# Ghi chú

# để ghi chú 1 dòng

“““

Để ghi chú 1 đoạn

”””

# Bố cục trong tập tin python

Không cần dấu ; cuối câu lệnh.

Trong python, tổ chức mã dựa trên THỤT LỀ, trái ngược với nhiều ngôn ngữ lập trình khác.

Chúng ta không thể sử dụng khoảng trắng 1 cách tự do (ngoài các dòng trống).

Các khối mã được tạo bằng cách sử dụng mức thụt lề chứ không phải dấu ngoặc nhọn {}.

Tips: Sử dụng Tab để thụt lề sang phải, và Shift + Tab để thụt lề sang trái.

# Cách lưu trữ

Có định dạng .py

# Cách xuất dữ liệu bằng câu lệnh print trong lập trình python

Cú pháp: print(object(s), sep=separator, end=end, file=file, flush=flush)

* Objects: đối tượng, dữ liệu được xuất
* Sep: ngăn cách
* End: kết thúc
* File: tên tập tin
* Flush: đẩy dữ liệu

# Cách nhập dữ liệu

Cú pháp: input(prompt)

* Prompt: chuỗi thông báo

# Biến, hằng số và từ khóa trong python

**BIẾN**

Một biến là một vị trí được đặt tên, nó được sử dụng để lưu trữ dữ liệu trong bộ nhớ. Giá trị của nó có thể thay đổi khi thực chạy chương trình.

Ví dụ: x=5

Dòng trên, chúng ta tạo ra 1 biến x có giá trị bằng 5

Ví dụ: y=10 #ban đầu y có giá trọ bằng 10

y=15 #y được thay đổi giá trị thành 15

Ví dụ: x,y,z = 1,2, “Xin chào” #gán giá trị cho nhiều biến

**HẰNG SỐ**

Hằng số là 1 loại biến có giá trị không thể thay đổi.

Trong python không thực sự có hằng số.

Hằng số thường được khai báo và gán trong một mô đun và người dùng hạn chế không thay đổi giá trị của nó. Ở đây, mô đun là một tệp mới chứa các biến, hàm, v.v. được nhập vào tệp chính. Bên trong mô đun, các hằng số được viết bằng tất cả các chữ cái in hoa và dấu gạch dưới chân ngăn cách các từ.

Ví dụ: PI =3,14

Ví dụ: import math

print(math.pi)

**CÁCH ĐẶT TÊN BIẾN VÀ HẰNG SỐ**

Chúng ta có thể sử dụng các chữ cái (a-z, A-Z), các con số (0-9), dấu gạch dưới \_ để đặt tên cho các biến và hằng số.

Ví dụ: content = “Lập trình python”

Tên biến nên có ý nghĩa và phù hợp với nội dung cần chứa

Nếu tên biến có nhiều từ hãy sử dụng dấu \_ để ngăn cách hoặc viết hoa các chữ cái của từng từ

Sử dụng chữ cái viết hoa (toàn bộ) để khai báo các hằng số

Không được bắt đầu bằng 1 chữ số (chương trình sẽ gặp lỗi)

Không sử dụng từ khóa làm tên biến

**CÁC TỪ KHÓA**

**A screen shot of a computer

Description automatically generated**

# Kiểu dữ liệu cơ bản

- Các kiểu dữ liệu cơ bản trong Python

Python là một ngôn ngữ thông dịch (không yêu cầu biên dịch), được đặc trưng bởi hệ thống kiểu động – bạn không bắt buộc phải khai báo kiểu của biến. Trình thông dịch tự đoán kiểu dữ liệu.

A white sheet with black text

Description automatically generated

- Ưu nhược điểm của kiểu dữ liệu động

Hệ thống kiểu dữ liệu động có cả 2 ưu điểm:

* Viết mã nhanh hơn
* Ít mã hơn

Và nhược điểm:

* Thời gian chạy lâu hơn
* Khả năng xảy ra lỗi khó gỡ lỗi

Python cho phép thay đổi kiểu dữ liệu của biến.

