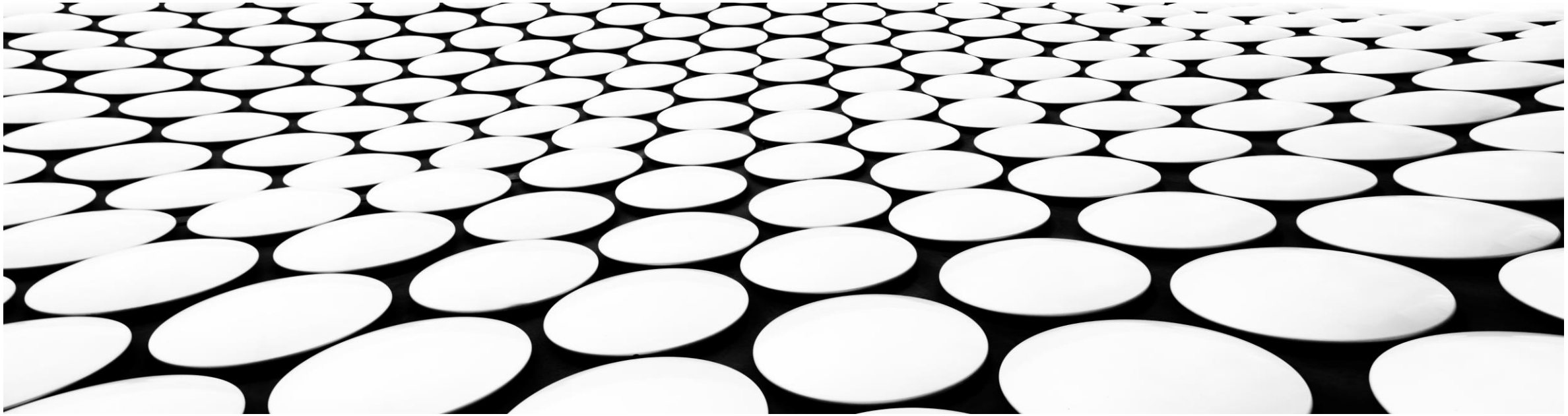
KĨ THUẬT LẬP TRÌNH PYTHON

NGUYỄN MẠNH HÙNG



CHƯƠNG 3.2: CLASS (LỚP)

CLASS LÀ GÌ?

- Class là kiểu dữ liệu cho phép người dùng tự định nghĩa, phù hợp với cách tổ chức, lưu trữ và xử lý dữ liệu của mình.
- Class là cơ sở của lập trình hướng đối tượng (OOP – Object Oriented Programming)

ĐỊNH NGHĨA CLASS

Mỗi lớp được thiết kế bao gồm 2 thành phần chính:

- **Thuộc tính**: các **biến** dùng để lưu trữ dữ liệu, thông tin mô tả đối tượng.
- **Phương thức**: **chương trình con** để tính toán, truy xuất thông tin của đối tượng.

Cú pháp:	Tạo đối tượng:
Class <tên lớp> : **[định nghĩa phương thức]**	**<tên đối tượng>** = **<tên lớp>**([tham số])

VÍ DỤ1: XÚC XẮC NHIỀU MẶT

- Thiết kế một ***class*** tổng quát **MSDie** để mô hình hoá **xúc xắc nhiều mặt**.
- Mỗi đối tượng **MSDie** chứa 2 thuộc tính sau:
  - **Số mặt** của xúc xắc
  - **Giá trị** hiện thời của xúc xắc
- Ta có thể thao tác xúc xắc bằng 3 phương thức sau:
  - Gieo xúc xắc (**roll**)
  - Đặt giá trị (**setValue**)
  - Lấy giá trị (**getValue**)

```
'''
Định nghĩa Class MSDie mô tả xúc xắc nhiều mặt có:
1) Thuộc tính:
    +) Số mặt
    +) Giá trị hiện tại
2) Phương thức:
    +) roll (gieo)
    +) setValue (đặt giá trị)
    +) getValue (lấy giá trị)
'''

from random import randint
class MSDie:
    def __init__(self , sides):
        self.sides = sides
        self.value = 1

    def roll(self):
        self.value = randint(1,self.sides)

    def getValue(self):
        return self.value

    def setValue(self , value):
        self.value = value
```

