VACATION CLASS

ALGORITHM AND DATA STRUCTURE

Week 3

18 September 2025

YEM DARO

OBJECT ORIENTED PROGRAMMING (OOP)

- Understanding **OOP** is **SUPER IMPORTANT !**
- Making learning languages easier
  - Java, Kotlin
  - Javascript
  - php
  - Dart
  - Python
  - C++
  - AND MORE

OOP

គិតពីតួអង្គ

- **Object Oriented:**
  - What are the Objects in my system?
    - Human, Car, Gun, Dog
    - Number, Word
    - Application, Web Page, Database
  - What should this Object have?
    - name, money, friends
    - API Source
  - What can this Object do?
    - Speak, Eat, Sleep
    - Get Movies from the internet

OOP

WHY OOP?

- Abstraction
  - Hide complicated detail
- Composition
  - Use Objects in your Object
- Extendable
  - Build on top of past works
- Don’t Repeat Yourself (DRY)
  - Write less code
  - Easy to find bug and kill the bug

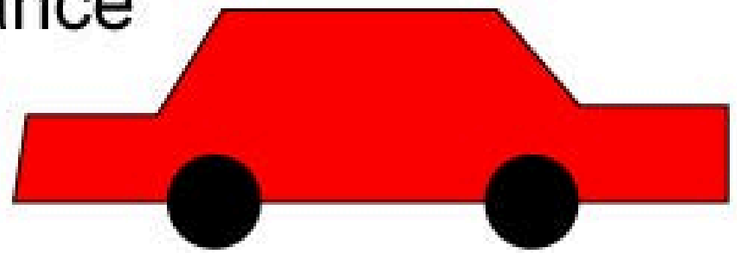
OOP

Class

Car

Create an instance

Object



Properties	**Methods** - behaviors
color	start()
price	backward()
km	forward()
model	stop()

Property values	**Methods**
color: red	start()
price: 23,000	backward()
km: 1,200	forward()
model: Audi	stop()

CLASS

The blueprint for your Object

Python

```
class Student():
    def __init__(self, first_name, last_name):
```

FIELD / PROPERTY

```
class Student():
    def __init__(self, first_name, last_name):
        self.first_name = first_name
        self.last_name = last_name
```

METHOD / INTERFACE

```python
class Student():
    def __init__(self, first_name, last_name):
        self.first_name = first_name
        self.last_name = last_name

    def display_info(self):
        print(f"Hi! I am {self.first_name} {self.last_name}")
```

Method

PYTHON FACT :D

```
class Student():
    def __init__(self, first_name, last_name):
```

Dunder Method:

(Double Underscore)

built-in methods in Python

CONSTRUCTOR

Method to create an Object from a Class

```
class Student():
    def __init__(self, first_name, last_name):
        self.first_name = first_name
        self.last_name = last_name
```

Constructor

CONSTRUCTOR

JAVA

```
public Class Student() {
    private String firstName;
    private String lastName;

    public Student(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }
}
```

Constructor

CONSTRUCTOR

Usage:

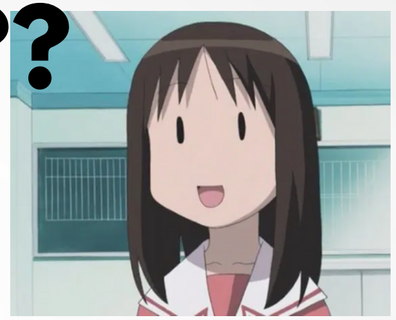
```
class Student():
    def __init__(self, first_name, last_name):
        self.first_name = first_name
        self.last_name = last_name
```

Create / Instantiate a Student Object:

```
s = Student("John", "Cena")
```

CONSTRUCTOR

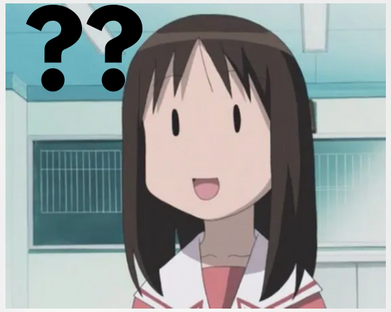
??



What is “self”?

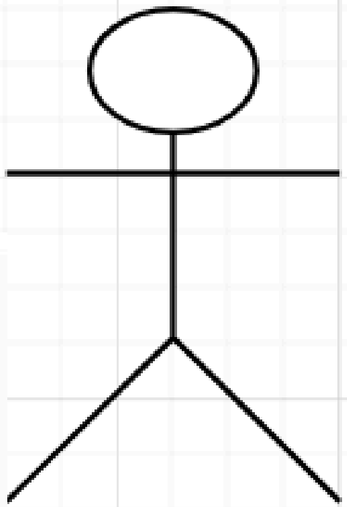
```
class Student():
    def __init__(self, first_name, last_name):
        self.first_name = first_name
        self.last_name = last_name
```

