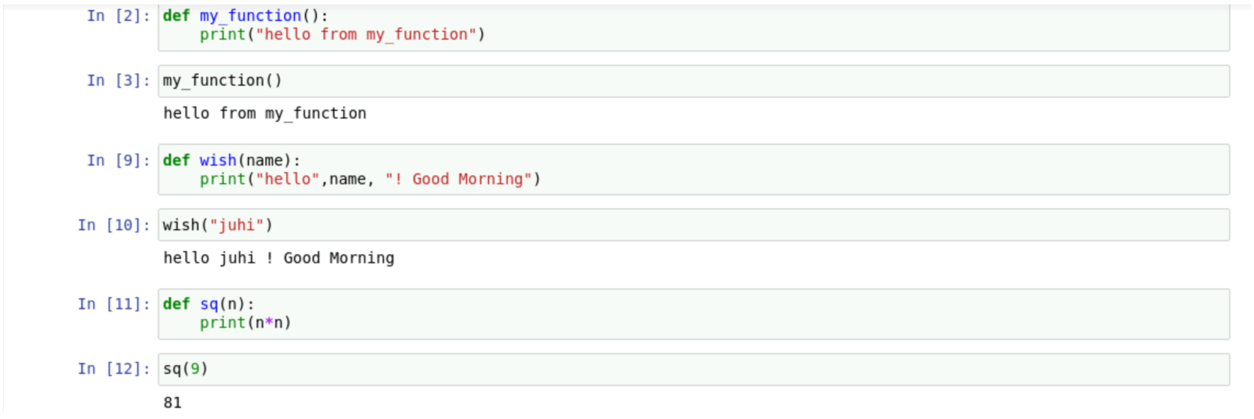
Functions:
Parameterized functions
Functions with return keyword

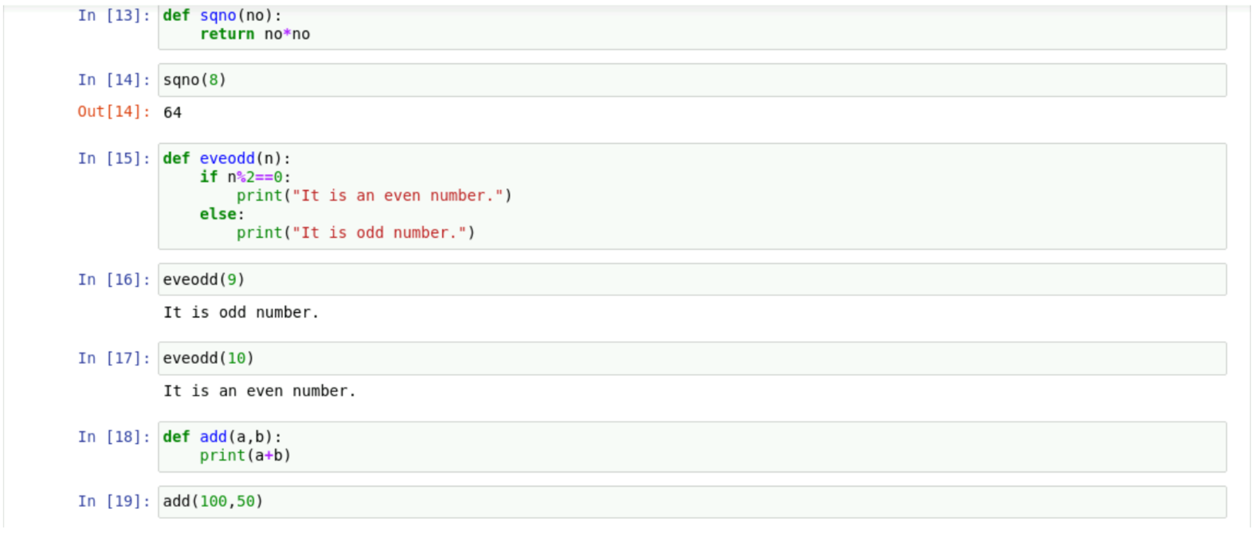
Types of Arguments:

1. Positional Arguments
2. Keyword Arguments
3. Default Arguments
4. Variable length Arguments

Recursive Functions: Function that calls itself, eg. Fibonacci, Factorial etc.
Decorator Function: It takes in a function and returns it by adding some functionality.
Generator Function: A generator function is yield keyword to yield a sequence of values one at a time, instead of returning all at once
List Comprehension
Lambda/Anonymous Function: no name for the function
Filter, Map
Class (consists of variables and methods(attributes/functions)



```
In [2]: def my_function():
            print("hello from my_function")

In [3]: my_function()

        hello from my_function

In [9]: def wish(name):
            print("hello",name, "! Good Morning")

In [10]: wish("juhi")

         hello juhi ! Good Morning

In [11]: def sq(n):
             print(n*n)

In [12]: sq(9)

         81
```



```
In [13]: def sqno(no):
             return no*no

In [14]: sqno(8)

Out[14]: 64

In [15]: def eveodd(n):
             if n%2==0:
                 print("It is an even number.")
             else:
                 print("It is odd number.")

In [16]: eveodd(9)

         It is odd number.

In [17]: eveodd(10)

         It is an even number.

In [18]: def add(a,b):
             print(a+b)

In [19]: add(100,50)
```

In [20]:
```
def sub(a,b):
    return a-b
```

In [21]:
```
sub(10,5)
```

Out[21]:
```
5
```

In [22]:
```
sub(5,19)
```

Out[22]:
```
-14
```

```
Keyword Argument
```

In [23]:
```
sub(b=5,a=19)
```

Out[23]:
```
14
```

In [24]:
```
def wish(name,msg):
    print("hello",name,msg)
```

In [25]:
```
wish(msg="Good Morning", name="Juhi")
```

```
hello Juhi Good Morning
```

```
Variable Length Argument
```

In [42]:
```
def add(*no):
    sum=0
    for i in no:
        sum=sum+i
        print(i)
    print(sum)
```

In [43]:
```
add(1,2)
```

```
1
2
3
```

In [44]:
```
add(10,100,11,1111)
```

```
10
100
11
1111
1232
```

In [47]:
```
def info(**args):
    for i,j in args.items():
        print(i,j)
```

In [48]:
```
info(Name="John",Age=30,City="Mumbai")
```

```
Name John
Age 30
City Mumbai
```

In [49]:
```
info(Name="John",Age=30,City="Mumbai",Dept="IT")
```

```
Name John
Age 30
City Mumbai
Dept IT
```

*args and **kwargs to provide flexibility in adding and searching for equipment and sites. Here are the tasks you need to complete: Create a Python script that defines empty lists for drilling equipment and drilling sites.

Implement a function add_equipment that takes the following parameters:

equipment_type (string): The type of equipment being added. *args (tuple): Additional details about the equipment (e.g., model, power, capacity). **kwargs (dictionary): Additional attributes of the equipment (e.g., vendor, power source). The function should create a dictionary representing the equipment, including its type, details (from *args), and attributes (from **kwargs). Then, it should append this dictionary to the drilling_equipment list.

In [55]:
```
DrillingEquipment=[]
DrillingSites=[]
```

In [59]:
```
def add_equipment(eq_type,*args,**kwargs):
    dict={
        "equipement":eq_type,
        "details":args,
        "attributes":kwargs
    }
    DrillingEquipment.append(dict)
```

In [60]:
```
add_equipment("Pump","Model X", 100,10,vendor="A")
```

In [61]:
```
print(DrillingEquipment)
```

```
[<class 'dict'>, {'equipement': 'Pump', 'details': ('Model X', 100, 10), 'attributes': {'vendor': 'A'}}]
```

In [63]:
```
def factorial(n):
    if n==0:
        return 1
    else:
        return n*factorial(n-1)
factorial(5)
```