PHƯƠNG THỨC

- Phương thức được định nghĩa giống như các hàm thông thường.
- Trong class MSDie, các phương thức có tham số đầu tiên được đặt tên là ***self***, tham chiếu đến đối tượng mà phương thức đang tác động. Tham số này có thể đặt tên thế nào tuỳ ý, nhưng thường đặt là ***self***.
- Phương thức **__init__** là hàm tạo (hàm cấu trúc) của đối tượng. Python gọi phương thức này để khởi tạo một đối tượng mới. Mục đích của hàm tạo là để cung cấp giá trị ban đầu cho các biến của đối tượng. Ví dụ:

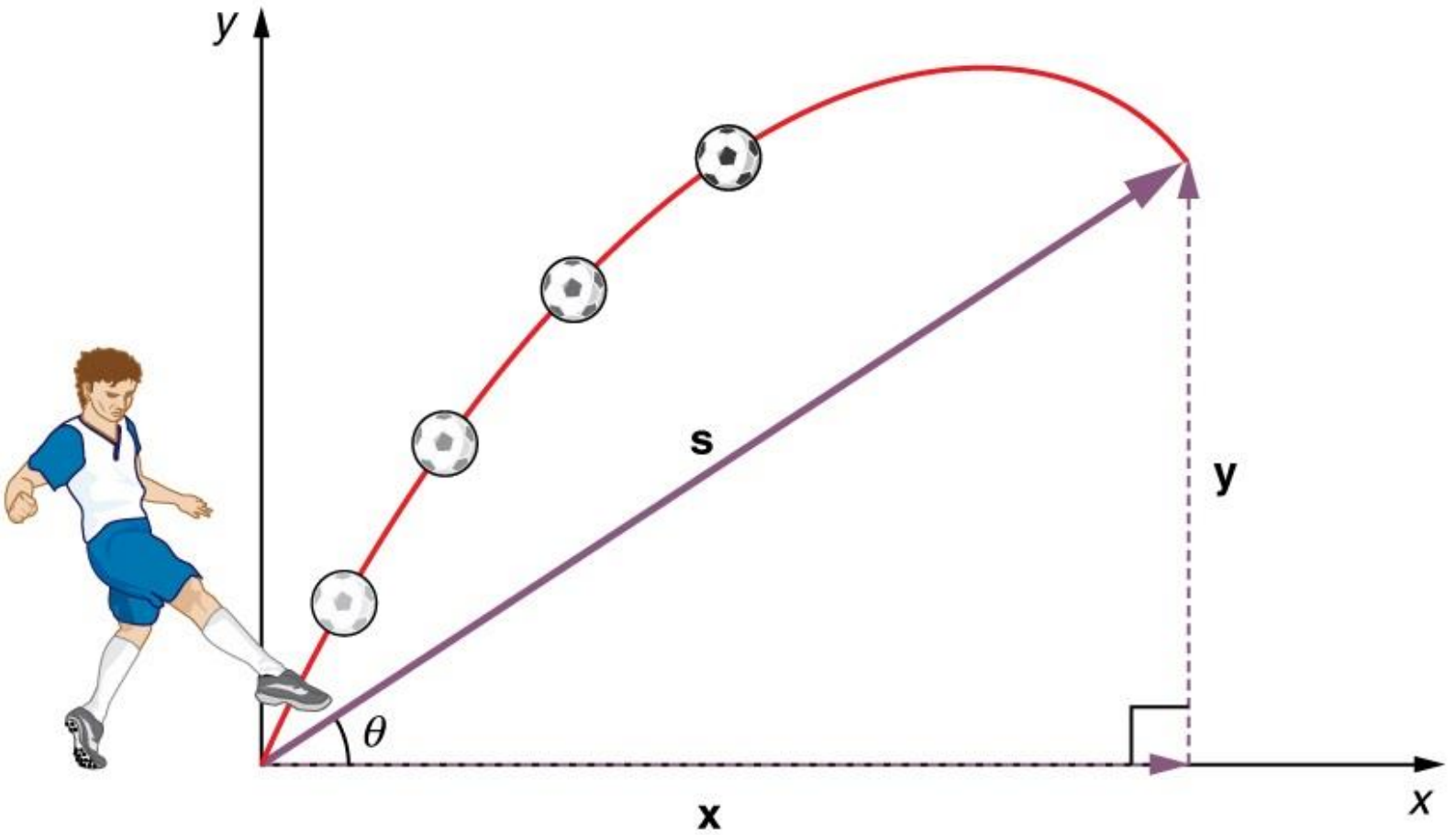
*die1 = **MSDie**(6)*

VÍ DỤ 2: CHUYỂN ĐỘNG QUẢ BÓNG

- Thiết kế một ***class*** **Projectile** để mô tả **chuyển động một quả bóng**.
- Tạo đối tượng quả bóng với góc, vận tốc, độ cao cho trước:

cball = **Projectile**(angle, velocity, height)

- Class Projectile có hàm tạo **__init__** sử dụng các giá trị này để khởi tạo giá trị cho các biến của **cball**, bao gồm: (***xpos***, ***ypos***, ***xvel***, ***yvel***).
- Class Projectile có phương thức **update** để thay đổi trạng thái của quả bóng, các phương thức **getX**, **getY** để tìm vị trí hiện tại của quả bóng.



$$v_x = v_0 \cos\alpha$$

$$v_y = v_0 \sin\alpha - gt$$

$$x = v_0 \cos\alpha \cdot t$$

$$y = v_0 \sin\alpha \cdot t - \frac{1}{2} g t^2$$

```
import math
class Projectile:

