Nested Dictionary

A dictionary can be defined inside dictionary as a value.

Example:

```
>>> emp_data={1:{'ename':'naresh','sal':5000},
...           2:{'ename':'suresh','sal':6000}}
>>> print(emp_data)
{1: {'ename': 'naresh', 'sal': 5000}, 2: {'ename': 'suresh', 'sal': 6000}}
>>> emp_data[1]['ename']
'naresh'
>>> emp_data[1]['sal']
5000
>>> emp_data[2]['ename']
'suresh'
>>> emp_data[2]['sal']
6000
```

What is difference between list, set and dictionary?

List	**Set**	**Dictionary**
List is a sequence data type or ordered collection	Set is a non sequence data type or unordered collection	Dictionary mapping collection
List allows only values	Set allows only values	Dictionary allows key and values
List is index based collection, where reading and writing is done using index	Set is non index based collection	Dictionary is key based collection, where reading and writing is done using key
List allows duplicate values	Set does not allows duplicate values	Dictionary does not allows duplicate keys

		but allows duplicate values
List created using []	Set is created using {value,}	Dictionary is created using {key:value,…}
In application development list is used to group individual objects where duplicates are allowed and reading and writing is done sequentially and randomly	In application development set is used to group individual objects where duplicates are not allowed and allows to perform mathematical operations.	In application development map is used to group individual objects where data is organized in key and values pair.

Sequences	**sets**	**mapping**
List	set	dictionary
Tuple	frozenset	
String		
Range		
Bytes		
Bytesarray		

