**Lập trình PYTHON**

**(BASIC)**

* Python là gì?
* Python là một ngôn ngữ lập trình phổ biến. Nó được tạo ra bới **Guido van Rossum**  tạo ra năm 1990 và được phát hành năm 1991.
* Nó được sử dụng cho:

+ phát triển web(phía máy chủ)

+ phát triển phần mềm

+ toán học

+ kịch bản hệ thống

* Python có thể làm gì?
* Python có thể được sử dụng trên máy chủ để tạo các ứng dụng web.
* Python có thể được sử dụng cùng với phần mềm để tạo quy trình công việc.
* Python có thể kết nối với các hệ thống cơ sở dữ liệu. Nó cũng có thể đọc và sửa đổi các tập tin.
* Python có thể được sử dụgn để xử lý dữ liệu lớn và thực hiện các phép toàn phức tạp.
* Python có thể được sử dụng để tạo mẫu nhanh hoặc phát triển phần mềm sẵn sàng sản xuất.
* Tại sau lại là python?
* Python hoạt động trên các nền tảng khác nhau(windows, mac, linux, raspberry pi..).
* Python có một cú pháp đơn giản tương tự như ngôn ngữ tiếng anh.
* Python có cú pháp cho phép các nhà phát triển viết chương trình với ít dòng hơn so với một ngôn ngữ lập trình khác.
* Python chạy trên một hệ thống thông dịch, có nghĩa là mã có thể có rất nhanh chóng.
* Python có thể được xử lý theo một cách thủ tục, một cách hướng đối tượng hoặc một cách chức năng.

**Phần 1: Chương trình cơ bản**

**Bài 1: Sơ lược về python**

+ Ưu điểm của python:

* Cú pháp tường minh, dễ đọc.
* Các khả năng tự xét mạnh.
* Hướng đối tượng trực giác.
* Cách thể hiện tự nhiên mã thủ tục.
* Hoàn toàn mô-đun hóa, hỗ trợ các gói theo cấp bậc.
* Xử lý lỗi dựa theo ngoại lệ.
* Kiểu dữ liệu động ở mức rất cao.
* Các thư viện chuẩn và các mô-đun ngoài bao quát hầu như mọi việc.
* Phần mở rộng và mô-đun dễ viết trong C, C++.
* Có thể nhúng trong ứng dụng như một giao diện kịch bản.
* Python mạnh mẽ và thực hiện rất nhanh.
* **Python được thiết kế với mục đích đơn giản, cú pháp ngắn gọn**

**1, Ghi chú trong python.**

Ghi chú là cách mỗi lập trình viên thường làm mỗi khi viết các đoạn code nó có tác dụng note lại nhưng tác dụng, lưu ý, cách dùng, ... và để đánh dấu.

+ Để (comment) chú thích cho dòng lệnh có hai cách

* Chú thích một dòng: ta dùng ký tự #
* Chú thích nhiều dòng: ta dùng 3 dấu ‘’’ hoặc “””

Ví dụ:

**2, Hàm print trong python.**

**+** print: có tác dụng hiển thị dữ liệu ra màn hình khi chương trình thực thi.

Cú pháp:



Trong đó: content là nội dung hay biến mà mình muốn hiển thị ra màn hình, nếu muốn hiển thị nhiều nội dung khác nhau trên cùng một lần print thì chúng ta chỉ cần ngăn các giữa các nội dung bằng dấy phẩy.

Ví dụ:

Shape

Description automatically generated with medium confidence

* **Thay đổi ngắt dòng print.**

+ Mặc định mỗi lần chúng ta print dữ liệu thì nó sẽ tự ngắt dòng cho dòng tiếp theo.

Ví dụ:

Shape

Description automatically generated with medium confidence

Vậy nó chúng ta không muốn nó tự động ngắt dòng mỗi khi kết thúc print nữa thì ta sử dụng keyword end thêm vào cuối print.

Ví dụ:

A screenshot of a computer

Description automatically generated with medium confidence

* Toán tử + dùng để nối chuỗi.
* Toán tử \* dùng để lặp chuỗi.

Shape

Description automatically generated with low confidence

Ta thấy để nối 2 chuỗi có kiểu dữ liệu khác nhau ta phải chuyển về cùng dạng chuỗi.

**Bài 2: Kiểu dữ liệu biến và ép kiểu**

**1, khai báo biến trong python.**

+ Biến là vùng chứa để lưu trữ các giá trị dữ liệu.

+ Biến được sử dụng để lưu trữ dữ liệu trong chương trình, dữ liệu được lưu trữ trong một biến có thể thay đổi trong khi chương trình chạy.

+ Trong python thì biến không cần khai báo tường minh tức là định nghĩa kiểu dữ liệu trước khi sử dụng, biến được tạo khi ta bắt đầu gán giá trị cho nó.

Vd: khai báo biến và gán trị cho a = 5



Có thể khai báo nhiều biến cùng một giá trị trên một dòng.



Hay là khai báo nhiều biến với các giá trị tương ứng của nó trên 1 dòng, phải đảm bảo số lượng biến khớp với số lượng giá trị.



Có thể lấy các giá trị biến từ trong list.

Shape

Description automatically generated with medium confidence

**2, Một số kiểu dữ liệu trong python**.

+ python cũng giống như một số ngôn ngữ khác, khi ta khai báo biển thì kiểu dữ liệu của nó sẽ được tự động phát hiện.

A picture containing text, screen, screenshot

Description automatically generated

**3, Kiểm tra kiểu dữ liệu.**

+ Trong python để kiểm tra kiểu dữ liệu của một biến chúng ta có thể sử dụng hàm type

Cú pháp:



Ví dụ:

Shape

Description automatically generated with low confidence

**Một số lưu ý khi đặt tên biến**

* Tên biến *có thể* được bắt đầu bằng ký tự \_ hoặc bất cứ chữ cái nào (in hoa hoặc in thường).
* Tên biến ***không*** được bắt đầu bằng chữ số. Ví dụ tên biến 100invalid là không hợp lệ.
* Tên biến ***không*** được bao gồm dấu cách trong đó.
* ***Không***sử dụng các từ khóa có sẵn trong Python như print, if, else... để đặt tên cho biến. Ví dụ tên biến print là không hợp lệ, tên biến print1, print\_2, printHello... là hợp lệ.

**4, Ép kiểu dữ liệu trong python.**

+ Sử dụng các hàm:

Graphical user interface, text, application

Description automatically generated

+ Ngoài kiểu dữ liệu số thì kiểu ký tự (str) cũng có các toán tử như +, \*, [], in, not in, ...

* Toán tử + dùng để nối chuỗi.
* Toán tử \* dùng để lặp chuỗi.

Ví dụ:

Shape

Description automatically generated with medium confidence

*Ta chỉ có thể nối hai chuỗi với nhau chứ không thể nối một chuỗi với 1 số.*

Ví dụ:

Text

Description automatically generated with medium confidence

**Để nối một chuỗi với một số bạn cần phải đưa số này về dạng chuỗi, trong python dùng hàm “str”.**

**Ví dụ: Khi ép kiểu thì biến đó sẽ có kiểu dữ liệu được ép.**

**Shape

Description automatically generated with medium confidence**

**Bài 3: Nhập xuất trong python**

**1, Để nhập dữ liệu cho biến từ bàn phím thì ta sử dụng hàm “input()”**

Ví dụ: Nhập tên từ bàn phím.

Shape

Description automatically generated with medium confidence

* ***Lưu ý: Khi nhập dữ liệu một biến từ bàn phím thì kiểu dữ liệu của biến đó luôn là “str” kể cả khi nhập biến đó giá trị số nguyên thì biến đó vẫn có kiểu dữ liệu là “str”.***

Shape

Description automatically generated with medium confidence

**2, Ép kiểu khi nhập.**

* + Khi nhập dữ liều vào từ bàn phím thì kiểu dữ liệu luôn là “str” (hàm input() có kiểu str), nếu muốn tính toán thì ta phải đưa về kiểu số bằng cách ép kiểu.

Ví dụ:

Shape

Description automatically generated with low confidence

* Biến age đã được ép kiểu từ ‘str’ thành ‘int’.
* Ép thành các kiểu dữ liệu khác cũng tương tự.

**3, phạm vi biến.**

a, Biến toàn cục

* Biến được khai báo bên ngoài hàm hoặc trong phạm vi toàn cục được gọi là biến toàn cục. Biến có thể xử dụng ở bất kì đâu trong chương trình kể cả trong hàm.
* Biến toàn cục có tên gọi là Global Variable.
* Khi sử dụng biến toàn cục bên trong một hàm thì ta không được thay đổi giá trị của biến.

Ví dụ: Để thay đổi giá trị biến ta sử dụng từ khóa global.

Shape

Description automatically generated with medium confidence

Từ khóa global dùng để khai báo biến toàn cục trong hàm và nếu muốn thay đổi giá trị của biến toàn cục.

b, Biến cục bộ

* Biến cục bộ là biến được khai báo bên trong hàm hoặc phạm vi cục bộ
* Biến cục bộ có tên là local Variable.
* Nếu khai báo biến trong hàm có cùng tên biến toàn cục, thì biến này sẽ là biến cục bộ và chỉ có thể sử dụng trong hàm, không liên quan gì biến toàn cục.

A picture containing shape

Description automatically generated

**Bài 4: Các kiểu dữ liệu số và toán tử.**

* **Các kiểu dữ liệu**

**Text

Description automatically generated with medium confidence**

* **Kiểu dữ liệu số.**

+ Một biến được khai báo là kiểu dữ liệu number trong python thì xét về mặt lưu trữ bộ nhớ thì nó sẽ không thay đổi được giá trị, mà khi ta gán giá trị mới cho nó thì thực chất nó sẽ tạo ra các ô nhớ mới khác để lưu trữ giá trị mới đó.

+ Trong python hỗ trợ 3 kiểu dữ liệu dạng number như sau:

* Int: kiểu số nguyên kiểu này có kích thước không giới hạn.
* Float: kiểu số thực, kiểu ngày ngoài kiểu viết bình thường ra thì nó cũng có thể hiện thị dưới dạng số mũ E(2.5e2 = 250). “ e, x: tương đương cới 10 \* x số 10.”
* Complex: kiểu số phức đây là kiểu dữ liệu hiếm khi sử dụng tới.

+ Để giải phóng một hoặc nhiều vùng nhớ cho biến trong python thì ta có thể sử dụng lệnh del:

Ví dụ:

Shape

Description automatically generated with medium confidence

Các toán tử đã biết là “ =, +, -, \*, /, %, ... ”

* **Python xây dựng 7 loài hình tính toán trên toán tử:**

**- Python Arithmetic Operator (toán tử số học)**

**- Python Relational Operator (toán tử quan hệ)**

**- Python Assignment Operator (toán tử gán)**

**- Python Membership Operator (toán tử membership)**

**- Python Identity Operator (toán tử identity)**

**- Python Logical Operator (toán tử logic)**

**- Python Bitwise Operator (toán tử thao tác bit)**

**1, Toán tử số học Arithmetic Operators**

+ Đây là toán tử dùng để thao tác với các phép toán như cộng, trừ, nhân, chia, chia dư ...

Table

Description automatically generated

**2, Toán tử quan hệ Relational Operators**

* **Các toán tử so sánh được dùng để so sánh hai giá trị với nhau.**
* Nếu đúng thì trả về **True**, sai thì trả về **False.**

**Graphical user interface, text, application, email

Description automatically generated**

**3, Toán tử gán Assignment Operators**

* Toán tử gán dùng để gán giá trị của vế phải sang cho vế trái.

Table

Description automatically generated

**4, Toán tử Python Membership Operators**

* **Toán tử membership là toán tử dùng để kiểm xem một giá trị có nằm trong giá trị khác hay không.**

**+ in và not in** là hai toán tử membership với cách sử dụng rất đơn giản.

**Ví dụ:** Kiểm tra một sau có nằm trong sâu khác hay không

**A picture containing text

Description automatically generated**

**5, Toán tử định danh Identity Operators**

* **Toán tử định danh identity:** được dùng để xác định xem hai biến có đang trỏ tới cùng một đối tượng hay không. Với các kiểu dữ liệu như int, str, float ... Thì toán tử này tương đương với toán tử so sánh(toán tử quan hệ) “==”.
  + ***Khác “==” ở chỗ...***
* **Trong python,** “is và is not “ là 2 toán tử định danh.

**Shape

Description automatically generated with medium confidence**

**6, Toán tử logic**

* **Toán tử logical : not, or và and:** là các toán tử được dùng để kết hợp các mệnh đề lại với nhau.

Table

Description automatically generated

Ví dụ: Với Python

Shape

Description automatically generated with medium confidence

**Bài 5: Câu lệnh điều kiện(lựa chọn) trong python**

**1, Mệnh đề if – else.**

* **Mệnh đề if – else:**  được dùng để kiểm tra một điều kiện có đúng hay không, nếu đúng thì khối lệnh trong if sẽ được thực thi, ngược lại nếu điều kiện sai thì khối lệnh bên trong else được thực thi:

A picture containing text, clock, device

Description automatically generated

***Lưu ý:*** *khối lệnh trong if và else chính là các đoạn code được thục lè vào trong( ở các ngôn ngữ lập trình khác thì khối lệnh được xác định bằng cặp dấu {} nhưng đối với Python thì khối lệnh được xác định bằng việc thụt lề).*

* **Toán tử ba ngôi trong Python**

**+ Toán tử ba gôi:** cũng được dùng để kiểm tra các điều kiện giống như mệnh đề if – else nhưng nó đơn giản hơn ở chỗ có thể viết gọn thành một dòng. **Toán tử ba ngôi** có thể thay thế cho mệnh đề if – else khi cần kiểm tra các điều kiện đơn giản.

**(C++:** condition ? true\_value : false\_value; **)**

**Python:** Toán tử ba ngôi lồng nhau



Ví dụ:

Text

Description automatically generated

* ***Có thể kết hợp toán tử quan hệ vào trong điều kiện của mệnh đề if – else***

Shape

Description automatically generated with medium confidence

**Bài 6: Vòng lặp trong Python**

* **Vòng lặp:**  Đơn giản là để lặp lại một đoạn code nhiều lần. Trong Python hỗ trợ 2 loại vòng lặp là **while** và vòng lặp **for**.

**1, Vòng lặp while trong python.**

Vòng lặp while được dùng để lặp lại một hành động cho tới khi điều kiện lặp không còn thỏa mãn nữa, không biết rõ điều kiện bao giờ dừng.

Cú pháp:



Ví dụ: Chương trình hiển thi các số từ 1 tới 4 dùng vòng lặp while:

Shape

Description automatically generated with medium confidence

**2, Vòng lặp for trong python.**

+ Vòng lặp for ở trong python có tác dụng lặp các biến dữ liệu có trong list, Tuple hoặc string hay còn gọi là một tập hợp cho trước.

Cú pháp:



Với data là một list, Tuple, chuỗi hoặc một khoảng cho trước... chứa giá trị cần lặp.

Ví dụ:

A picture containing text

Description automatically generated

* **Hàm sử dụng trong vòng lặp: range(a, z).**
* Để tạo ra các con số từ a -> z nó có nghĩ là từ a tới nhỏ hơn z 1 đơn vị.

Ví dụ:



* **List Comprehension(for rút gọn).**

+ list comprehension cung cấp một cú pháp ngắn hơn khi sử dụng for để tạo, hoặc hiện thì các phần tử hiện có.

Cú pháp:



Ví dụ:

Shape

Description automatically generated with medium confidence

Còn rất nhiều cách sử dụng...

Ví dụ: chuyển đổi các phần tử thành in hoa

A screenshot of a computer

Description automatically generated with medium confidence

Ví dụ: Gán các phần tử của list bằng 1 giá trị mới.

Shape

Description automatically generated with medium confidence

**3, Thoát vòng lặp bằng lệnh Break.**

* **Câu lệnh break:** được dùng để thoát khỏi vòng lặp, khi chương trình gặp câu lệnh break thì vòng lặp sẽ dừng và thoát khỏi vòng lặp.

Ví dụ:

A picture containing background pattern

Description automatically generated

Có thể thấy chương trình đã hiển thị ra các số từ 1 tới 2 thay vì từ một tới 4 vì: khi i == 3 thì chương trình gặp lệnh break và vòng lặp bị thoát.

**4, Câu lệnh continue**

* Khi gặp câu lệnh continue trong vòng lặp, các đoạn code bên dưới câu lệnh này trong vòng lặp sẽ bị bỏ qua và không thực thi và chuyển tới lần lặp tiếp theo.

Ví dụ:

Shape

Description automatically generated with medium confidence

Ta thấy, các giá trị của i khi chia lấy dư cho 2 khác không (số lẻ) khi gặp lệnh continue thì nó bỏ qua các lệnh sau nó trong vòng lặp mà chuyển tới vòng lặp tiếp theo, nên các giá trị lẻ bị bỏ qua còn các giá trị chẵn.

**+ Để làm tròn số thực ta dùng hàm “round(number, ndigits)”**: ndgits là số chữ số sau dấu phẩy cần làm tròn.

Ví dụ:

Shape

Description automatically generated with medium confidence

**Bài 7: List(mảng) trong python**

**1, List(mảng).**

**+ list** trong python là một kiểu dữ liệu cho phép lưu trữ nhiều kiểu dữ liệu khác nhau, chúng ta có thể truy xuất đến các phần tử bên trong nó thông qua vị trí của phần tử đó trong list.

+ Để khởi tạo một **list** trong python ta có thể sử dụng cặp dấu [].

Ví dụ:



* **Để truy cập đến các giá trị trong list.**
* Tương tự chuỗi.
* Các phần tử list được đánh dấu bắt đầu từ 0 theo chiều từ trái sang phải và từ -1 theo chiều từ phải qua.

Ví dụ:

**Shape

Description automatically generated with medium confidence**

***Lưu ý:*** *name[1] không phải là phần tử đầu tiên của list mà phải là name[0] vì chỉ số của list bắt đầu từ 0.*

+ Trường hợp phải in ra một phần của list(sciling) thì ta sử dụng cú pháp:



* Nếu để trống cả thì tương đương với đầu và cuối.

Ví dụ:

A picture containing shape

Description automatically generated

+ Vòng lặp for có thể lặp qua các phần tử của một tập hợp nên for thường được dùng để lặp qua các phần tử của list.

Shape

Description automatically generated with medium confidence

* **Thêm sửa xóa phần tử trong list.**

**+ Để thêm một phần tử vào cuối list** ta dùng hàm append():

Shape

Description automatically generated with medium confidence

Ta có thể dùng còng for và while để thêm các phần tử cho list.

A picture containing text

Description automatically generated

**+ Để sửa phần tử trong list.**

* Ta chỉ cần truy cập vào phần tử trong list(chỉ định hoặc sciling) và gán lại giá trị có nó hoặc xóa.

Ví dụ:

Shape

Description automatically generated with medium confidence

Nếu số lượng chèn vào không bằng số lượng mục thay thế thì số lượng phần tử danh sách sẽ thay đổi.

Shape

Description automatically generated with medium confidence

**+ Để xóa phần tử trong list.**

* Để xóa phần tử trong list ta cần truy cập tới phần tử đó và dùng hàm del để xóa.
* Sau khi chúng ta xóa phần tử trong list thì index của list sẽ được cập nhập lại.
* **list lồng nhau.**

Do list có thể chứa nhiều kiểu dữ liệu khác nhau nên chúng ta hoàn toàn có thể khai báo một list chứa một hoặc nhiều list khác nhau.

A picture containing text

Description automatically generated

Ta truy cập các phần tử của list lồng nhau như bình thường và theo cấp từ ngoài vào trong.

Ví dụ:

Shape

Description automatically generated with medium confidence

**2, Các hàm xử lý trong python.**

* **Các hàm thường được sử dụng trong list**

**2.1, list().**

* Hàm này dùng để chuyển đổi kiểu dữ liệu của một biến sang dạng list

Cú pháp:

**list(data)** Với data là tên của list cần chuyển đổi.

Ví dụ:

A picture containing background pattern

Description automatically generated

**2.2, len().**

- Hàm này trả về số lượng phần tử có trong **list.**

Cú pháp:

**len(list)** Với list là tên của list.

Ví dụ: Đếm số phần tử của mảng

Shape

Description automatically generated with medium confidence

**2.3, max().**

**+** Hàm này sẽ trả về phần tử có giá trị lớn nhất trong list.

- Nếu chuỗi thì nó sẽ trả về phần tử có độ dài chuỗi dài nhất.

- Nếu là số thì nó sẽ trả về phần tử có số lớn nhất.

Cú pháp:

**max(list)** Với list là list cần kiểm tra.

Ví dụ:

Shape

Description automatically generated with low confidence

**2.4, min().**

**+** Hàm này sẽ trả về phần tử có giá trị nhỏ nhất trong list.

* Nếu chuỗi thì nó sẽ trả về phần tử có độ dài chuỗi ngắn nhất.
* Nếu là số thì nó sẽ trả về phần tử có số nhỏ nhất.

Cú pháp

**min(list)** Với list là list cần kiểm tra

Ví dụ:

A screenshot of a computer

Description automatically generated with medium confidence

**2.5, append().**

**+** Phương thức này có tác dụng thêm phần tử vào cuối của một list.

Cú pháp:

**Tênlist.append(giatri)**

* Nếu giá trị đó nhập từ bàn phím thì
  + Tênlist.append(int(input())

Ví dụ:

Text

Description automatically generated with medium confidence

**2.6, extend().**

+ Hàm này có tác dụng kế thừa lại các phần tử của list2 và thêm vào list1.

*-> Nó có tác dụng như nối list này với list kia.*

Ví dụ:

Shape

Description automatically generated with medium confidence

Ta có thể làm cách khác như sử dụng toán tử + để nối 2 list lại với nhau.

**2.7, count().**

+ Phương thức này có tác dụng đếm số lần xuất hiện của một thành phần trong list.

Cú pháp:

**Tênlist.count(value)**  Với value là phần tử muốn đếm trong list.

Vd:

Shape

Description automatically generated with medium confidence

**2.8, index().**

+ Phương thức này có tác dụng trả về index (chỉ số) xuất hiện đầu tiên của phần tử mà bạn muốn tìm và nếu không như tìm thấy thì nó sẽ gọi exception.

