

Monolithic

VS

Modular

Monolithic

VS

Modular

EXPLICIT

EASY CODING

FAST CODING

IMPLICIT

MAINTENABLE

TESTABLE

REUSABLE

EASY TO READ

# Programming Paradigms

```
graph TD; A[Programming Paradigms] --> B[Imperative]; A --> C[Declarative]; B --> D[Procedural]; B --> E[Object-Oriented]; C --> F[Functional]; C --> G[Logic]; C --> H[Reactive];
```

Imperative

Declarative

Procedural

Object-Oriented

Functional

Logic

Reactive

# Programming Paradigms

```
graph TD; A[Programming Paradigms] --> B[Imperative]; A --> C[Declarative]; B --> D[Procedural]; B --> E[Object-Oriented]; C --> F[Functional]; C --> G[Logic]; C --> H[Reactive];
```

**Imperative**

**Declarative**

Procedural

Object-Oriented

Functional

Logic

Reactive

# Programming Paradigms

```
graph TD; A[Programming Paradigms] --> B[Imperative]; A --> C[Declarative]; B --> D[Procedural]; B --> E[Object-Oriented]; C --> F[Functional]; C --> G[Logic]; C --> H[Reactive]; D --- I["C<br/>Fortran<br/>Cobol<br/>Pascal"]; E --- J["Matlab<br/>Python<br/>C++<br/>Java"];
```

Imperative

Declarative

Procedural

Object-Oriented

Functional

Logic

Reactive

*Matlab*

*Python*

*C++*

*Java*

*C*

*Fortran*

*Cobol*

*Pascal*

*Haskell*

*Prolog*

*Erlang*

# Programming Paradigms

```
graph TD; A[Programming Paradigms] --> B[Imperative]; A --> C[Declarative]; B --> D[Procedural]; B --> E[Object-Oriented]; C --> F[Functional]; C --> G[Logic]; C --> H[Reactive]; D --> D1[Matlab]; D --> D2[C]; D --> D3[Fortran]; D --> D4[Cobol]; D --> D5[Pascal]; E --> E1[Python]; E --> E2[C++]; E --> E3[Java]; F --> F1[Matlab]; F --> F2[Python]; F --> F3["SQL/JavaScript, R"]; G --> G1[Prolog];
```

