In [2]:
```
#Q1
```

In [3]:
```
#def keyword is used to create function name
```

In [9]:
```
start = int(input("Enter the start of range:"))
end = int(input("Enter the end of range:"))

# iterating each number in list
for i in range(start, end+1):

    # checking condition
    if i % 2 != 0:
        print(i)
```

```
1
3
5
7
9
11
13
15
17
19
21
23
25
```

In [10]:
```
#Q2


# You can use *args&*kwargs as arguments to a function when you are unsure about
#the number of arguments to pass in the functions.

# with the help of *args no of arguments pass we also give name of person in Args.
```

In [14]:
```
#EX of *args is
def add(*numbers):
  total = 0
  for num in numbers:
        total += num
  return total
print(add(2,3))
print(add(2,3,5))
print(add(2,3,5,7))
print(add(2,3,5,7,9))
```

```
5
10
17
26
```

In [15]:
```
# **kwargs allows us to pass a variable number of keyword arguments to a Python fun
```

In [16]:
```
# with the help of **kwargs output in the form of Key Value pair
# EX of **Kwargs
def test1(**kwargs):
    return kwargs
test1(a=[1,2,3,4],b="sudh",c=23.45)
```

Out[16]:
```
{'a': [1, 2, 3, 4], 'b': 'sudh', 'c': 23.45}
```

In [17]:
```
#Q3
```

In [18]:
```
# In Python, an iterator is an object that allows you to iterate over collections o
# dictionaries,and sets.
```

In [19]:
```
#The Python iterators object is initialized using the iter() method. It uses the ne
```

In [20]:
```
list =[2,4,6,8,10,12,14,16,18,20]
ch_iterator = iter(list)

print(next(ch_iterator))
print(next(ch_iterator))
print(next(ch_iterator))
print(next(ch_iterator))
print(next(ch_iterator))
```

```
2
4
6
8
10
```

In [21]:
```
#Q4
```

In [22]:
```
# Generator function made data and give outcome.With the help of generator function
# optimize the entire memory location.

#yield keyword is used to create a generator function. A type of function that is m
```

In [36]:
```
# Ex of generator function

def fun_generator():
    yield "Hello world!!"
    yield "PW"


obj = fun_generator()

print(type(obj))
print(next(obj))
print(next(obj))
```

```
<class 'generator'>
Hello world!!
PW
```

In [37]:
```
#Q5
```

In [2]:
```
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

def prime_nums_generator():
    n=2
    while True:
        if is_prime(n):
            yield n
        n += 1

# Create the generator object
primes = prime_nums_generator()

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

# Generate and print the first 10 prime numbers
print("First",n,"Prime numbers:")
for _ in range(n):
    print(next(primes))
```

```
First 20 Prime numbers:
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
```

In [3]:
```
#Q6
```

In [4]:
```
def test_fib1():
    a,b=0,1
    while True:
        yield a
        a,b=b,a+b
fib=test_fib1()
for i in range(10):
    print(next(fib))
```

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

In [5]:
```
#Q7
```

In [11]:
```
s='pwskills'
list(map(lambda s:s.lower(),s))
```

Out[11]:
```
['p', 'w', 's', 'k', 'i', 'l', 'l', 's']
```

In [12]:
```
#Q8
```

In [16]:
```
n=int(input("Enter number:"))
temp=n
rev=0
while(n>0):
    dig=n%10
    rev=rev*10+dig
    n=n//10
if(temp==rev):
    print("The number is a palindrome!")
else:
    print("The number isn't a palindrome!")
```

```
The number is a palindrome!
```

In [17]:
```
#Q9
```

In [23]:
```
# Python program to print odd Numbers in a List

# list of numbers
list1 =range(1,101)

only_odd = [i for i in list1 if i % 2 == 1]

print(only_odd)
```

```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 4
3, 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]
```

In [ ]: