CS116 - LẬP TRÌNH PYTHON CHO MÁY HỌC

BÀI MỞ ĐẦU

TS. Nguyễn Vinh Tiệp

NỘI DUNG

1. **Giới thiệu môn học**
2. Lịch sử và thành tựu của máy học
3. Ôn tập ngôn ngữ lập trình Python

Giới thiệu môn học

Tuần 1 — **Giới thiệu, ML và Python**

- Lược sử và thành tựu của AI, ML
- Ôn tập ngôn ngữ Python

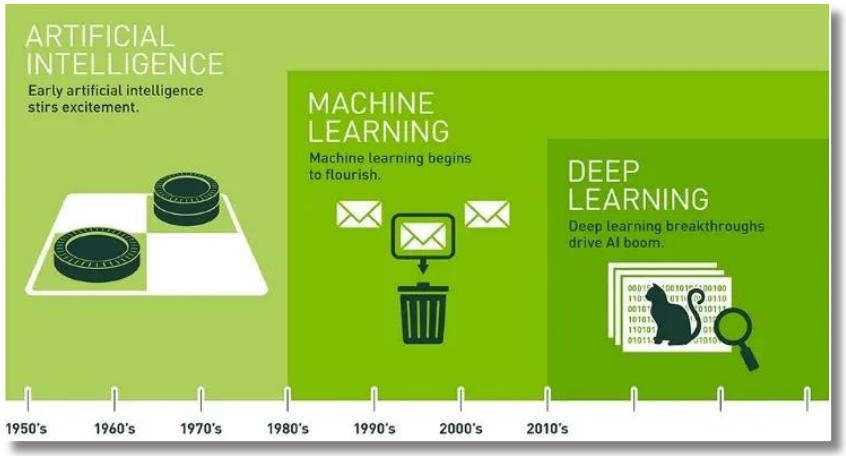
Tuần 2 — **Lập trình Numpy và Matplotlib**

Tuần 3 — **Quy trình xây dựng mô hình máy học**

Tuần 4 — **Tiền xử lý dữ liệu và đánh giá mô hình**

Tuần 5-6 — **Mô hình học không giám sát**

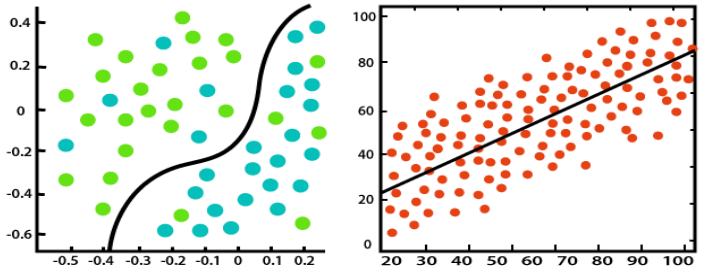
Nguồn: Nvidia



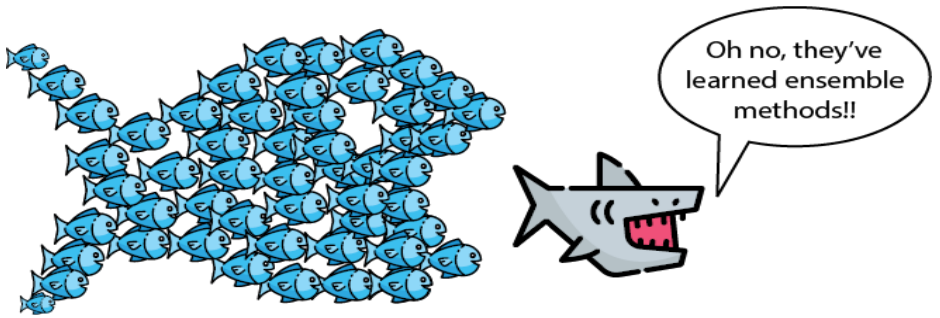
Giới thiệu môn học

Tuần 7-10	Mô hình học có giám sát
Tuần 11-12	Tối ưu siêu tham số
Tuần 13	Ensemble model: Học tổng hợp
Tuần 14-15	Báo cáo đồ án cuối kỳ Ôn tập

Nguồn: static.javatpoint.com



Hyperparameter
vs
Parameter



Nguồn: livebook.manning.com

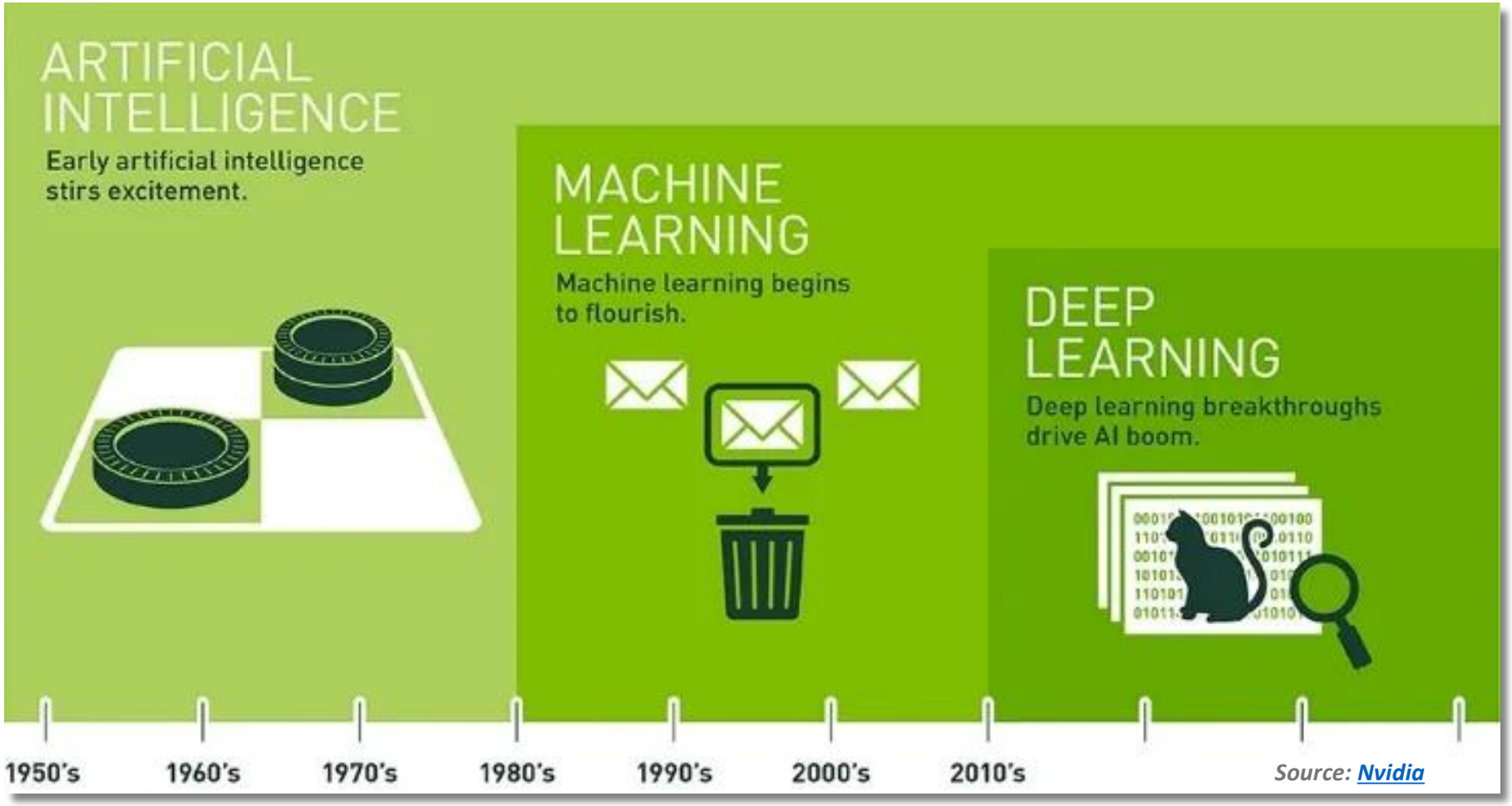
Hình thức đánh giá

Bài tập 30%	Đồ án 30%	Cuối kỳ 40%
• Quiz • Bài tập lập trình	• Làm theo nhóm • Nộp cuối kỳ	• Thi viết • Đề đóng

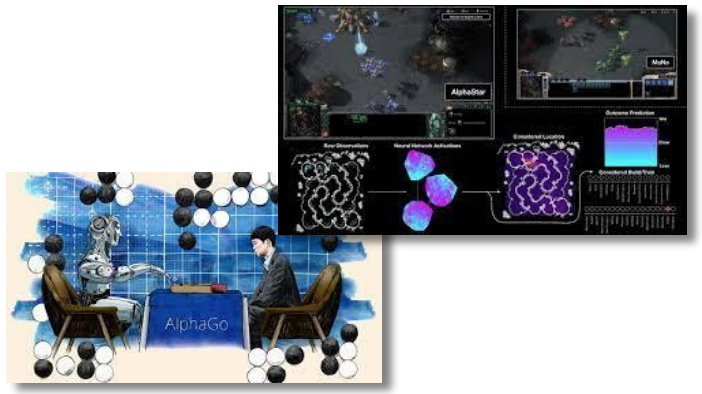
NỘI DUNG

1. Giới thiệu môn học
2. **Lịch sử và thành tựu của máy học**
3. Ôn tập ngôn ngữ lập trình Python

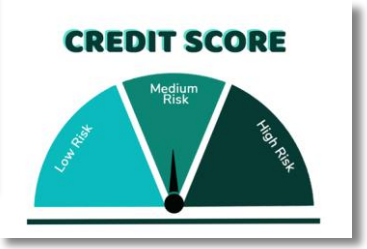
Lịch sử hình thành



Một số thành tựu

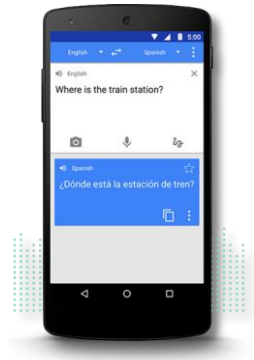


Robotics



Xử lý dữ liệu dạng bảng

Xử lý ảnh

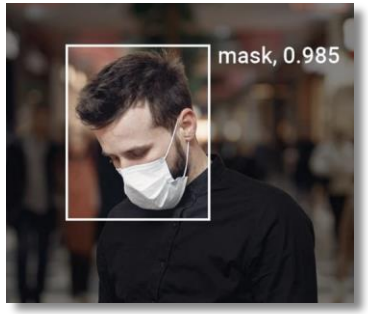


Văn bản

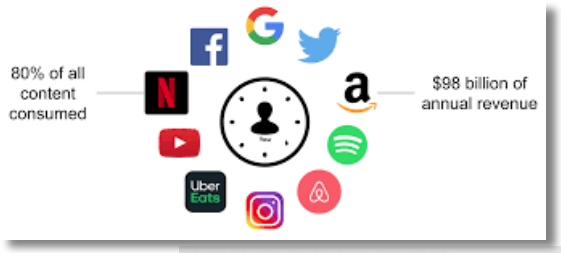
Âm thanh

Xử lý ngôn ngữ tự nhiên

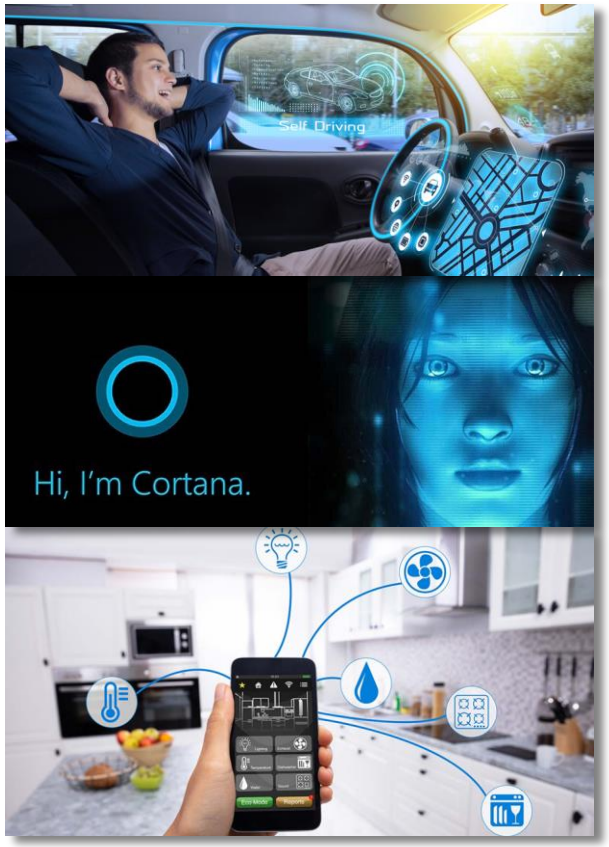
Thành tựu AI



Y sinh, Chăm sóc sức khỏe



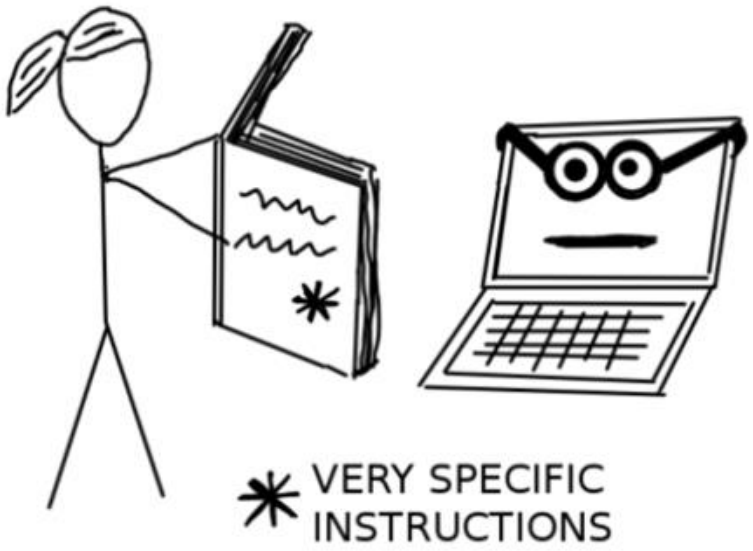
Tài chính, thương mại điện tử



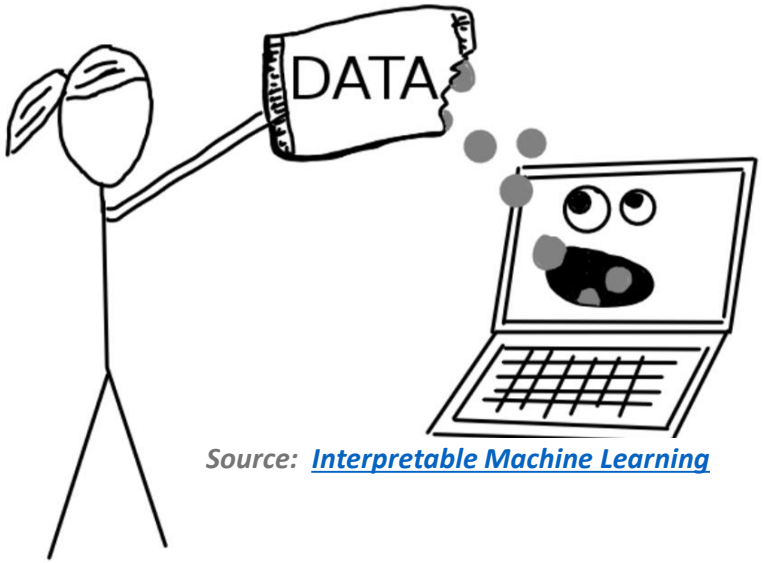
Môi trường “thông minh"

So sánh

Lập trình truyền thống



Học máy



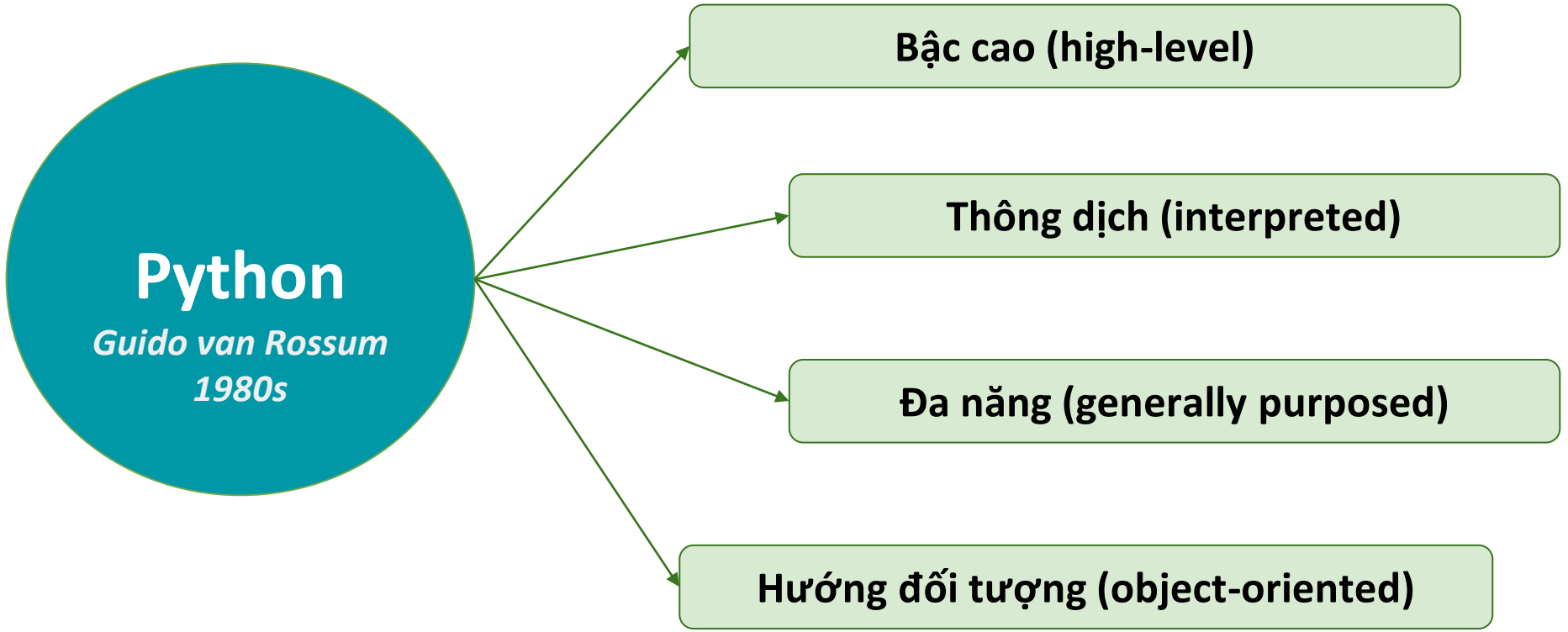
Source: [Interpretable Machine Learning]

Source: [Datalya]

NỘI DUNG

1. Giới thiệu môn học
2. Lịch sử và thành tựu của máy học
3. **Ôn tập ngôn ngữ lập trình Python**

Ôn tập Python



Ôn tập Python

Top lý do nên học Python

Đơn giản

Rõ ràng

Dễ học

Đa thư viện

Cộng đồng lớn

Những gã khổng lồ dùng Python

- Python where we can, C++ (Go) where we must
- Python Django Web Framework
- Multiple services in infrastructure management
- Back-end networking
- Server-side data analysis

Giới thiệu Python

C ++

```cpp
#include <iostream.h>

int main()
{
    std::cout << "Hello, world!";
    return 0;
}
```

Java

```java
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

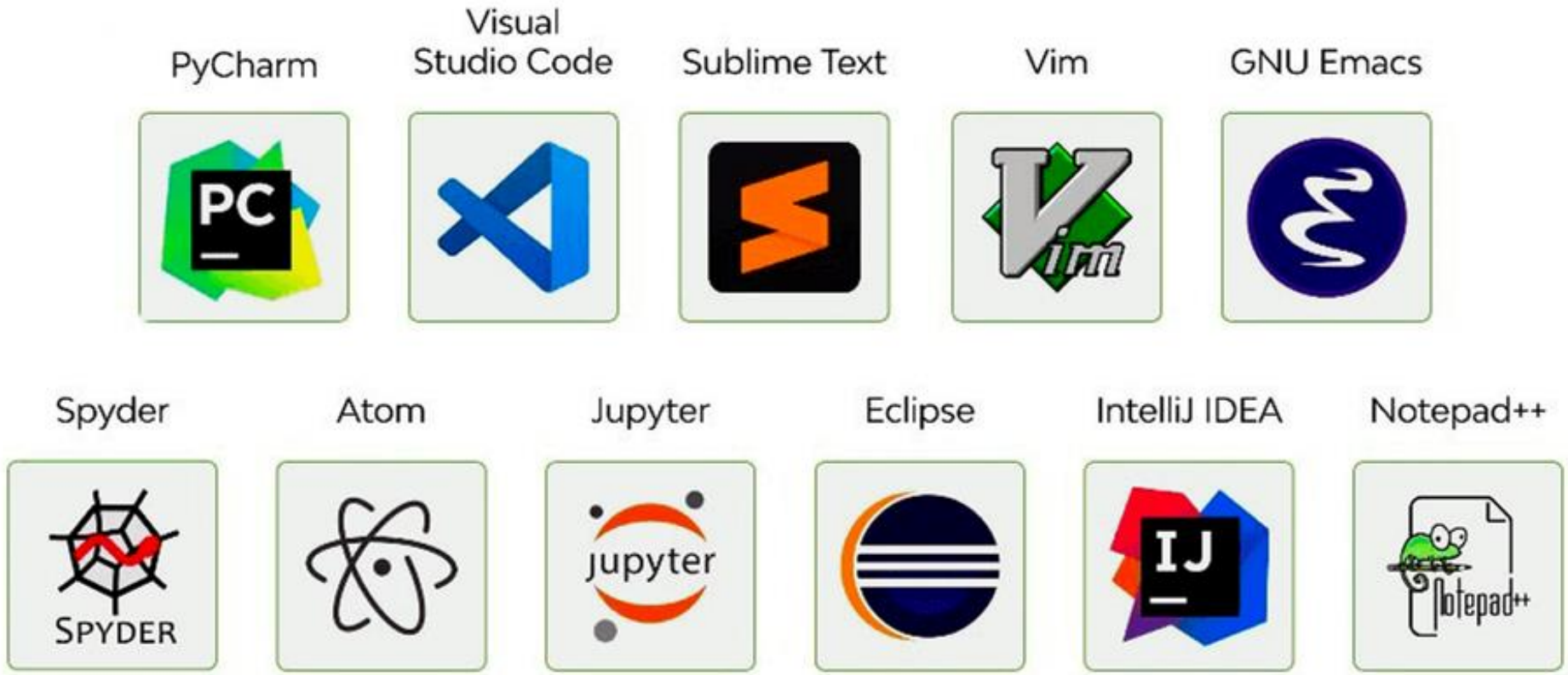
C#

```csharp
using System;
class Program
{
    public static void Main(string[] args)
    {
        Console.WriteLine("Hello, world!");
    }
}
```

Python