Out[63]:
```
120
```

In [64]:
```
def fibonacci(n):
    if n<=1:
        return n
    else:
        return fibonacci(n-1)+fibonacci(n-2)
fibonacci(5)
```

Out[64]:
```
5
```

In [65]:
```
def div(a,b):
    if a<b:
        a,b=b,a
    c=a/b
    print(c)
div(5,10)
```

```
2.0
```

In [70]:
```
def wish(name):
    print("Hello", name,"Good Morning")
```

In [78]:
```
def decor(func):
    def inner(name):
        if name=="Robert":
            print("Hello", name,"Bad Morning")
        else:
            return func(name)
    return inner

result=decor(wish)
```

In [80]:
```
result("Robert")
```

```
Hello Robert Bad Morning
```

Generator Function

```
In [91]: def count_up_to(n):
             i=1
             while(i<=n):
                 yield i
                 i=i+1
```

```
In [92]: for i in count_up_to(10):
             print(i)
```

```
1
2
3
4
5
6
7
8
9
10
```

```
In [94]: def fib(n):
             a,b=0,1
             while n>0:
                 yield a
                 a,b=b,a+b
                 n=n-1
```

```
In [96]: for i in fib(10):
             print(i)
```

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

List Comprehension

```
In [1]: num=[1,2,3,4,5]
        sq=[x**2 for x in num]
        print(sq)
```

```
[1, 4, 9, 16, 25]
```

```
In [4]: num=[1,2,3,4,5,6,7,8,9]
        even=[x for x in num if x%2==0]
        print(even)
```

```
[2, 4, 6, 8]
```

```
In [6]: list=["Python","Sql","Azure","Power BI"]
        f_list=[s[0] for s in list]
        print(f_list)
```

```
['P', 'S', 'A', 'P']
```

```
In [9]: list=["python","sql","azure","power BI","databricks"]
        u_list=[s[0].upper()+s[1:] for s in list]
        print(u_list)
```

```
['Python', 'Sql', 'Azure', 'Power BI', 'Databricks']
```

```
In [20]: sentence="the quick brown fox jumps over the lazy dog"
         sen=sentence.split(" ")
         x=[[s.upper(),len(s)] for s in sen]
         print(x)

[['THE', 3], ['QUICK', 5], ['BROWN', 5], ['FOX', 3], ['JUMPS', 5], ['OVER', 4], ['THE', 3], ['LAZY', 4], ['DOG',
3]]
```

Lambda/Anonymous Function

```
In [22]: f=lambda i:i*i

In [23]: f(3)

Out[23]: 9

In [24]: s=lambda a,b:a+b
         print("the sum is :",s(10,30))

the sum is : 40

In [26]: l=lambda a,b:a if a>b else b
         l(10,90)

In [26]: l=lambda a,b:a if a>b else b
         l(10,90)

Out[26]: 90

In [1]: def evenodd(n):
            if(n%2==0):
                return True
            else:
                return False
        num=[1,2,3,4,5,6,7,8,9]
        res=list(filter(evenodd,num))
        print(res)

[2, 4, 6, 8]

In [2]: l2=list(filter(lambda x:x%2==0,num))
        print(l2)

[2, 4, 6, 8]

In [4]: l=[10,11,12,13,14,15]
        def squareit(n):
            return n**2
        sq=list(map(squareit,l))
        print(sq)

[100, 121, 144, 169, 196, 225]

In [6]: l2=list(map(lambda x:x**2, l))
        print(l2)

[100, 121, 144, 169, 196, 225]

In [8]: people=[{'name':'Alice', 'age':30},
                {'name':'Bob', 'age':25},
                {'name':'Charlie', 'age':35}]
        res=sorted(people,key=lambda i: i['age'])
        print(res)

[{'name': 'Bob', 'age': 25}, {'name': 'Alice', 'age': 30}, {'name': 'Charlie', 'age': 35}]

In [9]: class computer:
            def config(self):
                print("i5","16GB")

In [10]: comp1=computer()
         comp1.config()

i5 16GB
```