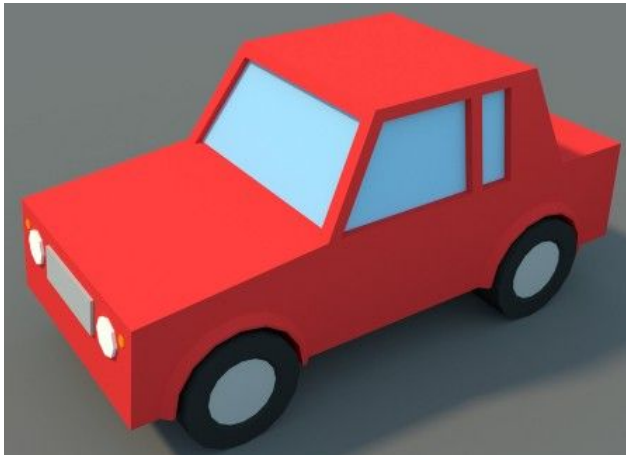
Python Programming

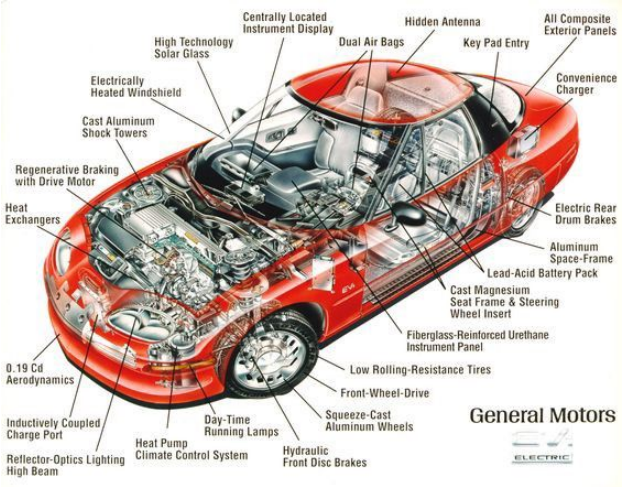
Abstraction

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

What vs How





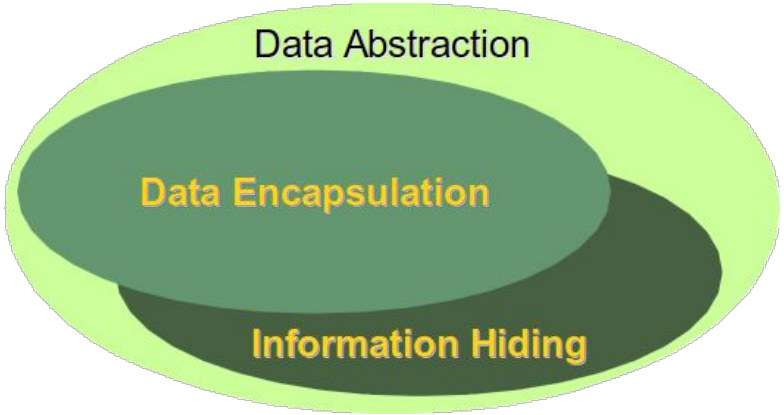
Img Src

What vs How

- Do you care how:
  - a TV/Car work? Google really search and find results? Browser access internet?
  - Python computes pow(2.0, -3.2)
  - Python handles OS to read/write from files using fstream?
- Most of time, the user cares with WHAT not HOW
  - What = Function takes and return
  - How = it is implemented. But
    - Some implementation can be slow (loop to sum 1 to n) or fast (sum = n * n+1 / 2)
    - Some might be buggy or stable (internet explorer vs Firefox)
    - Some might takes more memory (chrome vs Firefox)
  - We can **change** internal implementation of class **independently** without affecting the user.
    - User depends on **limited visible** functionalities of specific WHAT details

Abstraction Concept

- Abstraction is about **hiding** unwanted details while **showing** most essential in a given context
  - *The statement is easily explained & in administered C++/Java (public / private / separation)*
- For now think:
  - Abstract = Focus on High level (what not how)
    - Implementation is hidden
- Off-topic
  - Useful about abstraction in CS
  - Smart guys have high Abstract thinking skills
    - Algorithms, Problem solving, Management



Img Src

Abstraction Concept

- Does shape class cares **How** area method is implemented?
  - No. Hide/Abstract this details. It cares about **what**

```
class Shape:
    def __init__(self, name):
        super().__init__()
        self.name = name

    @property
    def area(self):
        raise NotImplementedError
```

What is wrong?

```python
class Shape:
    def __init__(self, name):
        super().__init__()
        self.name = name

    @property
    def area(self):
        raise NotImplementedError
```

```python
class Shape:
    def __init__(self, name):
        super().__init__()
        self.name = name

    def print(self):
        print(self.name, self.area)
```

- Shape class can’t provide implementation to the area method/property
- What if a user created object from shape?
  - In first solution, we handled that by raising exception. None in second code
- Shape is **incomplete** class. How to prevent object creation?
  - There must be a child class that implements missing methods (e.g. Rectangle)

Abstract Classes (ABC)

- Describes the behavior of an **incomplete** class
  - Future derived classes add their particular implementations
  - An abstract class should have at least one abstract method (e.g. shape area())
- Python allows us to **mark** a class or method as **abstract**
  - You **can’t create** an object from an abstract class that has an abstract method
- In next session, we see how to code that!

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."