```python
print("Hello, World!")
```

Môi trường lập trình



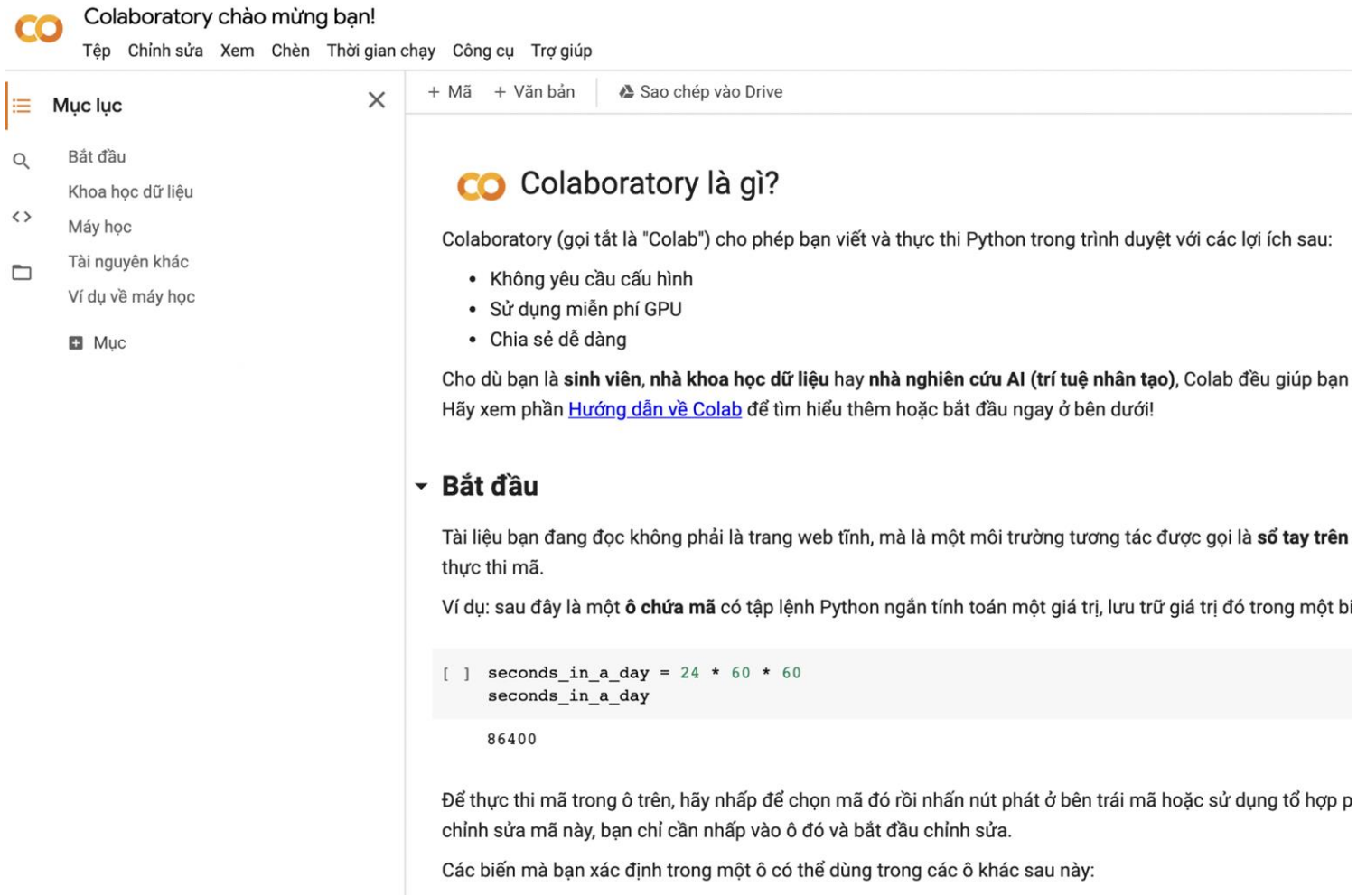
Source: [Top 10 Python IDEs and Code editors]

Môi trường lập trình

- Nền tảng miễn phí bởi Google
- **Jupyter notebook** bất kỳ đâu
- Chia sẻ dễ dàng

Truy cập **Colab** tại địa chỉ:

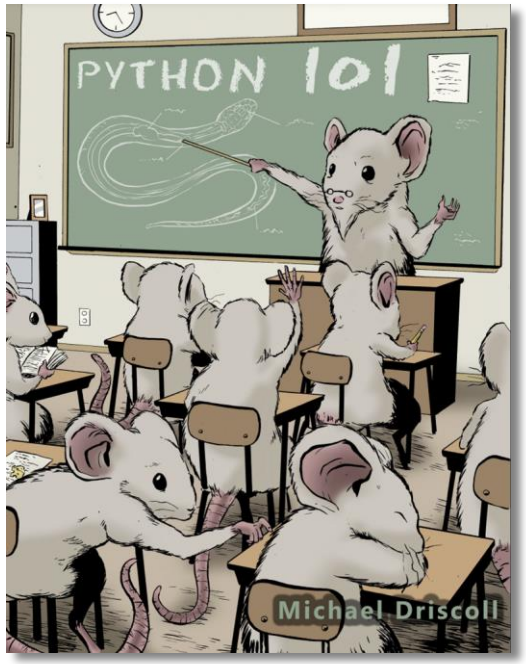
colab.research.google.com



Practice 1, practice 2

Tham khảo

1. Google Colab: colab.research.google.com, practice 1, practice 2.
2. Sách hay về Python cho người mới bắt đầu



Kiểu dữ liệu

Kiểu dữ liệu	Ý nghĩa	Ví dụ
`int`	Kiểu số nguyên (không có chứa dấu chấm thập phân)	`11`, `-15`
`float`	Kiểu số thực (có chứa dấu chấm thập phân)	`3.14`, `4.02`
`complex`	Kiểu số phức	`1 + 2j`, `3 - 4j`
`str`	Kiểu chuỗi	`'Python'`
`bool`	Kiểu luận lý	`TRUE`, `FALSE`

- Kiểm tra kiểu dữ liệu của biến bằng hàm `type()`
- Ép kiểu dữ liệu của biến

Cú pháp	Ý nghĩa
`float(x)`	chuyển đổi sang kiểu số thực
`int(x)`	chuyển đổi sang kiểu số
`str(x)`	chuyển đổi sang dạng chuỗi
`complex(x)`	chuyển đổi sang kiểu phức hợp
`tuple(x)`	chuyển đổi sang kiểu Tuple
`dict(x)`	chuyển đổi sang kiểu Dictionary
`hex(x)`	chuyển đổi sang hệ 16
`oct(x)`	chuyển đổi sang hệ 8
`chr(x)`	chuyển đổi sang dạng ký tự

Nguồn: Zootopi

Toán tử

Phép toán	Biểu thức	Ý nghĩa
+	`x + y`	Phép cộng
-	`x - y`	Phép trừ
*	`x * y`	Phép nhân
/	`x / y`	Phép chia
%	`x % y`	Phép chia lấy phần dư
//	`x // y`	Phép chia làm tròn xuống
**	`x ** y`	Phép mũ

Toán tử	Biểu thức	Ý nghĩa
<	`x < y`	So sánh giá trị x có nhỏ hơn y hay không
>	`x > y`	So sánh giá trị x có lớn hơn y hay không
<=	`x <= y`	So sánh giá trị x có nhỏ hơn hoặc bằng y hay không
>=	`x >= y`	So sánh giá trị x có lớn hơn hoặc bằng y hay không