CONSTRUCTOR



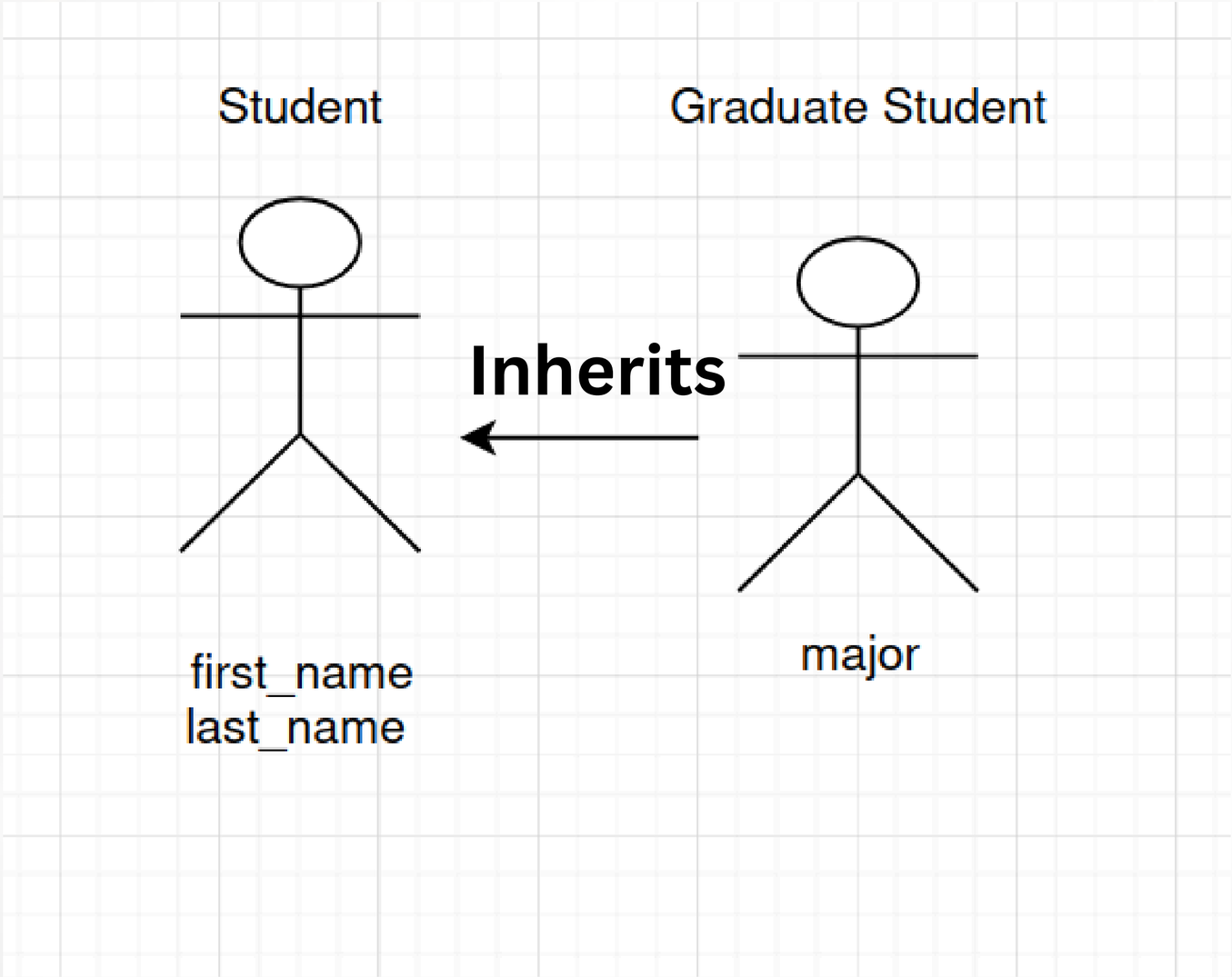
“self”: refers to the object itself

```
s = Student("John", "Cena")
```



Student
self.first_name = "John"
self.last_name = "Cena"

INHERITANCE



INHERITANCE

```
class Student():
    def __init__(self, first_name, last_name):
        self.first_name = first_name
        self.last_name = last_name

class GraduateStudent(Student):
    def __init__(self, first_name, last_name, major):
        super().__init__(first_name, last_name)
        self.major = 'CS'
```

Graduate Student inherits from Student

POLYMORPHISM

Function can accept different types

- Different data types
- Objects of different classes

```
def display_info(student):
    student.display_info()

display_info(s)
display_info(gs)
```

Function that accepts different types

GIT TIPS :DD

SHORTCUTS

- git config --global alias.**co** “checkout”
- git config --global alias.**br** “branch”
- git config --global alias.**lo** “log --oneline”
- git config --global alias.**cm** “commit -m”
- git config --global alias.**st** “status”

LEARN MORE

OBJECT ORIENTED PROGRAMMING

https://ocw.mit.edu/courses/6-0001-intro

PYTHON CLASSES AND INHERITANCE

https://ocw.mit.edu/courses/6-0001-intro