Programming for engineers II

Master in Sustainable Production Creation

Gustavo Martín Larrea Gallegos
[gustavo.larrea@list.lu]

University of Luxembourg

Master in Sustainable Product Creation

March 25th , 2025

Course plan

> Introduction to computation and programming languages (week 1)

> Elements of python programming (statements, operators, loops, variables, etc.) (week 2 and 3)

> Concepts of object-oriented programming (classes, objects, methods, etc.) (week 4 and 5)

> Basics of software design (week 6)

> Data structures and data visualization (week 7 and 8)

> Introduction to machine learning and AI tools (week 9 and 10)

> Systems design, APIs and applications (week 11)

So far...

> Using procedural programming, we learned the basic concepts, syntax and elements of the python programming language:

- Variables (assignments)
- Strings, Numbers, Bool
- Lists, Dicts, Tuples
- Loops
- Functions
- Conditional statements

So far...

> Using procedural programming, we learned the basic concepts, syntax and elements of the python programming language:

- Variables (assignments)
- Strings, Numbers, Bool
- Lists, Dicts, Tuples
- Loops
- Functions
- Conditional statements

Missing:

- Walrus operator
- Locals and globals
  - locals()
  - globals()

So far...

> Using procedural programming, we learned the basic concepts, syntax and elements of the python programming language:

- Variables (assignments)
- Strings, Numbers, Bool
- Lists, Dicts, Tuples
- Loops
- Functions
- Conditional statements

Missing:

- Looping over multiple lists zip()
- While loops

So far...

> Using procedural programming, we learned the basic concepts, syntax and elements of the python programming language:

- Variables (assignments)
- Strings, Numbers, Bool
- Lists, Dicts, Tuples
- Loops
- Functions
- Conditional statements

Missing:

- Decorators
- Generators
- Annotations
- coroutines

So far...

> The procedural paradigm is enough in some cases, but it is not easy to represent state. For this, we learned OOP, where abstractions are objects and data and code are embedded in them.

So far...

> The procedural paradigm is enough in some cases, but it is not easy to represent state. For this, we learned OOP, where abstractions are objects and data and code are embedded in them.

- Constructors
- Methods
- Attributes
- Polymorphism, Abstraction
- Inheritance

So far...

> The procedural paradigm is enough in some cases, but it is not easy to represent state. For this, we learned OOP, where abstractions are objects and data and code are embedded in them.

- Constructors
- Methods
- Attributes
- Polymorphism, Abstraction
- Inheritance

Missing:

- Decorators
- Generators
- Annotations
- coroutines

So far...

> The procedural paradigm is enough in some cases, but it is not easy to represent state. For this, we learned OOP, where abstractions are objects and data and code are embedded in them.

- Constructors
- Methods
- Attributes
- Polymorphism, Abstraction
- Inheritance

Missing:

- Behaviors (dunder methods)
- Static and class methods

So far...

> The procedural paradigm is enough in some cases, but it is not easy to represent state. For this, we learned OOP, where abstractions are objects and data and code are embedded in them.

- Constructors
- Methods
- Attributes
- Polymorphism, Abstraction
- Inheritance

Missing:

- Properties

So far...

> The procedural paradigm is enough in some cases, but it is not easy to represent state. For this, we learned OOP, where abstractions are objects and data and code are embedded in them.

- Constructors
- Methods
- Attributes
- Polymorphism, Abstraction
- Inheritance

Missing:

- Protocols

So far...

> The procedural paradigm is enough in some cases, but it is not easy to represent state. For this, we learned OOP, where abstractions are objects and data and code are embedded in them.

- Constructors
- Methods
- Attributes
- Polymorphism, Abstraction
- Inheritance

Missing:

- Multiple inheritance
- Class resolution

What is coming...

> The basics of python covered (sort of), we can now focus on building functional systems and software.

- Imports, modules and packages
- I/O
- Error handling
- Patterns and refactoring
- Testing
- Http requests
- Async / multithreading
- Python ecosystem
- Git and version control
- Code formatting/linting
- Building documentation
- CI/CD

[example]