- Cách kiểm tra dữ liệu của biến

Sử dụng câu lệnh: type(tên\_biến)

- Ép kiểu (chuyển đổi kiểu dữ liệu) dữ liệu cơ bản

Quá trình chuyển đổi giá trị của một kiểu dữ liệu (số nguyên, chuỗi, số float, v.v.) sang kiểu dữ liệu khác được gọi là ép kiểu. Python có 2 kiểu chuyển đổi kiểu.

- Chuyển đổi kiểu ngầm định: Python tự động chuyển đổi một kiểu dữ liệu này sang kiểu dữ liệu khác. Quá trình này không cần bất kỳ sự tham gia nào của người dùng.

Ví dụ:

a = 5

b = 2.0

c = a/b

print(‘Kiểu dữ liệu của a:’ , type(a))

print(‘Kiểu dữ liệu của b:’ , type(b))

print(‘Kiểu dữ liệu của c:’ , type(c))

Chúng ta có thể thấy c có kiểu dữ liệu float vì Python luôn chuyển đổi kiểu dữ liệu nhỏ hơn sang kiểu dữ liệu lớn hơn để tránh mất dữ liệu.

- Chuyển đổi kiểu rõ ràng: do chúng ta thực hiện gõ lệnh chuyển đổi kiểu của một đối tượng thành kiểu dữ liệu bắt buộc. Chúng tôi sử dụng các hàm có sẵn int(), float(), str(), v.v. để thực hiện chuyển đổi

Cú pháp: ten\_kieu\_du\_lieu(biến)

# Các phép toán số học cơ bản trong lập trình Python

A white rectangular grid with black text

Description automatically generated with medium confidence 

# Toán tử trong Python

- So sánh

A screenshot of a computer

Description automatically generated

- Logic

A screenshot of a phone

Description automatically generated

- Toán tử gán

A screenshot of a math test

Description automatically generated A screenshot of a math test

Description automatically generated

- Toán tử bit

Toán tử bit hoạt động trên các toán hạng các chữ số nhị phân

A screenshot of a computer

Description automatically generated

- Toán tử đặc biệt

A screenshot of a phone

Description automatically generated

- Toán tử điều kiện (toán tử 3 ngôi)

[trả về khi dk đúng] if [điều kiện] else [trả về khi dk sai]

# Giới thiệu thư viện toán học math trong lập trình Python

- Một số hàm quan trọng

math.ceil(x): Trả về giá trị trần của x, số nguyên nhỏ nhất lớn hơn hoặc bằng x.

math.fabs(x): trả về giá trị tuyệt đối của x.

math.floor(x): trả về sàn của x, số nguyên lớn nhất nhỏ hơn hoặc bằng x.

math.exp(x): trả về e lũy thừa x, trong đó e = 2,718281… là có số của logarit tự nhiên.

math.log(x , base): với 1 đối số thì trả về logarit tư nhiên của x (cơ số e).

math.pow(x,y): trả về lũy thừa y.

- Một số giá trị constants

math.pi

math.e

# Câu lệnh rẽ nhánh if .. else trong Python

Câu lệnh if .. else là câu lệnh ra quyết định thực hiện một đoạn mã khi điều kiện đúng sai

A diagram of a flowchart

Description automatically generated

A diagram of a flowchart

Description automatically generated

A white background with green and blue text

Description automatically generatedA diagram of a flowchart

Description automatically generated

# Kiểu dữ liệu list trong Python

List (Danh sách): là 1 chuỗi các mục có thứ tự. Nó là 1 trong những kiểu dữ liệu được sử dụng nhiều nhất trong Python và rất linh hoạt. Tất cả các mục trong list không cần phải cùng loại.

Khai báo một danh sách khá đơn giản. Các mục được phân tách bằng dấu phẩy được đặt trong dấu ngoặc [].

# Vòng lặp for trong Python

# Vòng lặp while trong Python

Được sử dụng để lặp một khối mã khi biểu thức kiểm tra (điều kiện) còn đúng. Chúng ta thường sử dụng vòng lặp này khi chúng ta không biết trước số lần lặp lại.

A diagram of a test expression

Description automatically generated

A close-up of a sign

Description automatically generated

Vòng lặp while với else: Tương tự vòng lặp for, vòng lặp while cũng có thể có 1 câu lệnh else. Phần else được thực thi nếu điều kiện trong vòng lặp while được đánh giá là False. Vòng lặp while có thể được kết thúc bằng câu lệnh break. Trong những trường hợp này, phần else bị bỏ qua. Do đó, một phần else của vòng lặp while sẽ chạy nếu không có câu lệnh break nào xảy ra khi điều kiện là sai.

# Cách sử dụng break và continue trong vòng lặp

Câu lệnh ngắt break Python: câu lệnh ngắt break kết thúc vòng lặp chứa nó. Điều khiển chương trình chuyển đến câu lệnh ngay sau phần thần của vòng lặp.

Nếu câu lệnh break nằm trong một vòng lặp lồng nhau, câu lệnh break sẽ kết thúc vòng lặp trong cùng.

A computer code with black text

Description automatically generated with medium confidence

Câu lệnh continue Python: được sử dụng để bỏ qua phần còn lại bền trong vòng lặp của lần lặp lại hiện tại. Vòng lặp không kết thúc nhưng tiếp tục với lần lặp tiếp theo

A screenshot of a computer code

Description automatically generated

# Kiểu dữ liệu Tuple trong Python

Tuple là một chuỗi các phần tử có thứ tự giống như 1 list. Sự khác biệt duy nhất là bộ giá trị là các hằng số. Tuples một khi được tạo ra thì giá trị của nó không thể sửa đổi.

Tuples được sử dụng để bảo vệ dữ liệu và thường nhanh hơn list vì chúng không thể thay đổi động.

Được định nghĩa trong dấu ngoặc đơn (), các mục được phân tách bằng dấu phẩy.

Giá trị của Tuple có thể bị trùng lặp.

Các thao tác với Tuple:

- Chúng ta có thể truy cập nhiều mục bằng cách tham khảo số chỉ mục, bên trong dấu ngoặc vuông.

# Kiểu dữ liệu Set trong Python

Set là một trong 4 kiểu dữ liệu tích hợp sẵn trong Python dùng để lưu trữ các tập hợp dữ liệu, 3 kiểu còn lại là List [], Tuple (), Dictionary {}, tất cả đều có chất lượng và cách sử dụng khác nhau.

Set là tập hợp không có thứ tự, giá trị không được trùng nhau, không thể thay đổi \* và không được lập chỉ mục. Lưu ý: Các mục Set là không thể thay đổi, nhưng bạn có thể xoá các mục và thêm các mục mới.

Sử dụng cặp ngoặc { }.

# Kiểu dữ liệu Dictionary trong Python

Từ điển được sử dụng để lưu trữ các giá trị dữ liệu trong các cặp key: value.

Từ điển là một tập hợp được sắp xếp theo thứ tự\*, có thể thay đổi và không cho phép trùng lặp.

Từ điển được viết bằng dấu ngoặc nhọn {} và có các khoá và giá trị.

Từ điển có thể thay đổi, nghĩa là chúng ta có thể thay đổi, thêm bớt các mục sau khi từ điển đã được tạo.

Từ điển không thể có 2 mục cùng với 1 khoá.

# Xây dựng function trong Python

Định nghĩa:

- Hàm là một khối mã chỉ chạy khi nó được gọi.

- Bạn có thể truyền dữ liệu, được gọi là tham số, vào một hàm.

- Kết quả là một hàm có thể trả về dữ liệu.