## Imperative

## Declarative

Procedural

Object-Oriented

Functional

Logic

Reactive

*Matlab*

*Python*

*C++*

*Java*

*Matlab*

*Python*

*SQL/JavaScript, R*

*C*

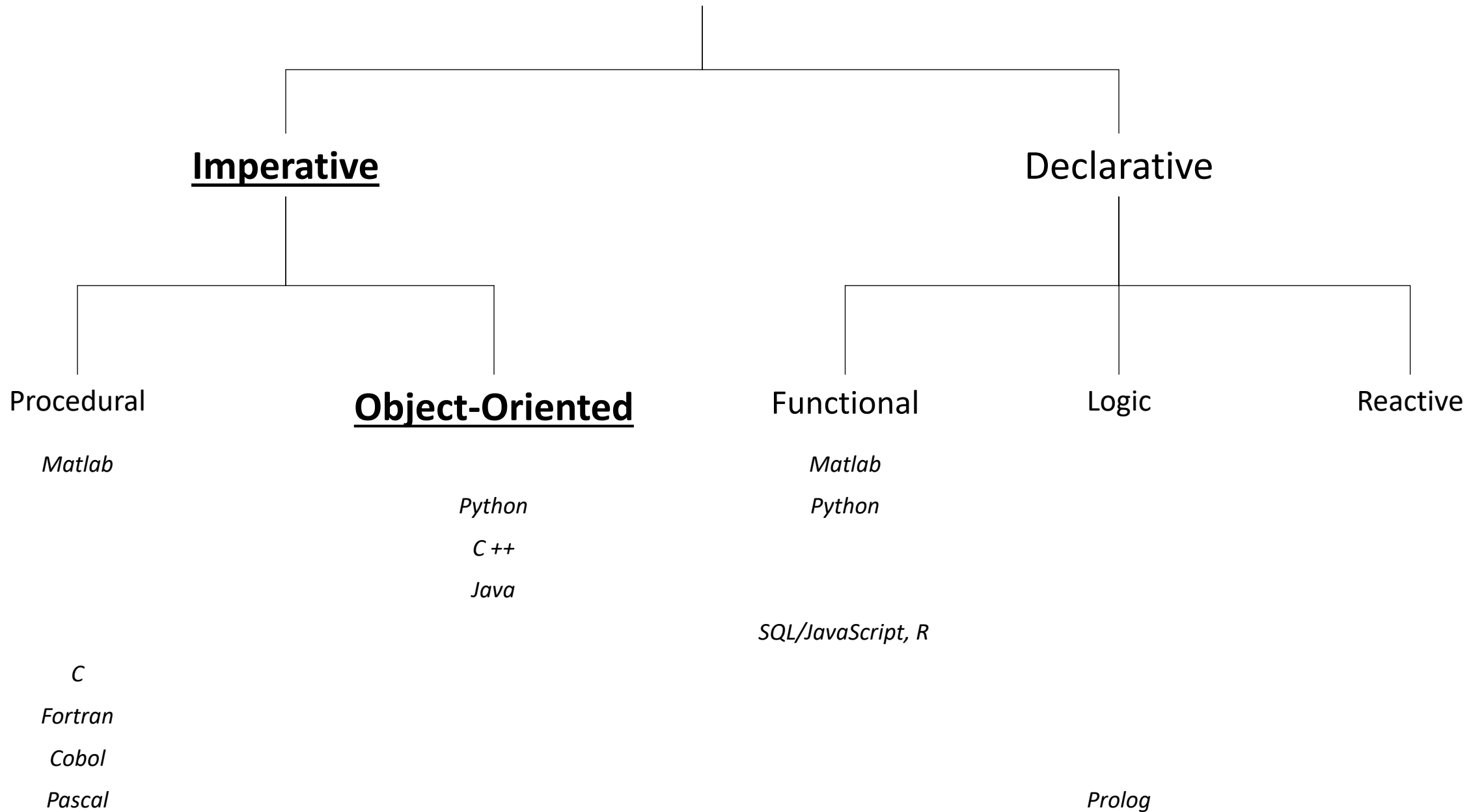
*Fortran*

*Cobol*

*Pascal*

*Prolog*

# Programming Paradigms



# **Object Oriented Programming (OOP)**

Abstraction

Inheritance

Encapsulation

Polymorphism



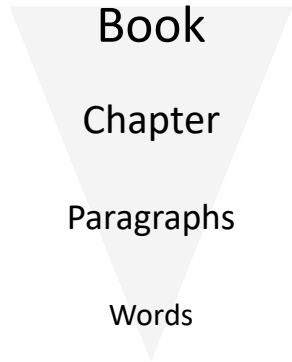
# Object Oriented Programming (OOP)

Abstraction

Inheritance

Encapsulation

Polymorphism



# Object Oriented Programming (OOP)

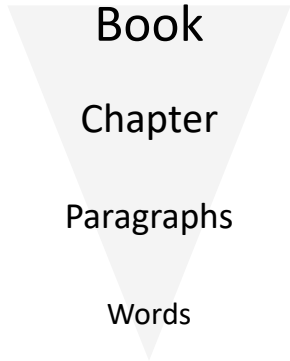
Abstraction

Inheritance

Encapsulation

Polymorphism

Code reuse



# Object Oriented Programming (OOP)

Abstraction

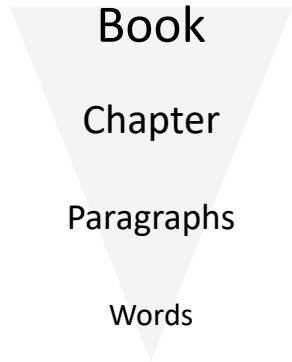
Inheritance

Encapsulation

Polymorphism

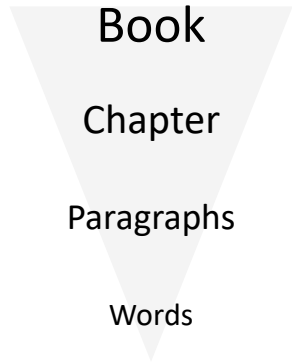
Code reuse

Scope changing  
state



# Object Oriented Programming (OOP)

Abstraction



Inheritance

Code reuse

Encapsulation

Scope changing  
state

Polymorphism

> 1 type