Nonlinear Dynamics and Chaos

PHYMSCFUN12

Wladimir E. Banda Barragán

wbanda@yachaytech.edu.ec

MSc in Fundamental Physics

Yachay Tech University - 2025

Crash course on Python

Python is a high-level, interpreted programming language.

It uses indentation to define code blocks.

In Python, **everything is an object.** Objects are at the code of the language and are self-contained containers.

Object-Oriented Programming (OOP) is a method for designing software that organises the code into containers that group both data (attributes) and functions (methods) that operate on that data into a single unit.

The data and the functions that manipulate that data are tightly bundled together within an object.

Objects in Python

Mutable Objects: objects whose state can be changed after they are created. We can modify, add, or remove elements from these objects without creating a new object in memory).

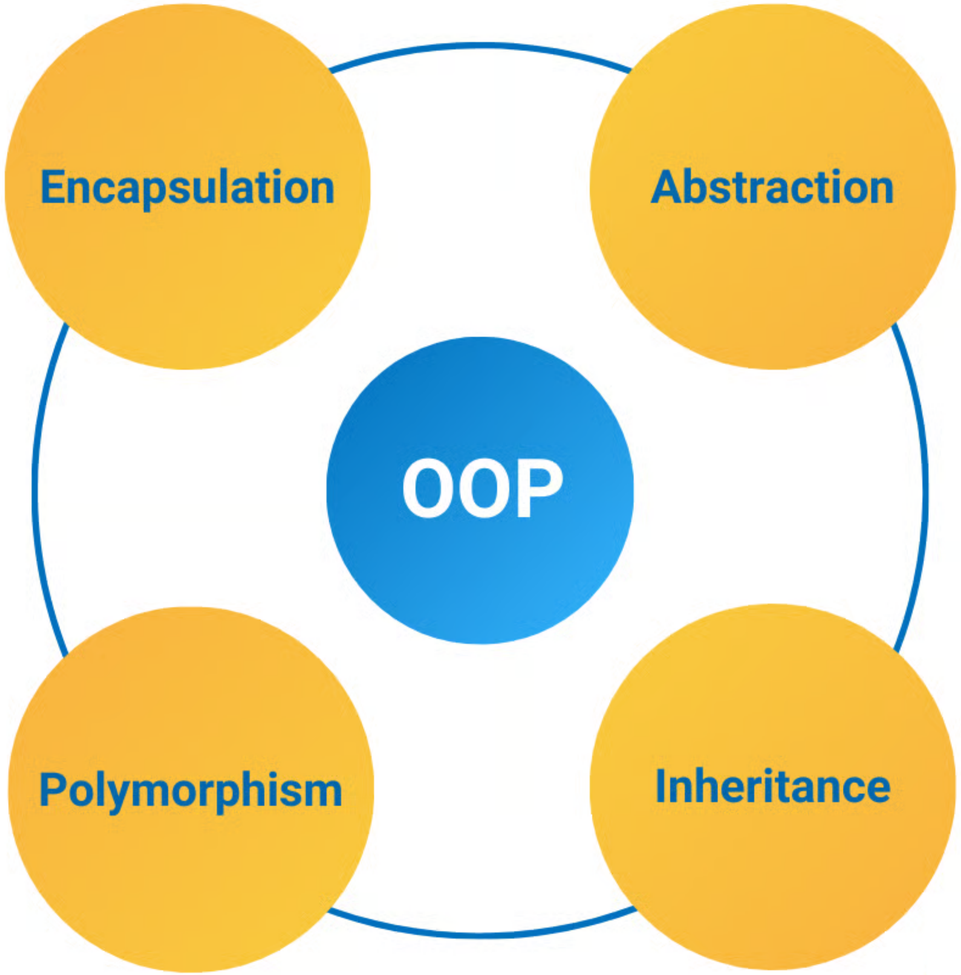
Examples: list, dict, set

Immutable Objects: objects whose state cannot be changed after they are created. Any operation that appears to modify an immutable object actually creates a new object in memory.

Examples: int, float, str, tuple, function, class

OOP helps to create modular, reusable, and more maintainable code. It makes it easier to manage large, complex programs by breaking them down into smaller, self-contained components.

Object-oriented Programming (OOP)



Encapsulation: Merging data **(attributes)** and the functions **(methods)** that operate on that data into a single unit (an object). This hides the internal state of an object from the outside world.

Abstraction: Hiding complex implementation details and showing only the essential features of an object. This simplifies the user's interaction with the object.

Polymorphism: The ability of objects of different classes to respond to the same method call in their own unique way. The word "polymorphism" means "many forms."

Inheritance: A mechanism where a new class (**derived class**) can inherit properties and behaviours from an existing class (**main class**). This promotes code reuse.

Tutorial Time:

You should go to the course GitHub repository and click on **Python Crash Course**.

https://github.com/MSc-Fundamental-Physics/nonlinear-dynamics-chaos

Functions in Python

def/return functions: Customised functions.

```
# Header
def thermal_pressure(nden, temp):
  # Body
  # Docstring:
  """
  Function computes the thermal pressure of ideal gases.
  Inputs: nden (number density), temp (temperature)
  Output: prs (pressure)
  Author: W.E.B.B.
  Date created: 28/04/23
  Date modified: 26/02/2024
  """

  # What you compute
  prs = nden*k_b*temp

  # What you return
  return prs
```

Header & Arguments

Docstring

Accesed via help()

Return Statement

Functions in Python

Lambda Functions: These are used to quickly define and use functions.

```
# Here we can use a lambda function
z_1D = lambda x: x**3
```

1-line Statement

Nested & Iterative Functions

```
def func1():
  “”” Local Variable Access “””
  msg = “Local Variable”
  def func2():
    print(msg)

  func2()

func1()
```

```
def factorial(n):
  """
  Calculate and return the factorial of n.
  """
  if n == 1 or n==0:
    # First case
    f_1 = 1
    return f_1
  else:
    # Other cases, we do a recursive call
    f_2 = n*factorial(n-1)
    return f_2
```

Classes in Python

Classes are fundamental concepts for object oriented programming with python.