Khai báo hàm:

- Trong Python, một hàm được định nghĩa bằng từ khoá def:

Ví dụ: def xinChao():

Print(“Xin chào”)

Gọi hàm:

Để gọi một hàm, hãy sử dụng tên hàm theo sau bởi dấu ngoặc đơn

Ví dụ: xinChao()

Đối số (Arguments):

Hàm có thể nhận vào các tham số:

def xinChao(hoVaTen):

    print("Xin chào"+ hoVaTen)

Một hàm có thể nhận nhiều đối số:

def xinChao(ho, tenLot, ten):

    print("Xin chào "+ ho + tenLot + ten)

Nếu chúng ta không biết có bao nhiêu đối số sẽ được truyền, chúng ta thêm một \* trước tên tham số trong định nghĩa hàm. Hàm sẽ nhận được nhiều đối số và có thể truy cập các mục tương ứng.

def thoiKhoaBieu(\*monHoc):

    print("Môn 1: " + monHoc[0])

    print("Môn 2: " + monHoc[1])

thoiKhoaBieu("Toán", "Lý", "Hoá", "Sinh", "Sử", "Địa", "Anh")

def total(\*value):

    sum = 0

    for x in value:

        sum += x

    print(sum)

total(1,2,3,4,5,6,7,8,9,10)

Sử dụng từ khoá cho nhiều đối số, thêm dấu \*\* vào trước tên tham số:

def xinChao(\*\*hoVaTen):

    print("Xin Chào:"+ hoVaTen["ten"])

xinChao(ten = "Nhân", tenLot = "Ngọc", ho = "Trần")

# Lambda function trong Python

Một hàm lambda là một hàm ẩn danh nhỏ.

Một hàm lambda có thể nhận bất kỳ số lượng đối số nào, nhưng chỉ có một biểu thức.

Cú pháp: lambda arguments : expression

Ví dụ: kiemTraSoChan = lambda a : (a%2==0)

print(kiemTraSoChan(5))

**Lập trình hướng đối tượng**

A drawing of a person and writing

Description automatically generated

A close-up of a white board

Description automatically generated

A close-up of some writing

Description automatically generatedA diagram of different types of words

Description automatically generated

## **UML là gì?**

**Ngôn ngữ mô hình hóa thống nhất** (tiếng Anh: Unified Modeling Language, viết tắt thành **UML**) là một ngôn ngữ mô hình gồm các ký hiệu đồ họa mà các phương pháp hướng đối tượng sử dụng để thiết kế các hệ thống thông tin một cách nhanh chóng.

Cách xây dựng các **mô hình trong UML** phù hợp mô tả các hệ thống thông tin cả về cấu trúc cũng như hoạt động. Cách tiếp cận theo **mô hình của UML** giúp ích rất nhiều cho những người thiết kế và thực hiện hệ thống thông tin cũng như những người sử dụng nó; tạo nên một cái nhìn bao quát và đầy đủ về hệ thống thông tin dự định xây dựng. Cách nhìn bao quát này giúp nắm bắt trọn vẹn các yêu cầu của người dùng; phục vụ từ giai đoạn phân tích đến việc thiết kế, thẩm định và kiểm tra sản phẩm ứng dụng công nghệ thông tin. Các mô hình hướng đối tượng được lập cũng là cơ sở cho việc ứng dụng các chương trình tự động sinh mã trong các ngôn ngữ lập trình hướng đối tượng, chẳng hạn như ngôn ngữ C++, Java,... Phương pháp mô hình này rất hữu dụng trong [lập trình hướng đối tượng](https://hoclaptrinh.vn/posts/lap-trinh-huong-doi-tuong-oop-la-gi). Các mô hình được sử dụng bao gồm Mô hình đối tượng (mô hình tĩnh) và Mô hình động.

**UML sử dụng một hệ thống ký hiệu thống nhất** biểu diễn các Phần tử mô hình (model elements). Tập hợp các phần tử mô hình tạo thành các Sơ đồ UML (UML diagrams). Có các loại sơ đồ UML chủ yếu sau:

* Sơ đồ lớp (Class Diagram)
* Sơ đồ đối tượng (Object Diagram)
* Sơ đồ tình huống sử dụng (Use Cases Diagram)
* Sơ đồ trình tự (Sequence Diagram)
* Sơ đồ cộng tác (Collaboration Diagram hay là Composite Structure Diagram)
* Sơ đồ trạng thái (State Machine Diagram)
* Sơ đồ thành phần (Component Diagram)
* Sơ đồ hoạt động (Activity Diagram)
* Sơ đồ triển khai (Deployment Diagram)
* Sơ đồ gói (Package Diagram)
* Sơ đồ liên lạc (Communication Diagram)
* Sơ đồ tương tác (Interaction Overview Diagram - UML 2.0)
* Sơ đồ phối hợp thời gian (Timing Diagram - UML 2.0)