==	`x == y`	So sánh giá trị x có bằng y hay không
!=	`x != y`	So sánh giá trị x có khác y hay không

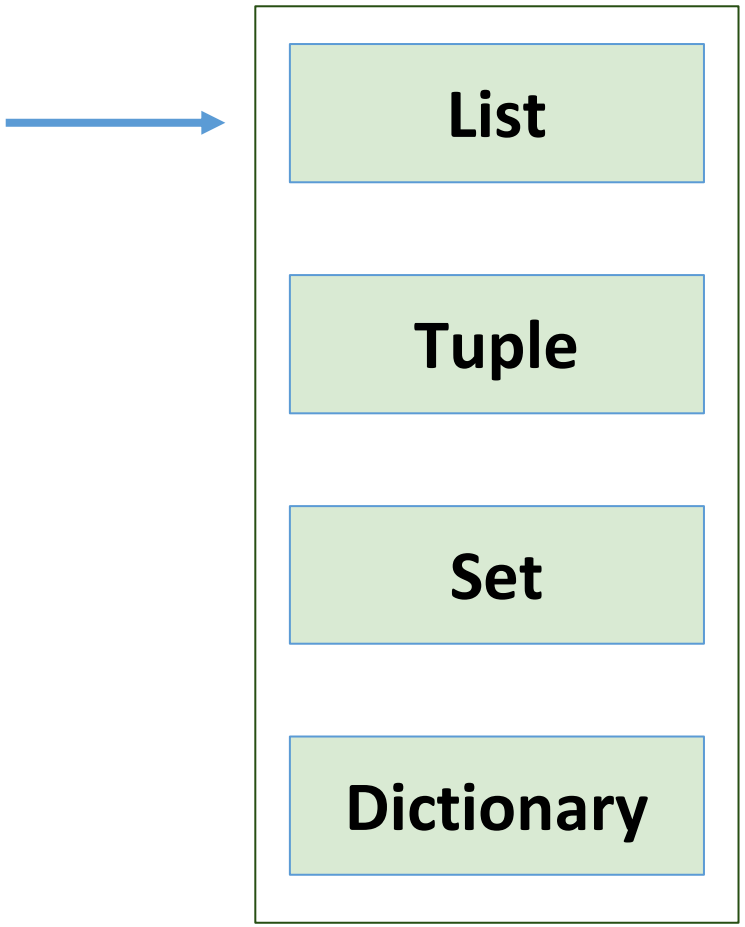
Nguồn: Zootopi

Toán tử

Toán tử	Biểu thức	Ý nghĩa
AND	`X and Y`	True nếu cả X và Y đều đúng
OR	`X or Y`	True nếu ít nhất một trong hai vế X hoặc Y đúng
NOT	`not X`	True nếu X sai và False nếu X đúng
XOR	`bool(Y) ^ bool(Y)`	True nếu X và Y cùng giá trị và False nếu ngược lại

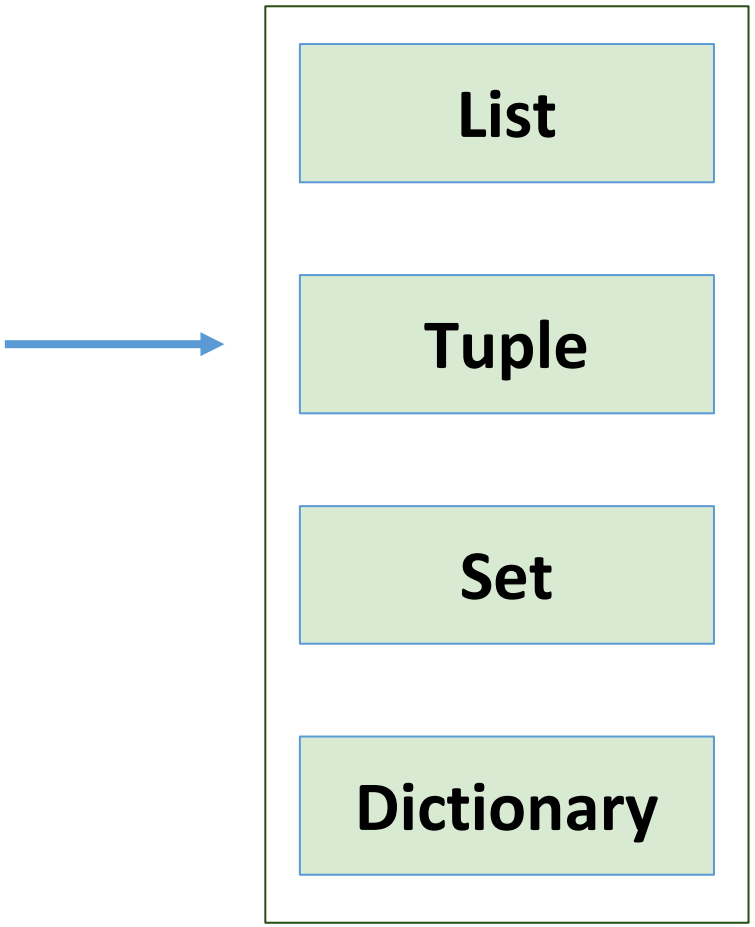
- Thứ tự ưu tiên của các loại toán tử trên tương tự như trong toán học.

Cấu trúc dữ liệu



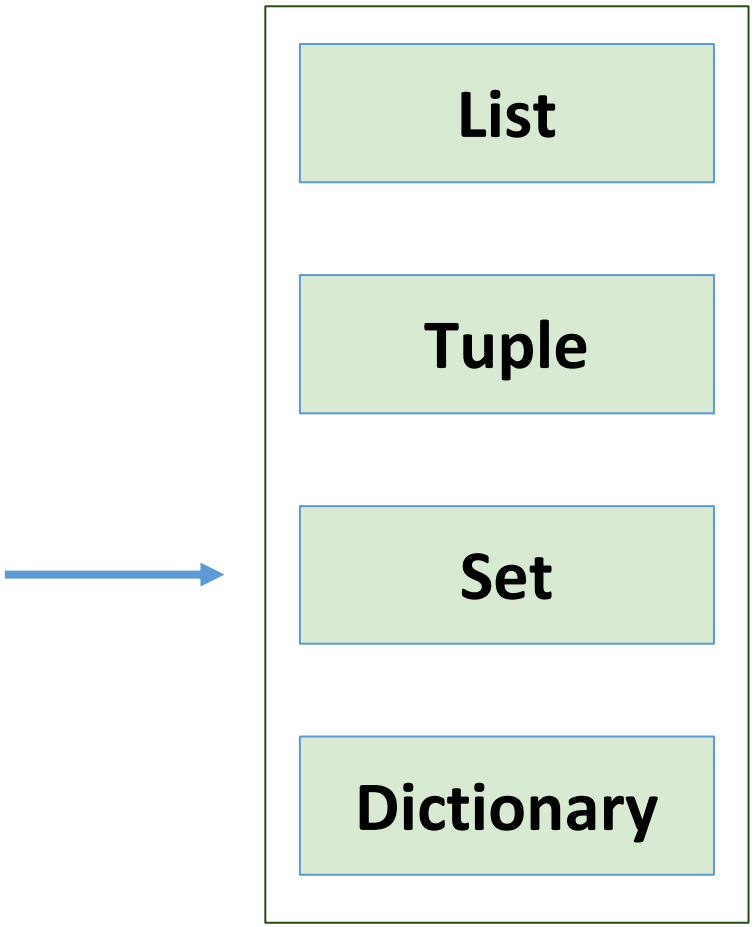
- **List** là một đối tượng dùng để lưu tập các đối tượng khác nhau, có thể thêm, bớt, thay đổi các giá trị và chứa các giá trị trùng nhau.
- Cú pháp: `[giá_trị_1, giá_trị_2, ...]`
- Ví dụ:
  - Khai báo một list các số tự nhiên từ 1 đến 10.
  - Lọc phần tử ở vị trí thứ 5.
  - Lọc lấy 5 phần tử đầu tiên của list.

Cấu trúc dữ liệu



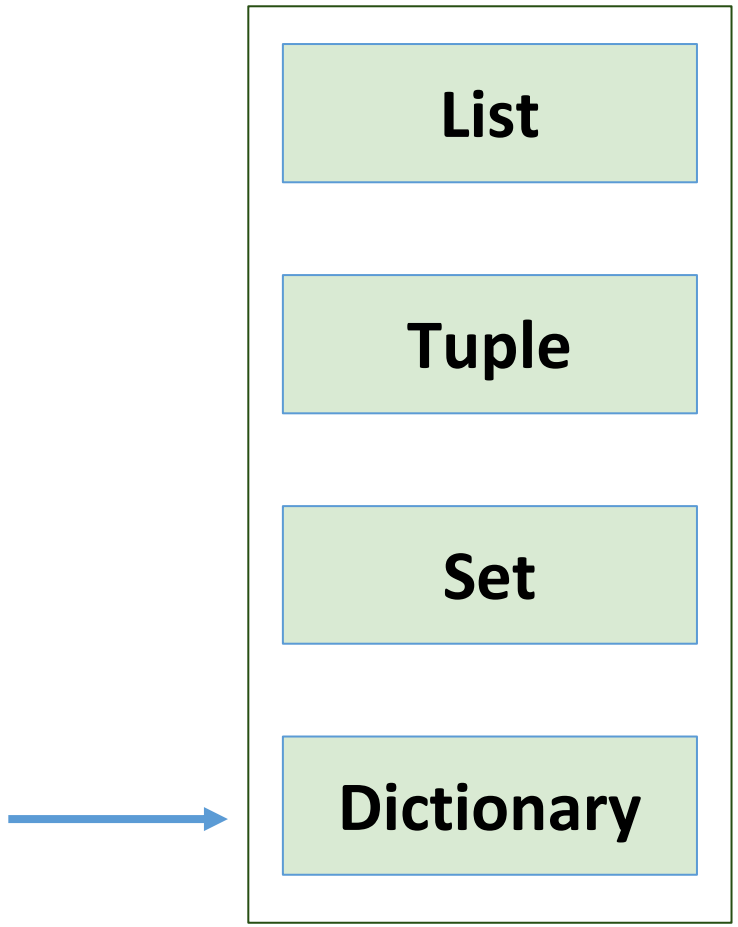
- **Tuple** là một đối tượng dùng để lưu tập các đối tượng khác nhau. Tuple không thể thêm, bớt, thay đổi các giá trị
- Cú pháp: `(giá_trị_1, giá_trị_2,... )`
- Ưu điểm:
  - **Bộ nhớ** khi dùng Tuple nhỏ hơn so với List.
  - Tốc độ xử lý khi dùng Tuple nhanh hơn so với List.

Cấu trúc dữ liệu



- **Set** là một đối tượng dùng để lưu tập các đối tượng khác nhau, **không** thể chứa các phần tử **trùng nhau.**
- Cú pháp: `{giá_trị_1, giá_trị_2,... }`
- Phù hợp xử lý các bài toán về tập hợp: *giao, hiệu, phần bù, ...*

Cấu trúc dữ liệu



- **Dictionary** là một danh sách các phần tử, trong đó mỗi phần tử là một cặp khóa và giá trị. Mỗi khoá thì chỉ ứng với một giá trị duy nhất.
- Cú pháp:

