In [28]:
```
# Q1
#The keyword used to create a dunction is = def

def odd_num():
    n=[]
    for i in range(0,25):
        if (i%2!=0):
            n.append(i)
    return n
```

In [29]:
```
odd_num()
```

Out[29]:
```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23]
```

In [36]:
```
#Q2
""" *args in function definitons in python is used  to pass  a variable number of argu
It used to pass a non-keyworded, variable-length argument list."""

""" *kwargs  in function definitions in Python is used to pass a keyworded, variable-le
We use the name kwargs with the double star"""

#Example (*args)

def add_all(*args):
    return sum(args)

add_all(1,4,5,6,7,7,9)


""" Example (**kwargs)"""

def print_kwargs(**kwargs):
    print(kwargs)
print_kwargs(goku=9001,krillin=1500, picoolo=2500)
```

```
{'goku': 9001, 'krillin': 1500, 'picoolo': 2500}
```

In [3]:
```
#Q3
'''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 throug

n Python, an iterator is an object which implements the iterator protocol,
which consist of the methods __iter__() and __next__().'''

my_list=[2,4,6,8,10,12,14,16,18,20]
my_iterator=iter(my_list)

for _ in range(5):
    element=next(my_iterator)
    print(element)
```

```
2
4
6
```

```
8
10
```

In [4]:

```python
#Q4
'''a generator function is a special type of function that generates a sequence of valu
Instead of returning a single value and terminating like a regular function,
a generator function can yield multiple values over time.
Each time the yield keyword is encountered in the function,
the function's state is saved, and the yielded value is returned.
The next time the generator function is called, it resumes from where it left off,
allowing for the generation of a sequence of values without having to store them all in

'''The yield keyword is used in a generator function to define the points at which the
When a value is yielded, the function's state is suspended, and the yielded value is re

#Example of Generator Function

def even_number(n):
    for i in range(n):
        if i % 2==0:
            yield i
even_gen=  even_number(10)

for num in even_gen:
    print(num)
```

```
0
2
4
6
8
```

In [5]:

```python
#Q5
def prime_numbers():
    primes = []
    num = 2
    while True:
        if all(num % prime != 0 for prime in primes):
            yield num
            primes.append(num)
        num += 1
        if num >= 1000:
            break

# Using the generator function to generate prime numbers
prime_gen = prime_numbers()

# Printing the first 20 prime numbers
for _ in range(20):
    prime = next(prime_gen)
    print(prime)
```

```
2
3
5
7
11
13
17
```

19
23
29
31
37
41
43
47
53
59
61
67
71