Cú pháp:

**Tênlist.index(value)**  Với value là phần tử muốn tìm trong list.

Ví dụ:

Shape

Description automatically generated with medium confidence

**2.9, insert().**

+ Phương thức có tác dụng thêm phần tử vào vị trí index của list, và các phần tử sau index đó sẽ được đẩy về phía sau.

Cú pháp:

**Tênlist.insert(index, value)** Với value là giá trị của phần tử được thêm vào.

Ví dụ:

Shape

Description automatically generated with medium confidence

**2.10, reverse().**

+ phương thức này có tác dụng đảo ngược vị trí của các phần tử trong list.

Cú pháp:

**Tênlist.reverse()**

Ví dụ:

Shape

Description automatically generated with medium confidence

**2.11, remove().**

*+* Phương thức này có tác dụng xóa phần tử khỏi list.

Cú pháp:

**Tênlist.remove(value)**  Với value là phần tử mà ta muốn xóa khỏi list.

Ví dụ:

Shape

Description automatically generated with medium confidence

**2.12, pop().**

+ Phương thức này có tác dụng xóa bỏ phần tử trong list dựa vào index(chỉ số) của nó.

Cú pháp:

**Tênlist.pop(index)**  Với index là vị trí của phần tử cần xóa.

***Nếu không điền index thì mặc định ta có: index vị trí của phần tử cuối của list.***

Ví dụ:

Shape

Description automatically generated with low confidence

**2.13, sort().**

+ Phương thức này có tác dụng sắp xếp lại các phần tử trong list theo một thứ tự xác định.

Cú pháp:

**Tênlist.sort(reverse, key)**

* **reverse**  là một boolean cấu hình kiểu sắp xếp. Nếu **reverse = True** thì list sẽ được sắp xếp từ lớn đến bé, nếu **reverse = False** thì list sẽ được sắp xếp từ bé đến lớn.
* **Mặc định reverse = False.** *( list sẽ được sắp xếp từ bé đến lớn )*
* **Key**  là **callback def** đề xử lý list hoặc là một **lamda funtion** (thường được dùng để sắp xếp các list **tuple** hoặc **dictionary.**

Ví dụ: Sắp xếp các phần tử của list tăng hoặc giảm.

**Shape

Description automatically generated with medium confidence**

**+ sắp xếp tùy chỉnh.**

* Ta có thể tùy chỉnh chức năng riêng của mình bằng cách sử dụng đối số tử khóa. Key = function.
* Hàm sẽ trả về một số được sử dụng để sắp xếp danh sách(số đầu là số thấp nhất đầu tiên).

Ví dụ: Sắp xếp danh sách dựa trên mức độ gần của số với 50:

A picture containing shape

Description automatically generated

* Mặc định, sort() phương thức này có phân biệt chữ hoa chữ thường, nên các chữ hoa sẽ được xếp trước các chữ cái thường được viết sau:

+ Chúng ta giải quyết các vấn đề này bằng cách sử dụng hàm tích hợp làm các hàm chính khi sắp xếp danh sách.

Vì vậy, nếu bạn muốn một hàm sắp xếp không phân biệt chữ hoa chữ thường, hãy sử dụng hàm str.lower làm hàm chính:

Ví dụ:

A screenshot of a computer

Description automatically generated with medium confidence

**2.14, clear().**

+ Phương thức này có tác dụng xóa bỏ hết tất cả các phần từ trong list.

Cú pháp:

**Tênlist.clear()**

Ví dụ:



**2.15, copy().**

+ phương thức này có tác dụng sao chép list có sẵn cho list mới.

Ví dụ:

Shape

Description automatically generated with medium confidence

**Bài 8: Chuỗi và các hàm trong chuỗi**

**1, Chuỗi là một tập hợp các kí tự Unicode.**

* Python không có kiểu kí tự (character) giống các ngôn ngữ lập trình khác, thay vào đó ta có thể hiểu rằng ký tự là một xâu có độ dài bằng 1.

**2, Để khai báo một chuỗi trong python ta có thể sử dụng dấu nháy đơn hoặc nháy đôi.**

* Ví dụ: “Mai Văn Hiếu” hoặc ‘Mai Văn Hiếu’, đều là một chuỗi, nhưng nên dùng “” vì có TH chuỗi chứa dấu ‘ .
* Ví dụ: string = ‘I’m hieu’ => lỗi syntaxError
* Hoặc có thể dùng “”” đề tạo một chuỗi trên nhiều dòng
* Ví dụ:

**Shape

Description automatically generated with medium confidence**

*+ Nó được sử dụng như comment nếu như không gán cho một biến cụ thể.*

**3, Một số ký tự đặc biệt trong chuỗi.**

+ \n: Ngắt xuống dòng và bắt đầu dòng mới.

+ \t: Đẩy nội dung phía sau nó cách một tab.

+ \a: Chuông cảnh báo.

+ \b: Xóa bỏ khoảng trắng phía trước nó.

* Để in ra các kí tự đặc biệt thì ta dùng \ trước kí tự đó.

Ví dụ:

Shape

Description automatically generated with medium confidence

Table

Description automatically generated

**4, Fomat chuỗi.**

* + **print(“%kiểu dữ liệu” %giá trị).**
* Kiểu dữ liệu mà muốn binding và thay thề vào vị trí đó.
* Giá trị là giá trị mà muốn binding vào vị trí được xác định trong chuỗi.

Ví dụ:

**Shape

Description automatically generated with low confidence**

Ta thấy nếu muốn binding nhiều chuỗi vào trong chuỗi thì mỗi giá trị ta muốn binding cách nhau bới 1 dấu , .

Mặc định nếu ta làm theo kiểu fomat này thì mỗi giá trị khi hiển thị ra sẽ được cách nhau bởi sep. Sep mặc định là bằng 1 dấu cách. Ta cũng có thể thay đổi sep để chèn ký tự vào giữa các giá trị.

Ví dụ: Chèn ký tự ký ‘—‘ vào giữa các biến, và cách xóa màn hình console khi debug code.

Graphical user interface, text

Description automatically generated

**+ Định dạng bằng phương thức format.**

* Các định dạng tương tự này cho phép python định dạng chuỗi một cách tốt hơn, và nội dung và thẩm mĩ.
* Ví dụ:

Text

Description automatically generated

Ta có thể chỉ định vị trí các giá trị sẵn cho các biến.

Ví dụ:

Text

Description automatically generated

Ta cũng thể làm tròn số bằng các fomat như sau.



Shape

Description automatically generated with medium confidence

Ta vừa có thể ép kiểu, chọn số khoảng trống hiện thị, làm tròn và con căn lề trái phải.

+ Các dạng fomat.

Table

Description automatically generated with medium confidence

**5, Truy cập tới từng giá trị của chuỗi.**

+ Chuỗi trong python được lưu trữ vào trong các ô nhớ tương ứng với một kí tự đơn (khác với các ngôn ngữ khác ) và các kí tự này được xếp liên tiếp với nhau. Vậy kiểu dữ liệu chuỗi trong python có thể được truy xuất đến từng ký tự trong nó

Cú pháp:



Với: index là vị trí kí tự muốn lấy ra, có thể truy xuất cả 2 chiều.

+ Tính từ đầu thì nó bắt đầu từ 0.

+ Tính từ cuối thì nó bắt đầu từ -1.

Ví dụ:

Shape

Description automatically generated with medium confidence

**Nếu trong trường học muốn lấy nội dung của đoạn chuỗi trong một đoạn chuỗi thì ta sử dụng.**



* Start: là vị trí đầu tiên lấy, nếu để trống thì nó sẽ lấy từ 0.
* End: là vị trí cuối lấy, nếu để trống end thì nó sẽ lấy hết chuỗi.
* Bước giúp ta cắt từng đoạn nhỏ.