    def __init__(self,angle,velocity,height):
        self.xpos = 0.0
        self.ypos = height
        theta = math.radians(angle)
        self.xvel = velocity*math.cos(theta)
        self.yvel = velocity*math.sin(theta)

    def getX(self):
        return self.xpos

    def getY(self):
        return self.ypos

    def update(self,time):
        self.xpos += time*self.xvel
        yvel_ = self.yvel - 9.8*time
        self.ypos += time * (self.yvel + yvel_)/2.0
        self.yvel = yvel_
```

VÍ DỤ 3: XỬ LÝ DỮ LIỆU SINH VIÊN

- Thiết kế một ***class*** **Student** chứa dữ liệu về sinh viên như Tên (name), Số tín chỉ (credits), Điểm đánh giá (points). Class có các phương thức truy xuất thông tin sinh viên: getName, getCredits, getPoints, và có phương thức tính điểm trung bình (GPA) của sinh viên bằng cách lấy Điểm đánh giá chia cho Số tín chỉ.
- Giả sử ta có 1 file ghi thông tin sinh viên (cách nhau 1 Tab):

```
Adams, Henry      127     228
Computewell, Susan        100       400
DibbleBit, Denny          18        41.5
Jones, Jim        48.5    155
Smith, Frank      37      125.33
```

- Viết một chương trình đọc dữ liệu từ file, thông tin về mỗi sinh viên được ghi vào một đối tượng Student.
- Chương trình sẽ tìm sinh viên có điểm GPA cao nhất và in ra màn hình Tên, số tín chỉ, và điểm GPA của sinh viên đó.