```
{khoá_1 : giá_trị_1, khoá_2 : giá_trị_2,... }
```

- Ví dụ:
  - Khai báo thông tin của một học sinh bằng Dictionary.

Cấu trúc dữ liệu

	List	Tuple	Set	Dictionary
Đặc trưng	Chứa bất kì kiểu dữ liệu nào, CÓ thứ tự, giá trị CÓ thể thay đổi.	Giá trị KHÔNG thể thay đổi, CÓ thứ tự, KHÔNG thể thay đổi.	Giá trị KHÔNG trùng lặp, CÓ thứ tự, CÓ thể thay đổi.	Theo cơ chế khoá: giá trị, khoá KHÔNG trùng lặp.
Cú pháp	`[]`	`()`	`{}`	`{Khoá: Giá trị}`
Mục đích	Lưu trữ danh sách.	Lưu các tập hợp không bị thay đổi, cho tốc độ nhanh.	Thực hiện các phép toán tập hợp.	Lưu trữ các dữ liệu có cấu trúc.

Bảng so sánh các cấu trúc dữ liệu cơ bản trong Python

Cấu trúc rẽ nhánh

```
if condition(True/False):
    business logic
elif condition(True/False): # --> optional
    business logic
elif condition(True/False): # --> optional
    business logic
elif condition(True/False): # --> optional
    business logic
    ....
else:
    business logic # --> optional
```

- Nếu sinh viên làm bài cuối khoá được hơn 8 điểm, thì học viên qua môn xuất sắc, in ra *'Excellently Passed'.*
- Nếu sinh viên làm bài cuối khoá được từ 6 đến 8 điểm, thì học viên qua môn, in ra *'Well Passed'.*
- Nếu không, học viên trượt môn, in ra Không Đạt ('Failed')

Cấu trúc vòng lặp

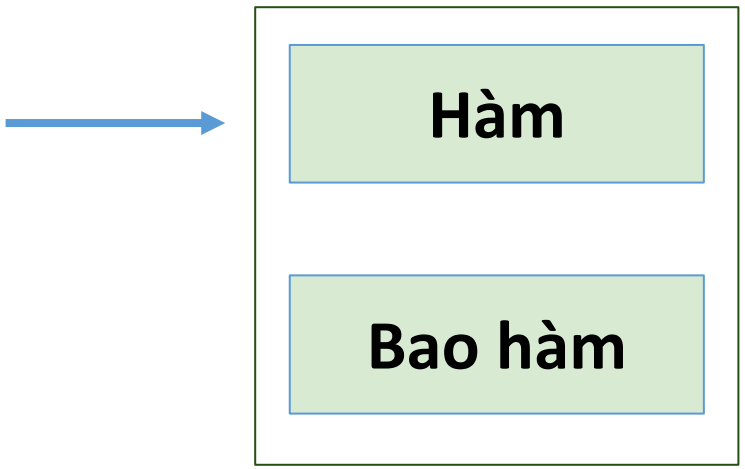
```
for value in iterable:
    business logic

for index in range(len(iterable)):
    business logic
```

```
while condition(True/False):
    business logic
```

Biết `pi = 3.14`, tính diện tích hình tròn với danh sách các bán kính như sau