Ví dụ:

Shape

Description automatically generated with medium confidence

Để nối 2 chuỗi với nhau chúng ta dùng toán tử +.

Ví dụ:

**Shape

Description automatically generated with medium confidence**

**6, Một số hàm và phương thức để xử lý chuỗi trong python.**

**6.1, Capitalize().**

+ Hàm này có tác dụng in hoa chữ cái đầu tiên của chuỗi.

Ví dụ:



**6.2, Center().**

+ Hàm này có tác dụng trả về chuỗi được hiển thị ở giữa một chuỗi.

Cú pháp:



- string: là chuỗi cần chuyển đổi.

- len: là số lượng ký tự của chuỗi mới.

- char: là ký tự sẽ hiện thì ở 2 bên chuỗi cũ. Mặc định nó sẽ là khoảng trắng.

Ví dụ:

Graphical user interface

Description automatically generated with medium confidence

*Chú ý: Nếu len mà nhỏ hơn độ dài chuỗi cần sử lý thì nó sẽ trả về chuỗi ban đầu.*

**6.3, count().**

+ Hàm này có tác dụng đếm xem trong chuỗi có bao nhiêu ký tự cần tìm.

Cú pháp:   


- s: là chuỗi cần tìm kiếm và đếm.

- strart: là index bắt đầu của chuỗi cần tìm, mặc định thì strart = 0.

- end: là index kết thúc của chuỗi cần tìm, mặc đình thì end = len() của chuỗi.

Ví dụ:

Text

Description automatically generated with low confidence

**6.4, encode().**

+ Hàm này có tác dụng encode(mã hóa) một chuỗi.

Cú pháp:



Text

Description automatically generated

Ví dụ:

Shape

Description automatically generated with medium confidence

**6.5, decode().**

+ Hàm này có tác dụng decode(giải mã) chuỗi trong python.

Cú pháp:



+ type và mode hoàn toàn giống encode().

+ Phương thức này được dùng để kiểm tra xem một chuỗi có chứa toàn các ký tự không.

Ví dụ:

Shape

Description automatically generated with medium confidence

**6.6, endswith().**

+ Hàm này có tác dụng kiểm tra xem chuỗi hoặc khoảng chuỗi có được kết thúc bằng ký tự nào đó hay không. Nó sẽ trả về True nếu đúng và False nếu sai.

Cú pháp:



Trong đó:

Text

Description automatically generated

Ví dụ:

Text

Description automatically generated with low confidence

**6.7, expandtabs().**

+ Hàm này có tác dụng tìm kiếm thay thế \t bằng các ký tự khoảng trắng.

Cú pháp:



Len: là số lượng khoảng trắng mà bạn muốn thay thế cho một \t. Mặc định thì len = 8.

**Ví dụ:**

A picture containing shape

Description automatically generated

**6.8, find().**

+ Hàm này có tác dụng tìm kiếm một chuỗi trong một chuỗi hoặc khoảng chuỗi. Nó sẽ trả về là vị trí bắt đầu của chuỗi tìm được trong chuỗi nếu tìm thấy và nếu không tìm thấy nó sẽ trả về -1.

Cú pháp:



Trong đó:

Text

Description automatically generated

Ví dụ:

Text

Description automatically generated

**6.9, index().**

+ Hàm này tương tự hàm find(), khác ở chỗ nếu không tìm thấy thì hàm này sẽ gọi exception.

Ví dụ:

Text

Description automatically generated

**6.10, isalnum().**

+ Hàm này có tác dụng kiểm tra xem một chuỗi có phải là chứa duy nhất các ký tự chữ hoặc số hay không.

- Nó sẽ trả về True nếu chuỗi chỉ chứa các ký tự chữ hoặc số.

- Nó sẽ trả về False nếu chuỗi chứa ký tự khác chữ và số.

Ví dụ:

Text

Description automatically generated

**6.11, isalpha().**

+ Hàm này có tác dụng kiểm tra xem một chuỗi có phải là chứa duy nhất các ký tự chữ hay không?

* Nó trả về True nếu chuỗi chỉ chứa duy nhất các ký tự chữ trong bảng chữ cái
* Nó trả về False nếu chuỗi chứa số hoặc ký tự đặc biệt khác.

Ví dụ:

Text

Description automatically generated

**6.12, isdigit().**

+ Hàm này có tác dụng kiểm tra xem một chuỗi có phải là chứa duy nhất các chữ số hay không?

Ví dụ:

Text

Description automatically generated

**6.13, islower().**

+ Hàm này có tác dụng kiểm tra xem một chuỗi có phải là in thường hay không?

Ví dụ:

Text

Description automatically generated

**6.14, isupper().**

+ Hàm này có tác dụng kiểm tra xem một chuỗi có phải là in Hoa không?

Ví dụ:

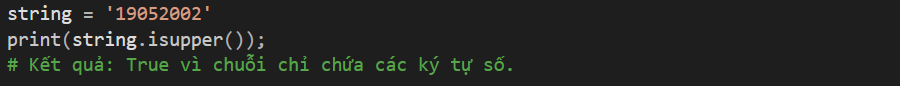
Text

Description automatically generated

**6.15, isnumeric().**

+ Hàm này có tác dụng kiểm tra xem một chuỗi có phải chỉ chứa duy nhất các ký tự số hay không?

Ví dụ:



**6.16, ispace().**

+ Hàm này có tác dụng kiểm tra xem một chuỗi có phải chỉ chứa duy nhất các ký tự khoảng trắng hay không. Return True or False.

Ví dụ:

Text

Description automatically generated with medium confidence

**6.17, istitle().**

+ Hàm này có tác dụng kiểm tra xem một chuỗi có phải là chuỗi title không.

* Chuỗi title là chuỗi có các chữ cái đầu đều được viết hoa(Mai Van H).

Ví dụ:

A picture containing text

Description automatically generated

**6.18, join().** + Hàm này có tác dụng nối một list các chuỗi thành một chuỗi sử dụng ký tự cho trước.

Cú pháp:



Trong đó: squence là string, list , ... mà ta cần nối các string vào.

Ví dụ:

Shape

Description automatically generated with medium confidence

- Mỗi ký tự sẽ được chèn vào giữa các phần từ của list như các mắt xích.

Gần giống như tham số sep.

Graphical user interface, text

Description automatically generated

Ta cũng có thể sử dụng join.

Text

Description automatically generated

Muốn sử dụng hàm join thì phải ép kiểu về str.

**6. 19, len().**

+ Hàm này có tác dụng trả về độ dài của chuỗi.

Ví dụ:

****

**6.20, ljust().**

+ Hàm này có tác dụng trả về một chuỗi với độ dài length được xác định, nếu chuỗi được chọn nhỏ hơn width thì nó sẽ sử dụng char đề bù chỗ thiếu đó về phía bên phải chuỗi.

Cú pháp:



Trong đó:

* Length là độ dài của chuỗi mới cần in ra.
* Char là ký tự bù vào chuỗi mới khi độ dài chuỗi cũ < length. Mặc định char là khoảng trắng.

Ví dụ:

Text

Description automatically generated with low confidence

+ Nếu len(name) > length thì nó sẽ hiện thị chuỗi ba đầu(name).

**6.21, rjust().**

+ Hàm này tương tự ljust(), với ljust() thì bù vào phía cuối thì rjust() lại bù vào đầu chuỗi.

Ví dụ:

A screenshot of a computer

Description automatically generated with low confidence

**6.22, lower().**

+ Hàm này có tác dụng chuyển đổi chuỗi về dạng in thường.