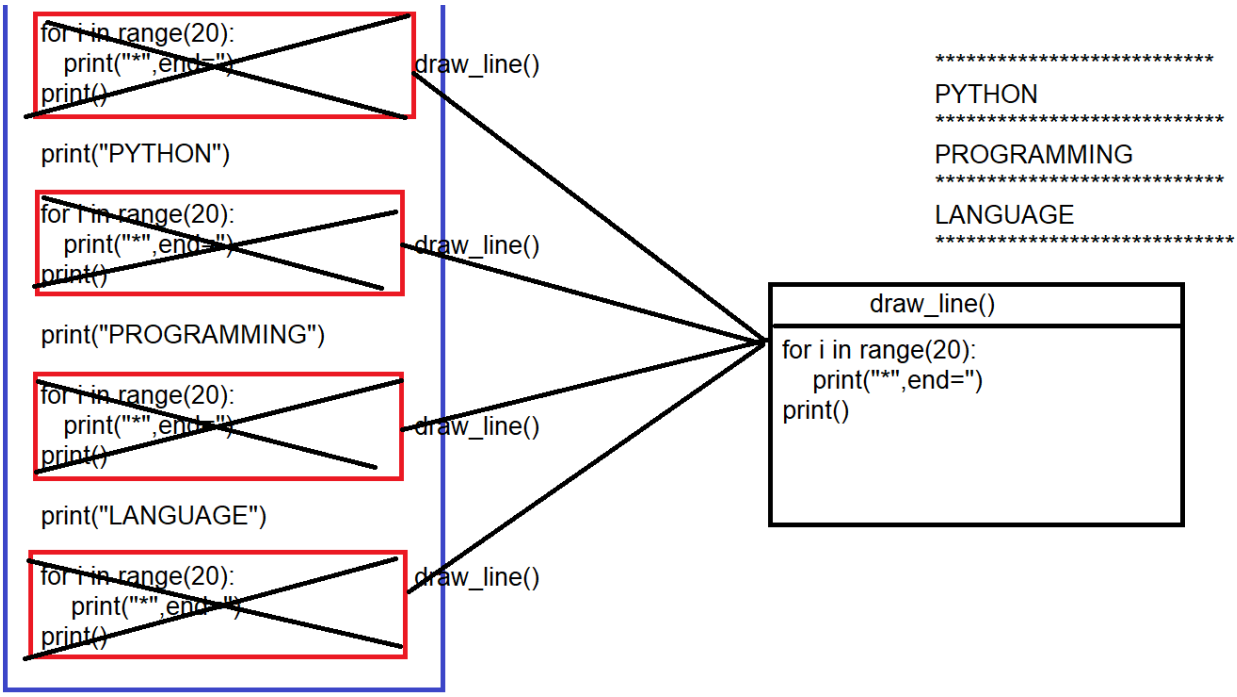
Functions

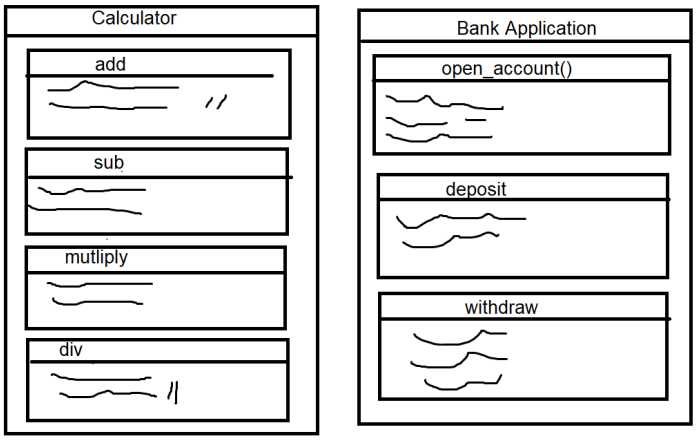
- What is Function?
- Advantages of functions
- Syntax and Writing function
- Calling or Invoking function
- Classification of Functions
  - No arguments and No return values
  - With arguments and No return values
  - With arguments and With return values
  - No arguments and With return values
  - Recursion
- Python argument type functions :
  - Default argument functions
  - Required(Positional) arguments function
  - Keyword arguments function
  - Variable arguments functions
- 'pass' keyword in functions
- Lambda functions/Anonymous functions
  - map()
  - filter()
  - reduce()
- Nested functions
- Non local variables, global variables
- Closures
- Decorators
- Generators
- Iterators
- Monkey patching

Python is a multi paradigm programming language. A programming paradigm defines set of rules and regulations for organizing of data and instructions.

1. Procedural Oriented Programming (POP)
2. Object Oriented Programming (OOP)

In procedural oriented programming, instructions are organized by dividing into small pieces, these small pieces called sub routines or functions.





What is function?

A function is small program within program.

A function is building block of procedural oriented programming.

A function is self contained **block** which contain set of instructions to perform some operation.

A function is named block.

These functions are two types

1. Predefined functions
2. User defined functions

Existing functions or functions provided by python library are called predefined functions.

Example: print(),input(),oct(),hex(),chr(),int(),float(),complex(),str(),…

Functions developed by programmer are called user defined functions (OR) application specific functions are called user defined functions.

Advantage of functions

1. **Reusability**: Functions allows writing code once and using many times.

2. **Modularity:** Modularity allows to divides code according their operations into small programs.
3. **Readability:** Easy to understand or maintain
4. **Efficiency:** functions increase efficiency of program by decreasing size of the program.

Function is divided into 2 parts.

1. Function definition
2. Function invoking/Calling/Execution

Function definition

Defining function is nothing but writing function.
A function is defined using "def" keyword.

Syntax:

```
def <function-name>([parameters]):
        ['''Doc String''']
        Statement-1
        Statement-2
        Statement-3
        .....
```

A function defined,

1. Function with parameters/arguments with return value
2. Function with parameters/arguments without return value
3. Function without parameters/arguments with return value
4. Function without parameters/arguments without return value

Function with input is defined with parameters/arguments
Function with output is defined with return value

Note: function name is an identifier, function name is defined in lowercase and if function name having multiple words, it is separated with _

Function is not executed automatically; it has to be called as executable statement.

Example:

```
# function without parameters/arguments
def say_hello():
    '''This is my first function in python'''
    print("Hello Python")

def fun1():
    print("fun1")

def fun2():
    print("fun2")

def fun3():
    print("fun3")


# Main
say_hello() # invoking function or calling function or executing function
say_hello()
say_hello()
print(say_hello.__doc__)
fun1()
fun2()
fun3()
```

```
fun3()
fun1()
fun2()
```

Output:

```
Hello Python
Hello Python
Hello Python
This is my first function in python
fun1
fun2
fun3
fun3
fun1
fun2
```

Example:

```
# function without parameters/arguments without return value
def draw_line():
    for i in range(20):
        print("*",end='')
    print()



draw_line() # UDF
print("PYTHON") # PDF
draw_line()
print("PROGRAMMING")
draw_line()
```

Output:

****

PYTHON

****

PROGRAMMING

****