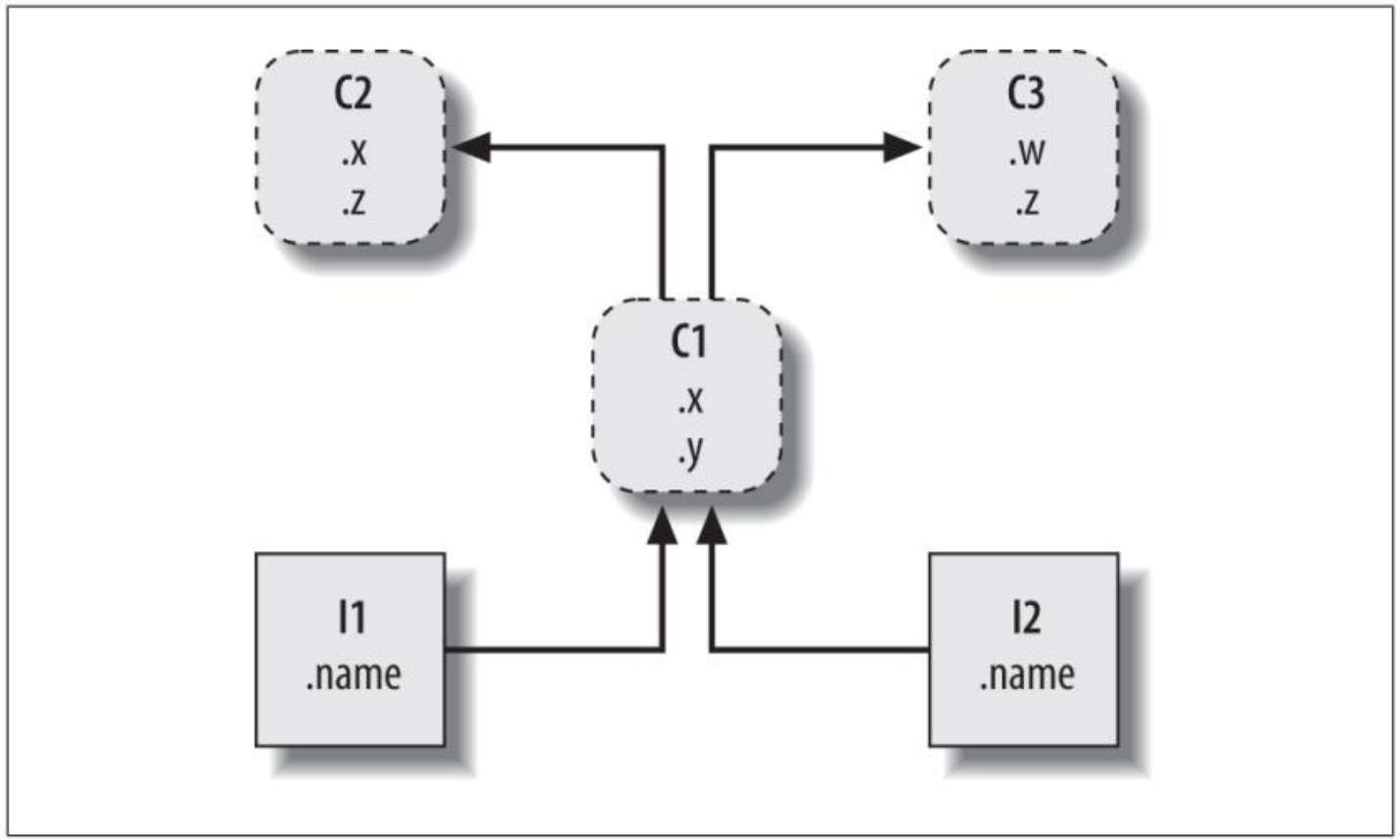
```
class Student:

    def __init__(self, name, credits, points):
        self.name = name
        self.credits = float(credits)
        self.points = float(points)

    def getName(self):
        return self.name

    def getCredits(self):
        return self.credits

    def getPoints(self):
        return self.points

    def gpa(self):
        return self.points/self.credits
```

```python
# Hàm đọc dữ liệu từ file và tạo đối tượng Student
def makeStudent(infoStr):
    # infoStr is a tab-separated line: name credits points
    # returns a corresponding Student object
    name, credits, points = infoStr.split("\t")
    return Student(name, credits, points)
```

```
filename = input("Mở file xử lý thông tin: ")
file = open(filename)  # mở file

# gán sinh viên giỏi nhất cho sinh viên đầu tiên trong file
best = makeStudent(file.readline())

# xử lý các dòng tiếp theo của file
for line in file:
    # ghi thông tin của dòng vào 1 đối tượng Student
    s = makeStudent(line)
    # nếu sinh viên là tốt nhất, ghi lại
    if s.gpa() > best.gpa():
        best = s

file.close()    #đóng file

# print information about the best student
print ("Sinh viên giỏi nhất là: ", best.getName())
print ("số tín chỉ: ", best.getCredits())
print ("Điểm trung bình: ", best.gpa())
```

