator iterator can be used in a for loop or with the next() function to execute the code inside the generator function one step at a time, rather than all at once. This is particularly useful for large sequences of values that you don't want to compute or store all at once in memory.

The yield keyword is used in a generator function to return a value and pause the function's execution. The next time the generator function is called, it will pick up where it left off, instead of starting from the beginning.

[8]:
```
def fibonacci(n):
    a, b = 0, 1
    for i in range(n):
        yield a
        a, b = b, a + b

# Using the generator function
for number in fibonacci(10):
    print(number)
```

```
0
1
1
```

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

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

[9]:
```
def prime_numbers():
    yield 2
    primes = [2]
    for i in range(3, 1000):
        is_prime = True
        for prime in primes:
            if prime > i**0.5:
                break
            if i % prime == 0:
                is_prime = False
                break
        if is_prime:
            primes.append(i)
            yield i

# Using the generator function
prime_gen = 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.

```
[10]: #fibonacci series

      num = 10

      a,b= 0,1
      count = 0

      while (count<num):
          print(a)
          c=a+b
          a=b
          b=c
          count = count+1
```

```
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’]

```
[15]: st =  'pwskills'
      [i for i in st]
```

```
[15]: ['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.

```
[16]: def is_palindrome(n):
          original = n
          reverse = 0
          while n > 0:
              last_digit = n % 10
              reverse = (reverse * 10) + last_digit
              n = n // 10
```

```
    return original == reverse

# Testing the function
number = 121
if is_palindrome(number):
    print(f"{number} is a palindrome")
else:
    print(f"{number} is not a palindrome")
```

```
121 is a palindrome
```

Q9. Write a code to print odd numbers from 1 to 100 using list comprehension.

```
[17]: [i for i in range(1,101) if i%2 !=0]
```

```
[17]: [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]
```

```
[ ]:
```