UNIT- II: The Turtle Module

- The turtle module is an extended reimplementation of the same-named module from the Python standard distribution up to version Python 2.5.
- The turtle module provides turtle graphics primitives, in both object-oriented and procedure-oriented ways.

Simple Repetition

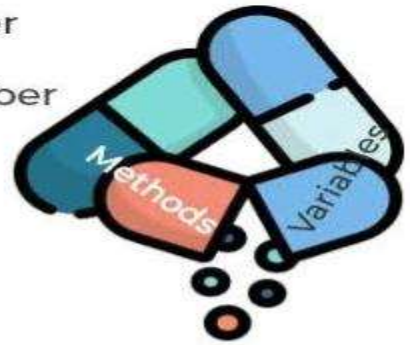
- A repetition structure causes a statement or set of statements to execute repeatedly.
- Repetition statements are called loops, and are used to repeat the same code multiple times in succession.
- Python has two types of loops: Condition-Controlled and Count-Controlled
- Condition-Controlled loop uses a true/false condition to control the number of times that it repeats – while.
- Count-Controlled loop repeats a specific number of times – for.

Encapsulation

- The process of wrapping up variables and methods into a single entity is known as Encapsulation.
- Private veriable:Prevent accidental change, an object's variable can only be changed by an object's method.
- Protected members (in C++ and JAVA) are those members of the class that cannot be accessed outside the class but can be accessed from within the class and its subclasses.

__a -> Private Member

_a -> Protected Member



Generalization

- Generalization is the process of extracting shared characteristics from two or more classes, and combining them into a generalized superclass. Shared characteristics can be attributes, associations, or methods.

Interface design

- In object-oriented languages like Python, the interface is a collection of method signatures that should be provided by the implementing class.
- Implementing an interface is a way of writing an organized code and achieve abstraction.

Docstring

- The Python docstrings provide a suitable way of associating documentation with:
- Python modules
- Python functions
- Python classes

Conditions and Recursion

Floor division and modulus

- The modulus-function computes the remainder of a division, which is the “leftover” of an integral division.
- The floor-function provides the lower-bound of an integral division. The upper-bound is computed by the ceil function.

```
A=5

b=2

print(a%b) # Prints 1 as leftover

print(a//b) # Prints 2, since 5/2=2.5 and the decimal is cut off

print(a – (a//b)*b)
```

Boolean expression and logical operations

- A boolean expression (or logical expression) evaluates to one of two states true or false. Python provides the boolean type that can be either set to False or True. Many functions and operations returns boolean objects.
- The not keyword can also be used to inverse a boolean type.
- >>> not True
- False

Conditional execution and alternative execution

Conditional execution

In order to write useful programs, we almost always need the ability to check conditions and change the behavior of the program accordingly. Conditional statements give us this ability. The simplest form is the if statement:

```
if x > 0 :
    print('x is positive')
```

Alternative execution

A second form of the if statement is alternative execution, in which there are two possibilities and the condition determines which one gets executed. The syntax looks like this:

```
if x%2 == 0 :
    print('x is even')
else :
    print('x is odd')
```

If the remainder when x is divided by 2 is 0, then we know that x is even, and the program displays a message to that effect. If the condition is false, the second set of statements is executed.

Chained condition and nested condition

- **Chained conditionals**
- Sometimes there are more than two possibilities and we need more than two branches. One way to express a computation like that is a chained conditional

```
if x < y:
    print('x is less than y')
elif x > y:
    print('x is greater than y')
else:
    print('x and y are equal')
```

elif is an abbreviation of "else if." Again, exactly one branch will be executed.

- **Nested conditionals**
- One conditional can also be nested within another.

```
If x == y:
    print('x and y are equal')
else:
    if x < y:
        print('x is less than y')
    else:
        print('x is greater than y')
```

- The outer conditional contains two branches. The first branch contains a simple statement. The second branch contains another if statement, which has two branches of its own.

Recursion and Infinite recursion

- The process in which a function calls itself directly or indirectly is called Recursion and the corresponding function is called a Recursive function.
- Types of Recursions:
- Recursion can be further classified into two kinds, depending on when they terminate:
- Finite Recursion
- Infinite Recursion
- **Finite Recursiome**
- Finite Recursion occurs when the recursion terminates after a finite number of recursive calls. A recursion terminates only when a base condition is met.
- **Infinite Recursion:**
- Infinite Recursion occurs when the recursion does not terminate after a finite number of recursive calls. As the base condition is never met, the recursion carries on infinitely.

FruitFul functions

- **Return value**

The function which return any value are called as fruitful function.

The function which does not return any value are called as void function.

- Void function without return statement for
- def fun1():
- print(“Python”)
- Void function with return statement
- def fun2():
- print(“ITVoyagers”)
- return

- **Incremental development**

- Steps for Development¶
- 1) Write pseudocode line by line on how logic works
- 2) Write code line by line in Python
- 3) Run program and validate code works with dummy data
- 4) Repeat steps 2-3 until your function works as anticipated

Composition and Boolean functions

- **Composition**
- Function composition is the way of combining two or more functions in such a way that the output of one function becomes the input of the second function and so on.
- Def add(x):
- return x + 2
- def multiply(x)
- return x * 2
- print("Adding 2 to 5 and multiplying the result with 2: ",multiply(add(5)))
- **Boolean functions**
- The bool() function converts the given value to a boolean value (True or False). If the given value is False, the bool function returns False else it returns True.
- Syntax of bool() function
- bool([value])

More Recursion

- The term Recursion can be defined as the process of defining something in terms of itself. In simple words, it is a process in which a function calls itself directly or indirectly.
- A complicated function can be split down into smaller sub-problems utilizing recursion.

Leap of faith

Following the flow of execution is one way to read programs, but it can quickly become labyrinthine. An alternative is what I call the “leap of faith.” When you come to a function call, instead of following the flow of execution, you assume that the function works correctly and returns the right result.

Checking types

- If a single argument (object) is passed to type() built-in, it returns type of the given object. If three arguments (name, bases and dict) are passed, it returns a new type object.

```
x = 5
s = “geeksforgeeks”
y = [1,2,3]
print(type(x))
print(type(s))
print(type(y))
```