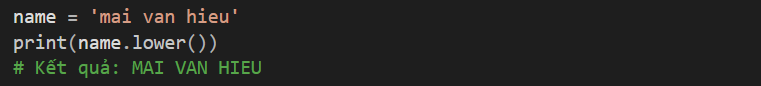
Ví dụ:



**6.23, upper().**

+ Hàm này có tác dụng chuyển đổi chuỗi về dạng in hoa.

Ví dụ:



**6.24, lstrip().**

+ Hàm này có tác dụng loại bỏ đi các ký tự char ở phần đầu của chuỗi.

Cú pháp:



Trong đó: char là kí tự ta muốn loại bỏ. Mặc định thì char sẽ là khoảng trắng.

Ví dụ:

Shape

Description automatically generated with medium confidence

**6.25, rstrip().**

+ Hàm này có tác dụng loại bỏ đi các ký tự char ở phần cuối của chuỗi.

Cú pháp:



Ví dụ:

A picture containing shape

Description automatically generated

**6.26, strip().**

+ Hàm này là sự kết hợp của lstrip() và rstrip(). Nó sẽ loại bỏ các ký tự char ở cả hai đầu của chuỗi.

Ví dụ:

A screenshot of a computer

Description automatically generated with medium confidence

**6.27, rfind().**

+ Hàm này tương tự hàm find(), nhưng hàm này nó sẽ trả về **index của chuỗi cuối cùng** tìm được trong chuỗi.

Ví dụ:

Text

Description automatically generated with medium confidence

+ Nếu không tìm thấy chuỗi thì trả về -1.

**6.28, rindex().**

+ Hàm này tương tự hàm index(), nhưng hàm này nó sẽ trả về index của chuỗi cuối cùng tìm được trong chuỗi.

Cú pháp:



Ví dụ:

Shape

Description automatically generated with medium confidence

+ Nếu không tìm thấy chuỗi gọi hàm exception.

**6.29, replace().**

+ Hàm này có tác dụng tìm kiếm và thay thế chuỗi tìm được bằng chuỗi mới.

Cú pháp:



Trong đó: Text

Description automatically generated

Ví dụ:

Text

Description automatically generated

**6. 30, max().**

+ hàm này trả về chữ cái có độ sắp xếp cuối cùng theo bảng chứ cái alphabet nằm trong chuỗi.

Ví dụ:

Shape

Description automatically generated with medium confidence

**6.31, min().**

+ Hàm này trả về chữ cái có độ sắp xếp đầu tiên theo bảng chữ cái alphabet nằm trong chuỗi.

Ví dụ:

Shape

Description automatically generated with medium confidence

**6.32, title().**

+ Hàm này có tác dụng chuyển đổi chuỗi sang kiểu title.

Ví dụ:

Shape

Description automatically generated with medium confidence

* Chuỗi kiểu title là chuỗi có các chữ cái đầu được in hoa.

**6.33, swapcase().**

+ Hàm này có tác dụng chuyển đổi chuỗi sang dạng nghịch đảo của nó.(nghịch đảo ở đây là hoa – thường).

Ví dụ:

Text

Description automatically generated with medium confidence

**6.34, zfill().**

+ Hàm này có tác dụng như hàm ljust(), nhưng nó sẽ chỉ thêm được các ký tự zero ‘0’ vào trước chuỗi.

Ví dụ:

Shape

Description automatically generated with medium confidence

Khi len(name) > length thì nó sẽ in ra chuỗi ban đầu(name).

**6.35, isdecimal().**

+ Hàm nó có tác dụng gần như hàm isdigit(), nó sẽ trả về True nếu chuỗi cần kiểm tra chỉ chứa các số thập phân,... và False ngược lại.

Ví dụ:

Text

Description automatically generated

**­­6.36, split().**

+ Hàm này có tác dụng tách chuỗi thành mảng bởi các char.

Cú pháp:



Trong đó:

+ char là ký tự cần tìm và tách chuỗi bởi nó. Mặc định char là khoảng trắng.

+ max là số lượng tách tối đa.

Ví dụ:

Shape

Description automatically generated with medium confidence

Ta thấy, nó từ một cái chuỗi tách thành list thành phần list được phân tách bởi kí tự char mặc định là khoảng cách.

Shape

Description automatically generated with medium confidence

**6.37, splitlines().**

+ Hàm này có tác dụng tách chuỗi bới các ký tự \n(xuống dòng).

Cú pháp:



Trong đó: max là số lần có thể cắt tối đa.

Ví dụ:

Graphical user interface, text

Description automatically generated

**6.38, startswith().**

+ Hàm này có tác dụng kiểm tra xem chuỗi hoặc khoảng chuỗi có được bắt đầu bới ký tự nào đó hay không. Nó sẻ trả về True nếu đúng và False nếu sai.

Cú pháp:



Trong đó:

* Str là chuỗi cần xác thực xem có phải chuỗi bắt đầu không.
* Start là index bắt đầu chuỗi cần xác thực. Mặc định là 0.
* End là index kết thực chuỗi cần xác thực. Mặc đình là len().

Ví dụ:

Text

Description automatically generated

**6.39, maketrans() + translate().**

+ Hàm maketrans có tác dụng tạo ra các translation cho chuỗi

Cú pháp:



* in là chuỗi các ký tự cần tìm.
* out là chuỗi chứa các ký tự cần thay thế.

+ Hàm translate có tác dụng thực thi việc dịch chuỗi

Ví dụ:`

Text

Description automatically generated

**Bài 9: Hàm trong python.**

**+ Hàm là một đoạn code, khi gọi hàm thì thực ra là đang gọi tới đoạn code đó(hàm có thể hiểu là viết tắt của một đoạn code)**

* **Mục đích là để sử dụng lại nó và làm cho chương trình logic hơn.**

**1, Khai báo hàm trong python.**

Để khai báo một hàm ta dùng từ khóa def.

Cú pháp:



Trong đó:

* Funname: là tên hàm.
* Param: là tham số của hàm.

**2, Cách gọi hàm.**

+ Để gọi hàm đã được khai báo thì ta sử dụng cú pháp:



Ví dụ:

Shape

Description automatically generated with medium confidence

**3, Hàm không có trả về giá trị.**

+ Hàm không có giá trị trả về là hàm khi kết thúc nó không trả về giá trị.

Shape

Description automatically generated with medium confidence

**4, Hàm có giá trị trả về.**

+ Hàm có giá trị trả về là hàm có thể truyền tham số và trả về giá trị bằng từ khóa return < value>.

Ví dụ: Viết hàm tính tổng của một list vừa nhập từ bàn phím.

A picture containing text

Description automatically generated

**Nếu khai báo hàm có truyền tham số mà khi gọi hàm nếu không truyền tham số thì chương trình sẽ báo lỗi.**

**5, Giá trị mặc nhiên của tham số hàm**

+ Khi truyền tham số thực tế vàl thì nó sẽ lấy giá trị của tham số thực tế, còn nếu không thì nó sẽ lấy giá trị mặc nhiên.

Ví dụ:

A picture containing text

Description automatically generated

+ Giá trị mặc nhiên phải khai báo theo thứ tự từ trái qua phải.

* Chỉ sử dụng tham số thứ 3
* Chỉ sử dụng tham số thứ 2, 3.
* Sử dụng cả 3 tham số
* Không thể sử dụng tham số trước mà không sử dụng tham số sau.