```
radius = [2, 4, 5, 8, 10, 20]
```

Hàm

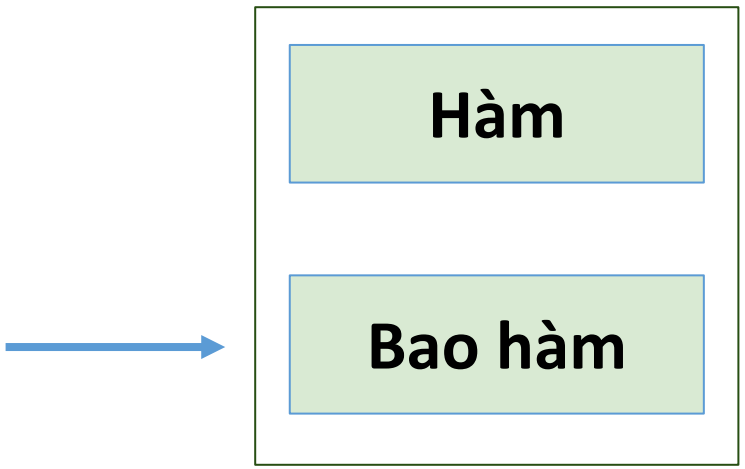


- Cú pháp:

```
def function_name(arg1, arg2, ...):
    business logic
    return result
```

- Ưu điểm:
  - Tái sử dụng mã nguồn
  - Dễ dàng quản lý mã nguồn

Hàm

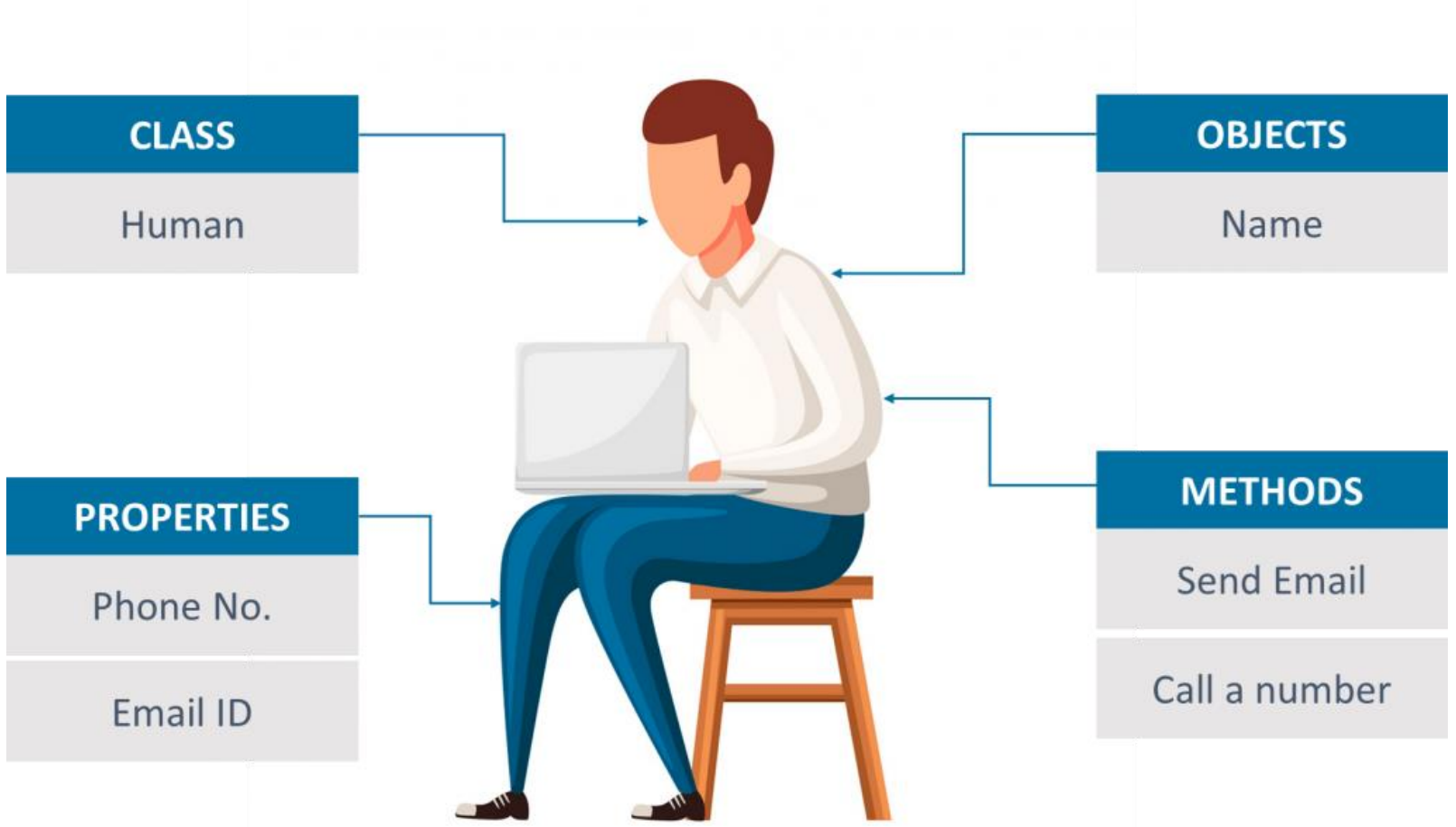


- Cú pháp:

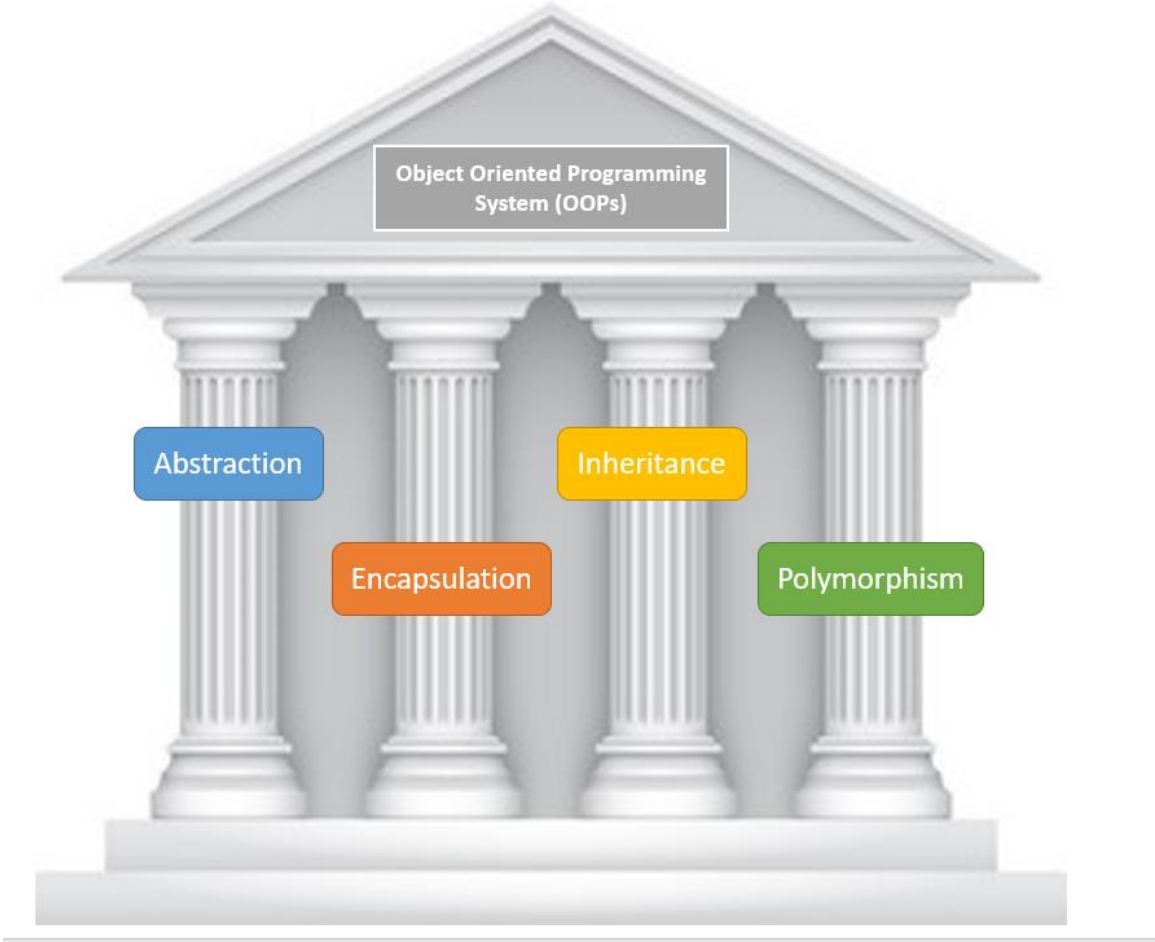
```
[expression for variable in iterable]
{expression for variable in iterable}
tuple(expression for variable in iterable)
{key:value for variable in iterable}
```

- Ưu điểm so với hàm thông thường:
  - Đoạn mã ngắn gọn hơn
  - Tốc độ xử lý nhanh hơn

Giới thiệu OOP



Tính chất OOP



Object

- Python hỗ trợ nhiều kiểu dữ liệu khác nhau:

  1234 3.14159 “Hello" [1, 5, 7, 11, 13]

- Mỗi ví dụ trên gọi là một object và mỗi object có:
  - Một kiểu dữ liệu (type)
  - Tập hợp các hàm/thủ tục để tương tác với object
- Một object là một instance (thực thể) của một kiểu dữ liệu
  - 1234 là một instance của kiểu số nguyên (int)
  - “Hello" là một instance của kiểu chuỗi (string)

Object

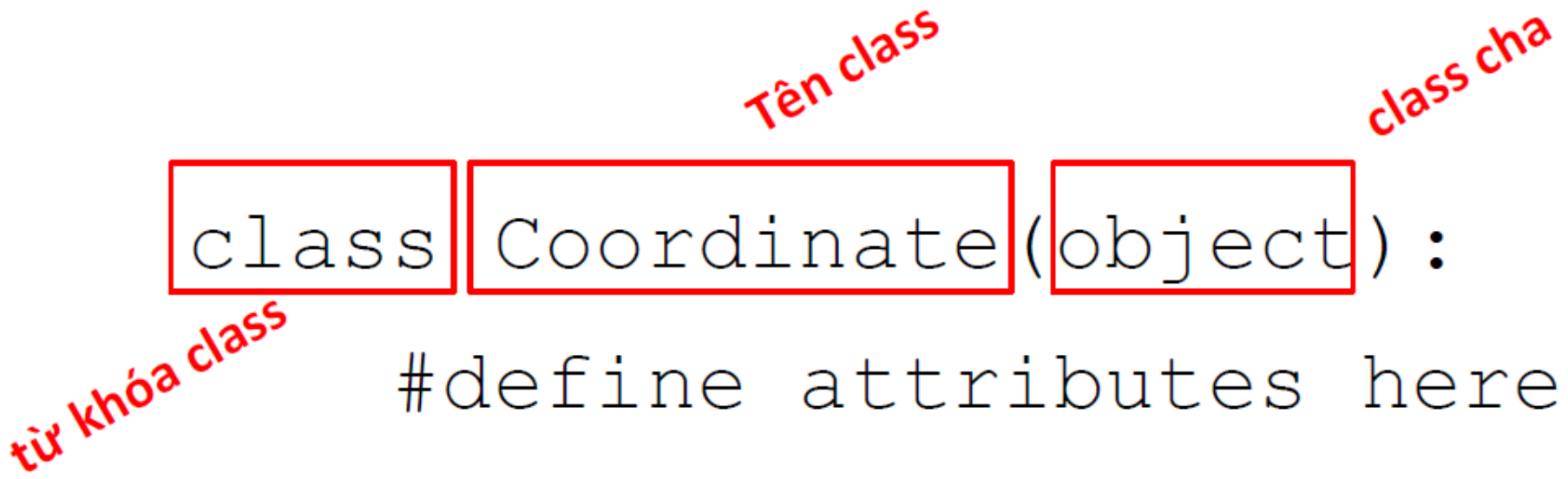
- Object là dữ liệu trừu tượng bao gồm:
  - Dữ liệu nội bộ (internal representation), được truy xuất thông qua thuộc tính dữ liệu
  - Giao diện (interface) để tương tác với object thông qua các phương thức (method), còn được gọi là hàm/thủ tục (function/procedure)

Class

- *Việc tạo class* bao gồm:
  - Định nghĩa **tên class**
  - Định nghĩa **các thuộc tính**
  - Định nghĩa **các phương thức**

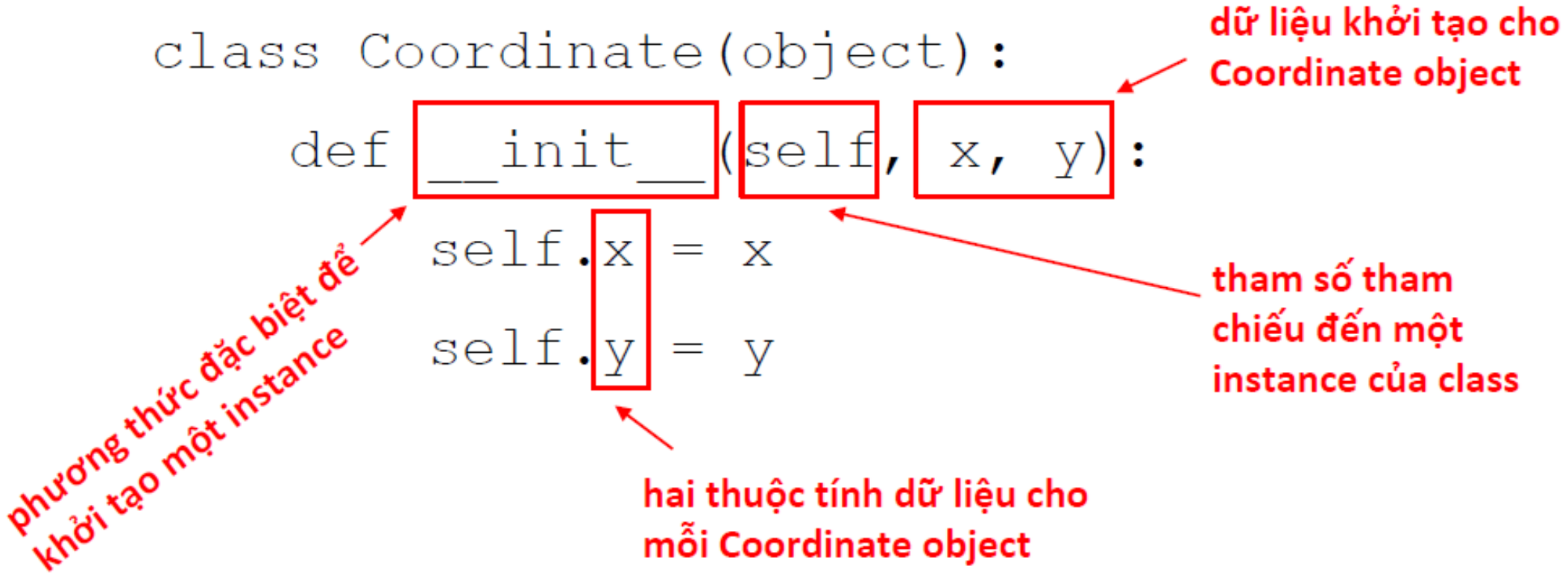
- *Việc sử dụng class* bao gồm:
  - Tạo một instance mới của class đó
  - Thực hiện các toán tử/phương thức trên instance vừa tạo

Cài đặt Class



Cài đặt phương thức __init__

- Sử dụng phương thức đặc biệt __init__ để khởi tạo các thuộc



Tạo và thao tác trên instance

