Module in python

A module allows you to logically organize your Python code. It helps to keep related code into a single file called module, it makes the code easier to understand and use.

Creating modules in Python

Simply, a module is a file consisting of Python code. A module can define functions, classes and variables. A module can also include runnable code

An Example

Say we have created a file MyFile.py that has following code

```
print("Welcome to MyFile.py")
PI = 3.14

def fact(x):
    print("Inside fact")
    if x == 1:
        return 1
    r = 1;
    c = x;
    while c >= 1:
        r = r*c
        c = c -1
    return r
```

And we want to use variables and classes of MyFile.py into another file yourFile.py such that both exists in same directory. So we have to use import statement in yourFile,py as given below

```
import MyFile
print("p1 has value",MyFile.PI)
print(MyFile.fact(5))
```

The above code will produce following output

```
Welcome to MyFile.py
p1 has value 3.14
Inside fact
120
```

from....import in python

Python's from statement allow to import specific attributes from a module into the current namespace. The from...import has the following syntaxes −

```
      from modname import name1[, name2[, ... nameN]]

      from modname import name1 as alt-name

      import modname

      import modname as alt-name
```

For example, to import the function fibonacci from the module fib, use the following statement −

```
      from MyFile import fact
```

This statement does not import the entire module fib into the current namespace; it just introduces the item fact from the module MyFile into the global symbol table of the importing module.

We can also use `from modname import *` in python. This provides an easy way to import all the items from a module into the current namespace.

Note
`import modname` Imports the module, and creates a reference to that module in the current namespace. Then we need to define completed module path to access a particular attribute or method from inside the module

With `from X import *` statement we can simply use a plain name to refer to things defined in module X. But X itself is not defined, so X.name doesn't work.

When you import a module, the Python interpreter searches for the module in the following sequences

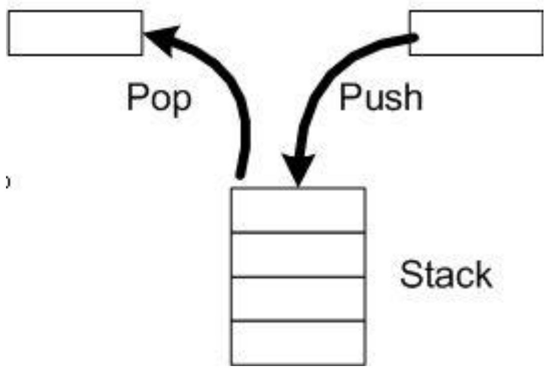
- The current directory.
- If the module isn't found, Python then searches each directory in the shell variable PYTHONPATH.
- If all else fails, Python checks the default path. On UNIX, this default path is normally C:\Users\KAPIL\AppData\Local\Programs\Python\Python36 or something like that.

Stack in Python

Stack is an abstract data type with a bounded (predefined) capacity. It is a simple data structure that allows adding and removing elements in a particular order. Every time an element is added, it goes on the top of the stack, the only element that can be removed is the element that was at the top of the stack, just like a pile of objects. A stack implements LIFO (Last in first out) system.

push() function is used to insert new elements into the Stack and **pop()** is used to delete an element from the stack. Both insertion and deletion are allowed at only one end of Stack called Top. Another operation is **peep()** operation in which value of top element is returned but does not removed from the stack

Stack is said to be in Overflow state when it is completely full and is said to be in Underflow state if it is completely empty.



Implementation of stack

```
"""
        function to check if stack is empty
"""
def isEmpty(stack):
        if stack==[]:
                return True
        else:
                return False

"""
        function to push element into the stack
"""
def Push(stack,item):
        #append the item at the end of list
        stack.append(item)
        #set top to index of newly insered element
        top=len(stack)-1

"""
   function to pop element into the stack
"""
```

```
def Pop(stack):
        #check for underflow
        if isEmpty(stack): # verifies whether the stack is empty or not
                print("Underflow")
        else:
                #use pop function to delete last element from list
                item = stack.pop()
                if len(stack)==0:
                        top = None
                else:
                        top = len(stack) - 1
                        print("Popped item is "+str(item))

"""
   function to display elements of stack
"""
def display(stack):
        if isEmpty(stack):
                print("Stack is empty")
        else:
                print("Elements in the stack are: ")
                top = len(stack) - 1
                for i in range(top,-1,-1):
                        print (str(stack[i]))

# executable code
if __name__ == "__main__":
        #create a list to be used as an stack
        stack=[]
        Push(stack,1)
        Push(stack,2)
        Push(stack,3)
        Push(stack,4)
        Pop(stack)
        display(stack)
```