Vì khi gọi hàm tham số đầu tiên nó sẽ gán cho tham số đầu tương ứng ở hàm, nên nếu giá trị cuối của hàm không được khai báo giá trị mặc nhiên thì ta không thể truyền giá trị vào nó khi trong hàm có biến khác đứng trước.

Ví dụ:

Text

Description automatically generated

+ Các biến khai báo trong hàm thì chỉ có thể dùng trong hàm đó, ta không thể thay đổi hay truy cập nó ngoài hàm, trừ list.

**6, Biến toàn cục(global variable).**

+ Biến toàn cục là biến có thể sử dụng ở tất cả mọi nơi trong chương trình.

Cú pháp:



Ví dụ:

A picture containing text

Description automatically generated

**7, Truyền vô số tham số vào hàm.**

+ Thực tế thì không phải lúc nào hàm nào ta cũng có thể xác định chính xác số tham số truyền vào hàm, vậy nên trong python có cung cấp một param gọi là biến đại diện cho các biến truyền vào hàm bằng cách thêm dấu \* vào trước param đó.

Ví dụ: Tính tổng các tham số truyền vào hàm.

Text

Description automatically generated

Val như là một tập hợp chứa các tham số trong hàm.

**Bài 10: Tuple trong python.**

**1, Type trong python.**

Typle được sử dụng để lưu trữ nhiều mục trong một biến duy nhất.

Typle trong python là một kiểu dữ liệu dùng để lưu trữ các đối tượng không thay đổi về sau(giống như hằng số), và nó được sắp xếp theo thứ tự và không thể thay đổi, cho phép các giá trị trùng lặp.

+ Để khai báo một enum:



Trong đó các Val là các giá trị của Tuple.

Ví dụ: Khai báo 1 typle chứa 3 ngày đầu tuần.



Có thể bỏ () thì python cũng nhận định nó là một Typle.



Để khai báo một Typle trống thì ta chỉ cần khai báo như sau:



Nếu Tuple chỉ có chứa duy nhất một giá trị thì bắt buộc ta phải thêm dấu phẩu sau nó.



**2, Truy cập đến các phần tử trong Tuple.**

+ Tương tự đối với chuỗi và list.

Ví dụ:

Shape

Description automatically generated with medium confidence

Để lấy ra độ dài Tuple ta sử dụng hàm len().

**3, Các tác vụ khác trên Tuple.**

* Khi một tuple được tạo, ta không thể thay đổi giá trị của nó. Nó gọi là bất biến.
* Nhưng ta có thể chuyển đổi bộ thành một list, thay đổi list và chuyển đổi lại thành tuple.

Ví dụ:

**Text

Description automatically generated**

**Ta có thể áp dụng cho các tác vụ sau:**

**+ Thay đổi giá trị**

Ví dụ:

Graphical user interface

Description automatically generated with medium confidence

**+ Xóa Tuple.**

* Để xóa một Tuple hoàn toàn hay nhiều tuple thì ta sử dụng hàm del.

Ví dụ: xóa phần tử của Tuple

Text

Description automatically generated with medium confidence

**+ Thêm phần tử.**

* Thực ra đây không hẳn là thêm mà chúng ta có thể tạo ra được một Tuple mới từ các Tuple bằng các sử dụng biểu thức + hai tuple.

Ví dụ:

Text

Description automatically generated

* Thêm bằng cách chuyển đổi thành list.

Text

Description automatically generated with low confidence

**4, Tuple lồng nhau.**

Cũng giống như list, ta có thể khai báo các Tuple lồng nhau.

Ví dụ:

**Text

Description automatically generated**

**5, Giải nén Tuple.**

Khi chúng ta tạo một Tuple, chúng ta thường gán giá trị cho nó, gọi là “đóng gói” một Tuple.

Ví dụ:



Tuy nhiên, để chích dẫn các giá trị trở lại thành các biến, thì gọi là “giải nén”.

Ví dụ:

Text

Description automatically generated with medium confidence

Lưu ý: Số lượng biến phải khớp với số lượng giá trị trong một Tuple, nếu không, thì ta phải sử dụng \* để gán tất cả các giá trị còn lại dưới dạng list.

**+ Sử dụng \*.**

Ví dụ: Gán phần còn lại của các giá trị dưới dạng list có tên là red.

Text

Description automatically generated

Nếu \* thêm vào tên biến trước biến cuối cùng thì python sẽ gán giá trị cho biến cho đến khi số giá trị còn lại khớp với số biến còn lại.

Text

Description automatically generated

**Bài 11: Dictionary trong python.**

**1, dictionary.**

+ Kiểu dữ liệu dictionary trong python là một kiểu dữ liệu lưu trữ các giá trị chứa key và value, nhìn một cách tổng quát thì giống Json. Và đối với kiểu dữ liệu này thì các giá trị bên trong nó không được sắp xếp theo một trật tự nào cả.

+ Để khai báo một dictionary ta dùng dấu {}.



Key và value là các key và giá trị của kiểu dữ liệu dictionary.

+ Quy tắc về tên key.

* Các phần tử đều phải có **key**.
* Và **Key** chỉ có thể là số hoặc chuỗi.
* **Key** phải là duy nhất, nếu không nó sẽ nhận giá trị của phần tử có key được xuất hiện cuối cùng.
* **Key** khi đã được khai báo thì không thể đổi được tên.
* **Key** có phân biệt hoa thường.

Ví dụ: Khai báo một dictionary có tên là Person.

Shape

Description automatically generated with medium confidence

**2, Truy cập đến các phần tử trong dictionary.**

Vì dictionary không được sắp xếp theo thứ tự nào nên chúng ta chỉ có thể truy xuất chúng ta phải dựa vào các key của nó.

Cú pháp:



Ví dụ:

A picture containing text

Description automatically generated

**3, Thay đổi giá trị của dictionary.**

+ Để thay đổi giá trị của dictionary thì ta truy cập tới phần tử cần thay đổi và thay đổi.

Ví dụ:

**Text

Description automatically generated with low confidence**

**4, Xóa phần tử trong dictionary.**

+ Để xóa phần tử trong dictionary thì ta sử dụng hàm del và chọn phần tử cần xóa.

Ví dụ:

Text

Description automatically generated

+ Nếu muốn xóa tất cả phần tử trong dictionary thì ta sử dụng phương thức clear.



+ Nếu muốn xóa hẳn dictionary thì ta dùng hàm del.



**5, dictionary lồng nhau.**

+ Cũng giống như list, Tuple thì trong dictionary ta cũng có thể lồng bất kì kiểu dữ liệu nào vào nó.

A picture containing text

Description automatically generated

**Bài 12: Input và Đọc ghi file trong python**

**1, Input().**

+ Trong python cũng cấp hàm input để nhận dữ liệu từ người dùng vào commandline(lòng lệnh).

Cú pháp:



Với something là nội dung cần hiện thị trước khi nhập dữ liệu.

Ví dụ:



**Chú ý: dữ liệu nhập vào luôn là kiểu ‘str’ .**

**2, Đọc ghi file.**

+ File là thứ rất quan trọng trong các dự án, ví dụ như chúng ta cần ghi log ra một file để sau này có thể kiểm soát được... Và ngôn ngữ nào cũng hỗ trợ chúng ta làm việc với file. Và trong python ta làm như sau.

* **Mở File.**

+ Để mở file trong python chúng ta sử dụng hàm open.



Trong đó:

* FilePath là đường dẫn đến địa chỉ của file.
* Mode là thông số thiết lập chế độ chúng ta mở file cấp nhưng quyền gì. Mặc đình mode sẽ là r (chế độ chỉ được đọc).
* Buffer là thông số đệm cho file mặc định là nó sẽ là 0.

Các chế độ mode.