A class defines a data type with both data and functions that can operate on the data.

An object is an instance of a class. Each object will have its own namespace (separate from other instances of the class and other functions, etc. in your program).

We use the dot operator, `.`, to access members of the class (data or functions). We've already been doing this a lot, strings, ints, lists, ... are all objects in python.

Documentation on Python Classes:

https://docs.python.org/3/tutorial/classes.html

Classes in Python

Naming conventions

The python community has some naming convections, defined in PEP-8:

https://www.python.org/dev/peps/pep-0008/

The widely adopted ones are:

Class names start with an uppercase, and use "camelcase" for multiword names, e.g. `ClassicalMechanics`

Variable names (including objects which are instances of a class) are lowercase and use underscores to separate words, e.g., `simulation_mechanics`

`M`odule names should be lowercase with underscores

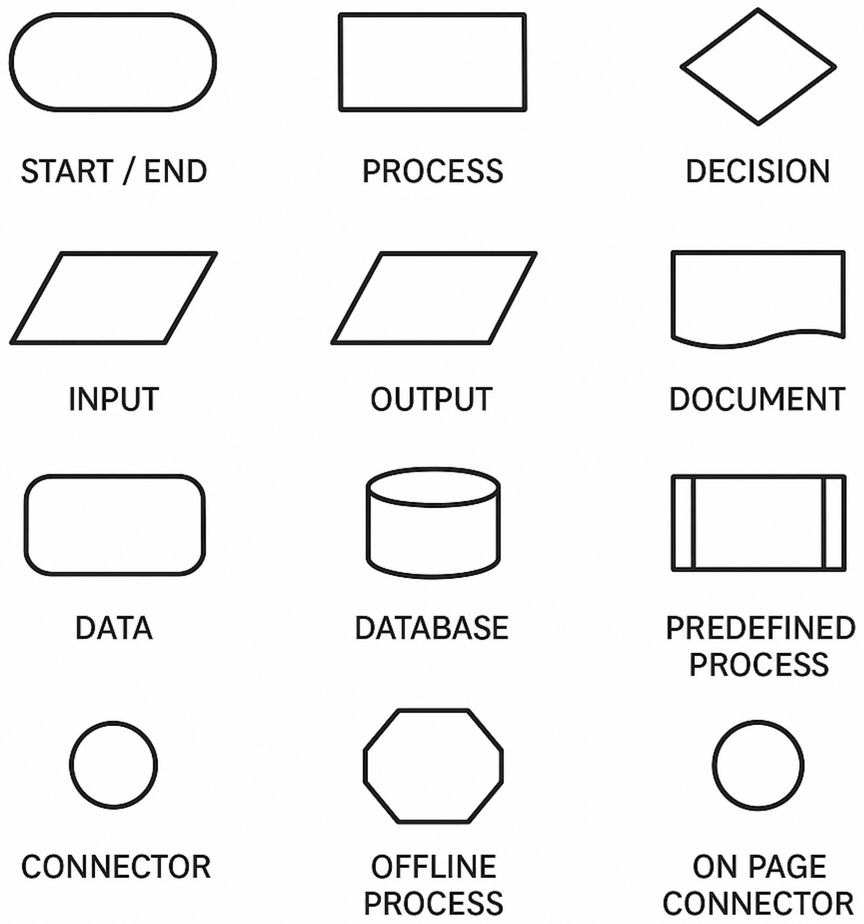
OOP Algorithms and Code WorkFlows

Code workflows are essential for software design as they:

- Separate logical planning from complex programming syntax.
- Create a universal visual language for code developing teams.
- Make complex processes and dependencies easier to understand.
- Help identify logical errors and optimise flow before coding.
- Provide intuitive, lasting documentation for any system.
- Demonstrate a true understanding of a problem's structure and solution.

OOP Algorithms and Code WorkFlows

Usual symbols include:



ANSI Standard:
https://nvlpubs.nist.gov/nistpubs/Legacy/FIPS/fipspub24.pdf

WorkFlow Editors

There are a few (free) workflow editors:

1. Raptor

RAPTOR is a flowchart-based programming environment, designed specifically to help students visualise their algorithms and avoid syntactic baggage.

It is free and allows to visualise logic and create executable flowcharts.

Link: https://raptor.martincarlisle.com

2. yEd Graph Editor

yED is used for creating large, complex, and professional-looking diagrams.

yED works offline.

Link: https://www.yworks.com/products/yed

WorkFlow Editors

There are a few (free) workflow editors:

3. Diagrams.net (draw.io)

Diagrams.net is a web-based tool to create workflow diagrams with flowchart shapes.

It is free, open-source and integrates with Google Drive, OneDrive, and GitHub.

Link: https://app.diagrams.net/

4. Code2Flow

Also web-based with a free tier (limited number of charts per day.)

You can generate flowcharts from existing code.

Link: https://code2flow.com/

Tutorial Time:

You should go to the course GitHub repository and click on **Python Classes notebook.**

https://github.com/MSc-Fundamental-Physics/nonlinear-dynamics-chaos