MD. ARSHAD </p>

ALISHARMEEN02@GMAIL.COM </p>

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.

In [1]:
```
# def keyward used to create a function
def List():
    a = []
    for i in range(1,25):
        if  i%2!=0:
            a.append(i)
    print(a)
```

In [2]:
```
List
```

Out[2]:
```
<function __main__.List()>
```

In [3]:
```
List()
```

```
[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.

In [4]:
```
# *args -> Is use for take multiple arguments in function.
def name(*args):
    for i in args:
        print(i, end=",")

l = ["mohd","arshad","ali"]
name(l)
```

```
['mohd', 'arshad', 'ali'],
```

In [5]:
```
s = ["a","b","c","d","e","f","g"]
name(s)
```

```
['a', 'b', 'c', 'd', 'e', 'f', 'g'],
```

In [6]:
```
def name(n,*args):
    print(n)
    for i in args:
        print(i, end=",")

l = ["mohd","arshad","ali"]
name("aaaaaaaa",l)
```

```
aaaaaaaa
['mohd', 'arshad', 'ali'],
```

In [7]:
```
def name(n,*a):
    print(n)
    for i in a:
        print(i, end=",")

l = ["d","ad","i"]
name("nnn",l)
```

```
nnn
['d', 'ad', 'i'],
```

In [8]:
```
# *Kwargs -> Is use for take multiple arguments in function like Dict. key-value pair.
def Dict(**kwargs):
    for k,v in kwargs:
        print(k,v)
```

In [9]:
```
l = {"a":12,"b":13,"c":14,"d":15,"e":16,"f":17}
```

In [10]:
```
dict(l)
```

Out[10]:
```
{'a': 12, 'b': 13, 'c': 14, 'd': 15, 'e': 16, 'f': 17}
```

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 is an object which contains a countable number of values and it is used over iterable objects like :- list,tuples, sets, e.t.c.
- Iterators are used mostly to itrate or convert other objects to an iterator using iter() function
- Iterators uses iter() and next() functions.
- Lists, tuples,dictionaries and sets are all iterable objects . They are iterable containers which you can get an iterator from
- Technicliay the iterator protocol, which consist of the methods **iter** () and **next**().
- Every iterator is not a generator.

In [11]:
```
# WITHOUT ITER FUNCTION
l = [2,4,6,8,10,12,14,16,18,20]
for i in range(5):
    print(l[i], end=", ")
```

```
2, 4, 6, 8, 10,
```

In [12]:
```
# WITH ITER FUNCTION
l = [2,4,6,8,10,12,14,16,18,20]
l1 = iter(l)
print(next(l1))
print(next(l1))
print(next(l1))
print(next(l1))
print(next(l1))
```

```
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

- Generators are functions that return a sequence of values. Wee use yield statement to return the value from function.
- Yield statement returns the elements from a generator function into a generator object.

In [13]:
```
def disp(a,b):
    yield a
    yield b
re = disp(100,200)
print(re)
print(type(re))
print(next(re))
print(next(re))
```

```
<generator object disp at 0x00002095C156D60>
<class 'generator'>
100
200
```

In [14]:
```
def show(a,b):
    while a<=b:
        yield a
        a = a+1

re = show(1,5)
print(re)
print(type(re))
print(next(re))
print(next(re))
print(next(re))
print(next(re))
print(next(re))
```

```
<generator object show at 0x00002095C156D00>
<class 'generator'>
1
2
3
4
5
```

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

In [15]:
```
a = []
count = 0
for i in range(1,1000):
    count = 0
    for j in range(1,i+1):
        if i%j==0:
            count= count+1
    if count==2:
        a.append(i)
```

In [16]:
```
print(a)
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997]
```

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

In [17]:
```
a,b = 0,1
counter = 0
while counter<=10:
    print(a,end="|")
    c = a+b
    a = b
    b = c
    counter = counter+1
```

```
0|1|1|2|3|5|8|13|21|34|55|
```

Q7. Write a List Comprehension to iterate through the given string: 'pwskills'.

Expected output: ['p', 'w', 's', 'k', 'i', 'l', 'l', 's']

In [18]:
```
p = "pwskills"
a = [i for i in p]
print(a)
```

```
['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.

In [19]:
```
i = int(input("enter the number here :- "))
rev = 0
x=i
while (i>0):
    rev = (rev*10) + i%10
    i = i//10
if (x==rev):
    print("It's a Palindrome number ")
else:
    print("It's not a palindrome number")
```

```
enter the number here :- 10
It's not a palindrome number
```

In [20]:
```
i = int(input("enter the number here :- "))
rev = 0
x=i
while (i>0):
    rev = (rev*10) + i%10
    i = i//10
if (x==rev):
    print("It's a Palindrome number ")
else:
    print("It's not a palindrome number")
```

```
enter the number here :- 121
It's a Palindrome number
```

In [21]:
```
i = int(input("enter the number here :- "))
rev = 0
x=i
while (i>0):
    rev = (rev*10) + i%10
    i = i//10
if (x==rev):
    print("It's a Palindrome number ")
else:
    print("It's not a palindrome number")
```

```
enter the number here :- 12121
It's a Palindrome number
```

In [22]:
```
i = int(input("enter the number here :- "))
rev = 0
x=i
while (i>0):
    rev = (rev*10) + i%10
    i = i//10
if (x==rev):
    print("It's a Palindrome number ")
else:
    print("It's not a palindrome number")
```

```
enter the number here :- 1212121
It's a Palindrome number
```

In [23]:
```
i = int(input("enter the number here :- "))
rev = 0
x=i
while (i>0):
    rev = (rev*10) + i%10
    i = i//10
if (x==rev):
    print("It's a Palindrome number ")
else:
    print("It's not a palindrome number")
```

```
enter the number here :- 12121212121
It's 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

In [24]:
```
# odd numbers
a = [ i for i in range(100) if i%2!=0]
print(a)
```

```
[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]
```

In [25]:
```
# even numbers
a = [ i for i in range(100) if i%2==0]
print(a)
```

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98]
```

Thank You ,That's All </p>