Graphical user interface, text, application, email

Description automatically generated

Ví dụ: mở file input.md ở chế độ đọc và ghi nếu không tồn tại file thì nó sẽ tạo một file mới và ghi nội dung vào đó, nếu tồn tại thì nó sẽ ghi đè lên nội dung cũ trong file.



* **Đóng file.**

+ Để đóng một file đang được mở, thì chúng ta sử dụng phương thức close().

Cú pháp:



Trong đó: file0bject là đối tượng mà ta thu được khi sử dụng hàm open().

+ Để đảm bảo quy chế đóng mở và giải phóng bộ nhớ cho chương trình thì ta phải luôn đóng file khi kết thúc thao tác với file.



* **Đọc file.**

+ Sau khi mở được file ra để đọc được file thì chúng ta sử dụng phương thức read().

Cú pháp:



Trong đó:

- fileOject là đối tượng thu được khi sử dụng hàm open().

* Length là dung lượng của dữ liệu mà chúng ta muốn đọc, nếu để trống tham số này thì nó sẽ đọc đến hết file hoặc nếu file lớn quá thì nó sẽ đọc đến khi giới hạn của bộ nhớ cho phép.

Ví dụ: Đọc và xuất ra dữ liệu đọc được trong file input.

A picture containing shape

Description automatically generated

* **Ghi file.**

+ Để ghi được file thì ta phải chắc chắn là file đang trong trạng thái mở và chế độ có thể ghi.

+ Sử dụng phương thức write().

Cú pháp:



Trong đó: - fileOject là đối tượng thu được khi sử dụng hàm open().

* Data là dữ liệu mà ta muốn ghi vào trong file.

Ví dụ: Ghi vào file output.txt

Text

Description automatically generated

**3, Các thuộc tính trong file.**

+ Nếu cần kiểm tra các thông số của file hiện tại thì ta sử dụng các thuộc tính:

|  |  |
| --- | --- |
| Thuộc tính | Chú thích |
| file.name | Trả về tên của file đang được mở |
| file.mode | Trả về chế độ của file đang mở |
| file.closed | Trả về True nếu file đã được đóng, False nếu file chưa đóng. |

Ví dụ:

A picture containing text

Description automatically generated  
 **4, Phương thức trong đọc file.**

**4.1,** **readline()**

+Phương thức này cho phép đọc từng dòng trong file:

Cú pháp:



Ví dụ: Đọc từng dòng của file input.txt

Chart, text

Description automatically generated

+ Trường hợp không dùng rstrip(‘\n’)

Text

Description automatically generated

+ Trường hợp dùng rstrip(‘\n’)

Text

Description automatically generated

**4.2, readlines()**

+ Phương thức này trả về toàn bộ dòng còn lại trong file và trả về giá trị rỗng khi kết thúc file.

Cú pháp:



Ví dụ:

Text

Description automatically generated

**4.3, tell() và seek().**

**a, tell().**

+ Phương thức này trả về vị trí hiện tại của con trỏ bên trong file.

Ví dụ:

Text

Description automatically generated’

**b, seek().**

+ phương thức này thay đổi vị trí hiện tại bên trong file.

Ví dụ:

Text

Description automatically generated

Ta thấy, name và name\_ giống nhau vì file đã được đọc 2 lần cùng 1 vị trí.

**Bài 13, Modules trong python.**

**1, Modules trong python.**

+ Module có thể coi như là một file có phần mở rộng .py, các file này chứa mã python đó có thể là các biến, hàm, hoặc lớp nào đó.

* Nhưng hàm python hay phải sử dụng sẽ được định nghĩa trong những module và nhập vào python thay vì code lại nhưng dòng code đó.
* Module dùng khi chúng ta muốn biến các chương trình lớn thành những file nhỏ hơn để dễ quản lí và tổ chức, ,mỗi module có một hoặc các chứ năng cụ thể.
* Sử dụng module để có thể sử dụng lại code(nó như các thư viện hay các hàm do mình tự định nghĩa và ta chỉ có việc import vào và sử dụng như các module có sẵn).
* Các thuộc tính tương tự có thể đặc trong cùng một module(module trong module).

Ví dụ như làm tròn có thể làm tròn lên làm tròn xuống.

* Tạo một module có tên là module1. Sau đó nó sẽ tạo một mục \_\_pycache\_\_ lưu module đã định nghĩa.



* Import module1 vào và sử dụng như sau:

A picture containing text

Description automatically generated

* Ta sẽ tìm hiểu sau hơn qua từng kiểu import.

**a, import.**

* + Để import một module có vào trong file hiện tại thì

Cú pháp:

****

Trong đó: module1... là các modules mà ta muốn import vào file hiện tại.

* Trong python cũng có sẵn các module nhưng ta cũng có thể tự tạo ra các module do mìn định nghĩa và import vào file.

+ Modules có thể có nhiều thuộc tính.

Ví dụ: module1 cộng 2 số



* Import module vào và sử dụng.

A picture containing shape

Description automatically generated

Ví dụ: module có nhiều thuộc tính

Shape

Description automatically generated with medium confidence

Text

Description automatically generated with medium confidence

* ***Tên file chứa code mà ta muốn import chính là tên module vd(module1).***

**b, from – import.**

+ Giả sử trong trường học ta không muốn sử dụng toàn bộ module mà chỉ muốn sử dụng một số thứ trong đó.

Cú pháp:



Trong đó: modules là tên module , các something là các thuộc tính ta muốn sử dụng.

+ Trường hợp muốn dùng cả thì ta thay something bằng \*.

Ví dụ:

Text

Description automatically generated

**A picture containing text

Description automatically generated**

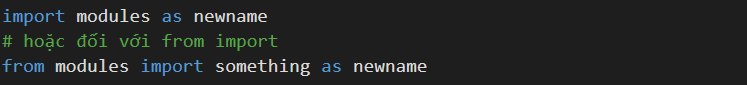
**+ Chú ý: -** khi sử dụng **from – module**  thì ta phải chỉ đích danh các đối tượng đó, hay còn gọi là gọi thẳng đối tượng đó cần sử dụng.

+ Khi sử dụng **from – import \***  thì tên của các đối tượng không được bắt đầu bằng ‘\_’.

**c, Định danh cho module.**

* Trường hợp modules khó nhớ và dài mà ta không muốn gọi modules như thế, hoặc đơn giản là muốn đổi tên thì trong python ta có thể gán định danh mới cho modules khi import chúng bằng keyword as.

Cú pháp:

****

Ví dụ:

Text

Description automatically generated

**d, Kiểm tra thuộc tính và phương thức của module.**

* Hàm dir có công dụng trả về danh sách các tên được định nghĩa trong module.

Ví dụ:

**Text

Description automatically generated with low confidence**

+ Đầu ra là danh sách các tên được định nghĩa trong module, dược sắp xếp theo thứ tự. Các thuộc tính bắt đầu bằng \_\_ là các thuộc tính mặc định của python, được liên kết với module.

**e, Tải lại module với hàm reload.**

* Khi ta import một module thì nó sẽ thực hiện một lần duy nhất, dù ta có sử dụng đoạn code import bao nhiêu lần đi nữa.
* Để sử dụng nhiền lần thì ta sử dụng hàm reload(), để tải lại module, rồi sử dụng như bình thường.

Cú pháp:



Ví dụ:

**Bài 14, Packages trong python.**

**1, Package.**

+ **pckage** trong python là một thư mục chưa một hoặc nhiều modules hay các package khác nhau, nó được tạo ra nhằm mục đích phân bố các modules có cùng chức năng hay một cái gì đó, dễ quản lí source code