```
c = Coordinate(3,4)
origin = Coordinate(0,0)
print(c.x)
print(origin.x)
```

tạo một object mới có kiểu Coordinate và truyền 3 và 4 vào phương thức __init__

sử dụng dấu chấm "." để truy cập thuộc tính của instance c

Phương thức

- Phương thức (method) của class là các hàm chỉ làm việc với class đó
- Sử dụng self làm tham số đầu tiên của tất cả các method
- Toán tử "." được sử dụng để truy cập các thuộc tính dữ liệu và phương thức của một object

Phương thức

- Phương thức (method) của class là các hàm chỉ làm việc với class đó
- Sử dụng self làm tham số đầu tiên của tất cả các method
- Toán tử "." đư ' liệu và phương thức c

```
class Coordinate(object):
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def distance(self, other):
        x_diff_sq = (self.x-other.x)**2
        y_diff_sq = (self.y-other.y)**2
        return (x_diff_sq + y_diff_sq)**0.5
```

sử dụng self để tham chiếu đến chính instance đó

các tham số khác của phương thức

dấu chấm để truy cập dữ liệu

Gọi phương thức

```
def distance(self, other):
    # code here
```

định nghĩa phương thức

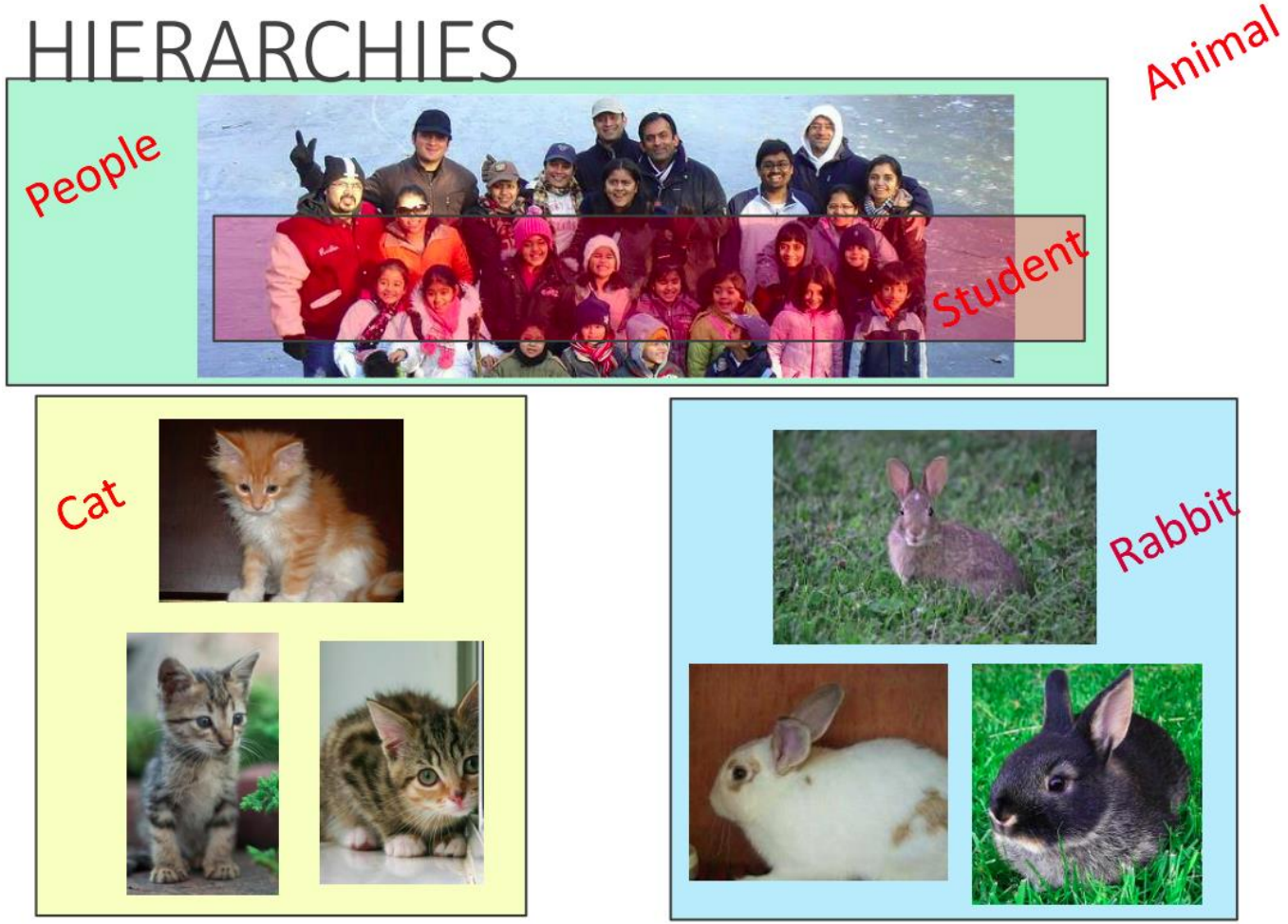
```
c = Coordinate(3,4)
zero = Coordinate(0,0)
print(c.distance(zero))
```

object gọi hàm

tên phương thức

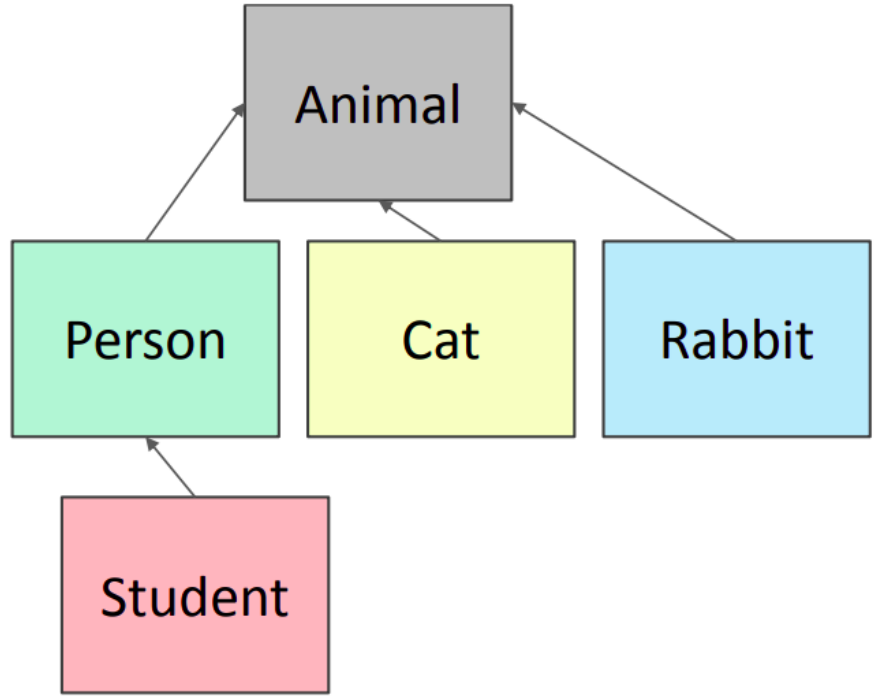
các tham số không bao gồm self
(self ám chỉ đến object c)

Kế thừa



Kế thừa

- Lớp cha (parent class hay superclass)
- Lớp con (child class hay subclass)
  - Kế thừa tất cả thuộc tính dữ liệu và hành vi/phương thức của lớp cha
  - Có thể thêm thuộc tính mới
  - Có thể thêm phương thức mới
  - Có thể ghi đè (override) phương thức cũ



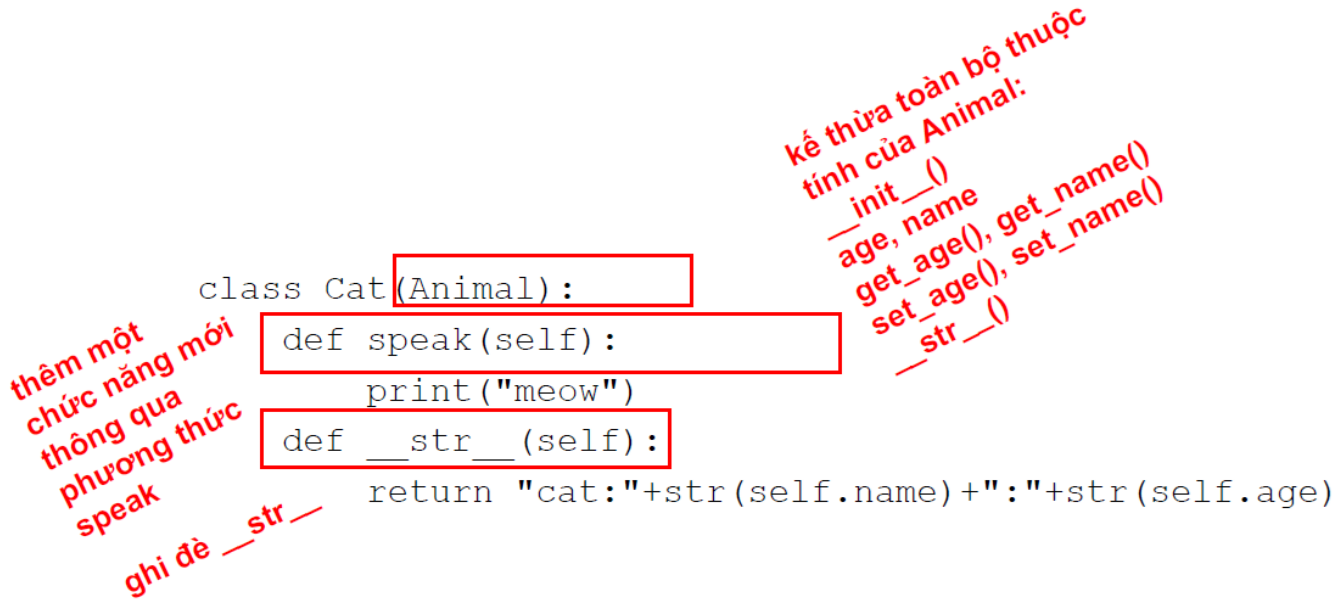
Kế thừa

- Lớp cha (parent class hay superclass)

```
class Animal(object):
    def __init__(self, age):
        self.age = age
        self.name = None
    def get_age(self):
        return self.age
    def get_name(self):
        return self.name
    def set_age(self, newage):
        self.age = newage
    def set_name(self, newname=""):
        self.name = newname
    def __str__(self):
        return "animal:"+str(self.name)+":"+str(self.age)
```

Kế thừa

- Lớp con (child class hay subclass)



- Thêm mới phương thức speak(), instance của lớp Cat có thể sử dụng phương thức này trong khi instance của lớp Animal thì không thể

BÀI QUIZ VÀ HỎI ĐÁP