CẤU TRÚC CÂY CLASS

- Trong OOP, các class thường được thiết kế lồng nhau.
- C2, C3 là lớp cha (**superclass**)
- I1, I2 là thực thể (đối tượng) của class C1.



VÍ DỤ 4: KẾ THỪA, TUỲ CHỈNH, MỞ RỘNG

- Đầu tiên, thiết kế 1 ***class*** **Employee**, với các thuộc tính ***name*** (Tên), ***job*** (Công việc), ***pay*** (Lương), và phương thức **info** (thông tin Tên + Lương), **giveRaise** (tăng lương với tỉ lệ phần trăm cho trước).
- Tiếp theo, thiết kế 1 ***class*** **Manager**, là lớp con của lớp **Employee**, khai báo:

  class Manager(**Employee**):

  Phương thức **giveRaise** được tuỳ chỉnh bằng cách tăng thêm hệ số quản lý.
- Cuối cùng, thiết kế 1 ***class*** **Team**, tổ chức nhóm làm việc, với các phương thức **addMember** (thêm đối tượng Employee), **showAll** (hiển thị toàn bộ thành viên).

```
class Employee:

    def __init__(self, name, job=None, pay=0):
        self.name = name
        self.job = job
        self.pay = pay

    def info(self):
        return self.name, self.job, self.pay

    def giveRaise(self, percent):
        self.pay = int(self.pay*(1+percent))
```

```
class Manager(Employee):

    def __init__(self, name, pay):
        Employee.__init__(self, name, 'mgr', pay)

    def giveRaise(self, percent, bonus=.10):
        Employee.giveRaise(self, percent+bonus)
```

```
class Team:

    def __init__(self, *args):
        self.members = list(args)

    def addMember(self, person):
        self.members.append(person)

    def showAll(self):
        for person in self.members:
            print(person.info())
```

HÀM SUPER()

- Trong class con, có thể dung hàm **super()** thay thế cho tên class cha.

```
class Manager(Employee):

    def __init__(self, name, pay):
        super().__init__(name, 'mgr', pay)

    def giveRaise(self, percent, bonus=.10):
        super().giveRaise(percent+bonus)
```

BÀI TẬP 3-20

- Xây dựng ***class*** **Ball**, lưu thông tin về 1 quả bóng hình cầu, có các thuộc tính **radius** (bán kính), **color** (màu sắc). Class có hàm tạo **__init__**, có các phương thức **info** (xem thông tin), **setColor** (đặt màu sắc), **surfaceArea** (tính diện tích bề mặt) **volume** (tính thể tích).

BÀI TẬP 3-21

- Xây dựng ***class*** **linReg**, mô tả hàm hồi quy tuyến tính đơn. Class có các thuộc tính **data** (dữ liệu), **intercept_** (hệ số tự do), **coef_** (hệ số của x), và các phương thức:
  - **__init__**: tạo đối tượng hồi quy cho bộ dữ liệu $\boldsymbol{data = \{(x_i; y_i)\}_{i=1,\dots,n}}$
  - **fit**: ước lượng các tham số của hàm hồi quy
  - **addPoint**: thêm điểm dữ liệu vào đối tượng hồi quy
  - **predict**: dự báo giá trị $\mathbf{y_0}$ tại $\mathbf{x_0}$.