Structured Programming Approach

→ Program is made as a single structure

→ No jumping statement

Eg: GOTO

→ Consists
- Selection
- Sequence
- Iteration.

→ Well structured and seperated modules

→ Entry & exit is a single time event.

✓	✗
- Easy to read & understand - User friendly - Easy to maintain - Problem based - Requires less effort & time - Easy to debug - Mostly machine independent	- Long time is required for conversion to machine code. - Real world scenarios are not sequential. - Development takes longer time.

Programming Language Theory

PLT is unofficially denoted as the lowercase greek letter λ (lambda). PLT is a branch of computer science that deals with design, implementation, analysis, characterization and classification of formal languages known as prog. lang

and their individual features. This is an active research area. PLT makes use of other branches like

- mathematics
  - computability theory
  - category theory
  - Set theory
- software engineering
- linguistics
- cognitive science

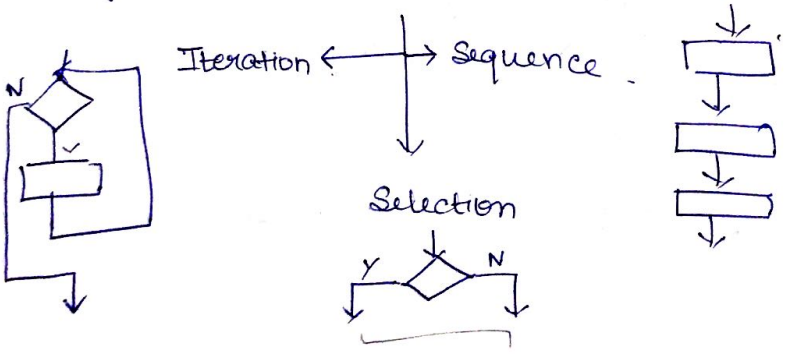
Sub disciplines are

→ Formal semantics
→ Type theory
→ Program analysis & transformation
→ comparitive programming language analysis
→ Generic and meta programming
→ Domain specific languages
→ Compiler construction
→ Run-time systems.

Bohm - Jacopini theorem

- Also called structured program theorem
- Theorem:

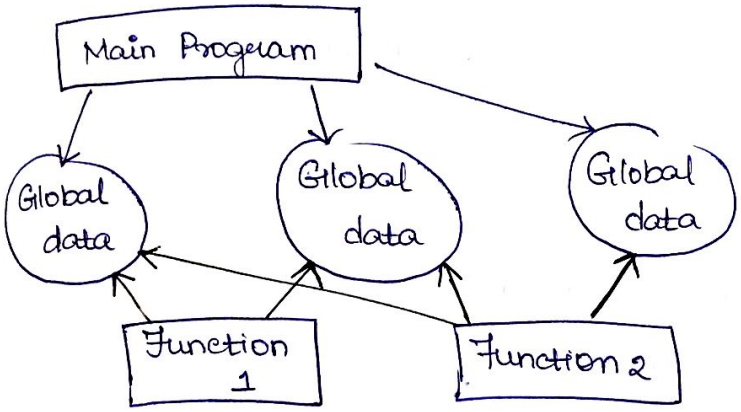
A class of control-flow graphs (or flow charts) can compute any computable function if it combines the subprograms in only 3 specific ways. (control structures)



Procedural Programming Approach.

→ derived from imperative programming
→ concept of procedure call
→ Procedure is a series of computational step.
→ Stack register supports procedural programming
→ The program is divided into multiple sub-programs or modules / functions.

→ Global variables are used widely and accessing it inside the function is possible.
→ Change in global variables affects function's computation.



Demerits

→ Data privacy
→ Difficult to maintain
→ Focuses on functions & not on data
→ Cannot apply to real-world problem

Characteristics

→ large program is divided into managable procedures
→ Different function can access global data.
→ Functions can alter / change global data.
→ Top Down approach

→ Reusability
→ Efficient use of memory
→ Track of flow is easy

Object Oriented Programming Paradigm

→ Based on objects which consist of data and methods

→ Uses modularity and reusability

→ Bottom - Up Approach.

→ Data with methods operate upon object's data.

→ Interaction through functions

→ hierachy of classes is united by inheritance relationship.

Class: a specification, blueprint.

Object: physical presence of class in memory

Instance: a unique copy of the object

Encapsulation:

→ idea of wrapping data and methods

→ private, protected, public
  - protected: can be assessed outside class customary: not to access.

Abstraction:

→ hide the functionality
→ What is done ✓
  How it is done X

abstraction is acheived by abstract classes and interfaces

Syntax

```
from abc import ABC  # Abstract Base Class
class ClassName (ABC):
```

Eg:

```
from abc import ABC
class Car(ABC):
    def cost(self):
        pass

class Tesla(Car):
    def cost(self):
        print("$5000")
```

3 levels
→ Internal level
→ conceptual
→ external

Polymorphism

→ occurence in different forms
→ represents different types in different scenarios.

Eg: Inbuilt

a+b	"H" + "I" = "HI"

len() → length of string
→ No. of items in list
→ No. of keys in dict.

⊙ Class

```
class Cat:
    def sound(self):
        print("Meow")

class Dog:
    def sound(self):
        print("Bark")

for animal in
c = Cat()
D = Dog()
for animal in (C,D):
    animal.sound()
```

O/P Meow Bark

⊙ Method overriding

```
Class Shape:
    def p-shape(self):
        pass
Class Square:
    def p-shape(self):
        print("Square")

S = square()
S.p-shape()
```

Inheritance

→ allows us to inherit all methods and properties

→ Parent & child

- `__init__()`
- `super()` → makes the child class inherit all methods & properties
- `__del__()`

Access Specifiers

public	default	name
private	one underscore	_name
protected	two underscore	__name