A whiteboard with a list of tasks

Description automatically generated

A whiteboard with writing on it

Description automatically generated

A close-up of a text

Description automatically generated

A diagram of a string

Description automatically generated

**Lớp, đối tượng, thuộc tính và phương thức**

1. Object
2. Class
3. Method
4. Inheritance (kế thừa)
5. Polymorphism (đa hình)
6. Encapsulation (đóng gói)
7. Data abstraction (trừu tượng)

**Lớp / Đối tượng**

Class: một phạm trù đối tượng. Lớp định nghĩa tất cả các thuộc tính chung (thuộc tính và phương thức) của các đối tượng khác nhau thuộc về nó. Các đối tượng, được tạo dựa trên định nghĩa của các lớp cụ thể là một trong những tính năng chính của lập trình hướng đối tượng.

Dưới đây là một ví dụ về một lớp đơn giản, giúp bạn hiểu sự khác biệt giữa thuộc tính lớp và thuộc tính phần tử (trường hợp đơn lẻ của lớp).

#Ví dụ về tạo class đơn giản

class SimpleClass:

    #Attribute (thuộc tính)

    i=5

    #\_init\_

    def \_\_init\_\_(self):

        #Attribute of an instance of class

        self.j = 7

    #Method

    def printMe(self):

        print(self.j)

objectA = SimpleClass()

objectB = SimpleClass()

objectA.printMe()           #.printMe gọi phương thức (method) bên trong đối tượng

print(objectB.i)

#Thay đổi giá trị của thuộc tính

objectA.i = 100

objectB.j = 500

print(objectA.i)

objectB.printMe()

Static method:

class SimpleClass2():

    #Constructor

    def \_\_init\_\_(self) -> None:

        self.name = "Nhân"

    #Method

    def hello(self):

        print("Hello "+self.name)

    #Static methods

    @staticmethod

    def hi(name):

        print("Hi "+name)

objectC = SimpleClass2()

objectD = SimpleClass2()

objectC.hello()

objectD.hi("Peter")

SimpleClass2.hi("Nam")

**Kế thừa trong Python**

Mọi ngôn ngữ lập trình hướng đối tượng sẽ không đáng để xem hoặc sử dụng, nếu nó không hỗ trợ tính kế thừa.

Python hỗ trợ tính kế thừa.

Các lớp có thể kế thừa từ các lớp khác.

Một lớp có thể kế thừa các thuộc tính và phương thức hành vi từ một lớp khác, được gọi là lớp cha (lớp cơ sở).

Lớp kế thừa từ lớp cha được gọi là lớp con (lớp thừa kế).

A diagram of a diagram

Description automatically generated with medium confidence

# Try and Except – Bắt lỗi trong Python

try:

#Đoạn code dự đoán có lỗi

except:

#Hành động khi lỗi xảy ra

else:

#Thực thi đoạn này nếu như mã không có lỗi

finally:

#Cho phép bạn thực thi mã, bất kể kết quả của các khối try có bị lỗi hay không

# Làm việc với File trong Python

#open()  
# "x" - tạo file  
*"""  
try:  
 f = open("vidu1.txt", "x")  
except Exception as e:  
 print(e)  
"""*#"w" - ghi dữ liệu vào file  
#"a" - nối vào file  
try:  
 with open("vidu1.txt", "w") as f:  
 f.write("Hello")  
 f.close()  
except Exception as e:  
 print(e)  
  
try:  
 with open("vidu1.txt", "a") as f:  
 f.write("Hello")  
 f.close()  
except Exception as e:  
 print(e)  
  
#"r" - đọc file  
try:  
 with open("vidu1.txt", "r") as f:  
 noidung = f.read()  
 print(noidung)  
except Exception as e:  
 print(e)  
  
#encoding = utf -8  
f =open('sample-file.txt', encoding='utf-8')  
f = open('a-new-flie.txt',mode='w',encoding='utf-8')