Lập trình Python nâng cao

Lập trình hướng đối tượng

Khai báo Class trong Python

Trong Python, một Class thường có cấu trúc như sau:

|  |
| --- |
| class ClassName:        # Hàm khởi tạo đối tượng    def \_\_init\_\_(self, var1, var2, ...):        self.var1 = var1        self.var2 = var2        ....            # Các method        def method1(self, ...)        # Nội dung hàm            def method2(self, ...)        # Nội dung hàm |

Ví dụ:

|  |
| --- |
| class Person:    def \_\_init\_\_(self, name, address):        self.name = name        self.address = address            def sayHello(self):        print(f'Xin chào, tên tôi là {self.name}')            def updateAddress(self, newAddress):        self.address = newAddress |

### Khởi tạo và sử dụng đối tượng

Cú pháp để khởi tạo một đối tượng của một Class:

obj = ClassName(var1, var2, …)

Để truy nhập đến phương thức của đối tượng:

obj.method1(...)

obj.method2(...)

Để truy nhập đến thuộc tính đối tượng:

|  |
| --- |
| obj.attr1 |

Ví dụ:

|  |
| --- |
| person = Person('Nguyễn Văn An', 'Hà Nội') person.sayHello() person.updateAddress('TP. HCM') print(f'{person.name} sống tại {person.address}') |

### Thừa kế

class ChildName(ParentClass):

Lớp con sẽ kế thừa các thuộc tính và phương thức của lớp cha.

Ngoài ra lớp con có thể “ghi đè” phương thức đã có ở lớp cha bằng cách khai báo một phương thức cùng tên.

class Person:  
   def \_\_init\_\_(self, name, address):  
       self.name = name  
       self.address = address  
         
   def sayHello(self):  
       print(f'Xin chào, tên tôi là {self.name}')      
         
   def updateAddress(self, newAddress):  
       self.address = newAddress  
  
class Student(Person):  
   def \_\_init\_\_(self, name, address, schoolYear):  
       self.name = name  
       self.address = address  
       self.schoolYear = schoolYear  
         
   def sayHello(self):  
       print(f'Xin chào, tên tôi là {self.name}, học khóa {self.schoolYear}')  
     
student = Student('Nguyễn Văn An', 'Hà Nội', 2019)  
student.sayHello()

Nhược điểm của thread:

* Mặc dù các thread chạy song song với nhau nhưng tại một thời điểm chỉ một thread được chạy. Nói cách khác, dù nhiều thread được tạo ra, chương trình chỉ sử dụng một core CPU để chạy các thread theo trình tự luân phiên nhau. Nguyên nhân của hiện tượng này là do Python sử dụng **GIL** (**Global interface lock**) để tránh hiện tượng nhiều thread cùng làm thay đổi một đối tượng tại một thời điểm dẫn đến hiện tượng dữ liệu bị phá hỏng. Bên trong thread, những đoạn nào không sử dụng GIL thì có thể thực hiện đồng thời trên nhiều core CPU, những đoạn nào sử dụng GIL thì sẽ luân phiên nhau thực hiện trên một core CPU.

Các đoạn code Python thông thường đều sử dụng GIL. Chỉ một số hàm của các thư viện của Python không sử dụng GIL như các hàm truy nhập IO (bàn phìm, File, Download, Upload qua internet …). Do đó về cơ bản, việc sử dụng nhiều Thread không giúp tăng tốc độ xử lý chương trình.

### Process

Process cũng giống như Thread cho phép xử lý các tác vụ song song với nhau, nhưng cho phép các công việc được thực hiện đồng thời trên nhiều core CPU một lúc.

from multiprocessing import Process  
  
def processFunc(pid):      
   s = 0  
   for i in range(10000000):  
       s += 1  
         
   print(f'Process id : {pid}, result : {s}')  
  
if \_\_name\_\_ == '\_\_main\_\_':  
   processes = []  
     
   for i in range(5):  
       p = Process(target=processFunc, args=(i,))  
       processes.append(p)  
       p.start()  
         
   for p in processes:  
       p.join()  
     
   print('All process completed')

Nếu so với cách sử dụng Thread, đoạn chương trình trên sẽ thực hiện nhanh hơn vì các tác vụ tính toán được thực hiện trên nhiều core CPU cùng lúc.

Nhược điểm của Process:

* So với Thread, khi sử dụng process thì các tác vụ được phân phối đều trên các core CPU. Tuy nhiên nhược điểm của Process là không thể truy nhập đến một đối tượng chung của chương trình. Khi Process được tạo ra, nó clone toàn bộ các biến của chương trình chính, do đó bên trong mỗi process, các biến có tên giống với biến của chương trình chính thực chất là các biến mới.
* .join()

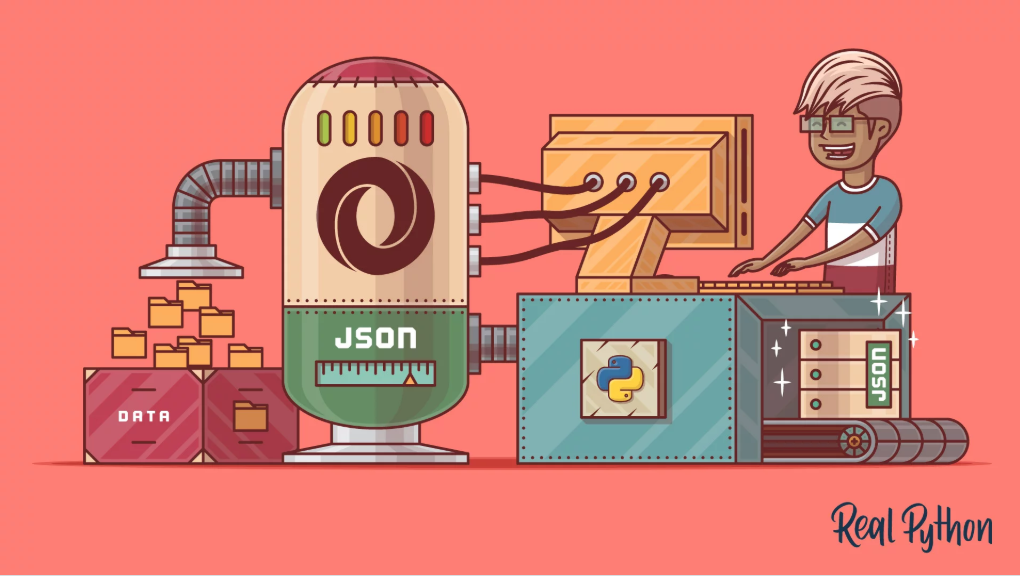
Dùng để hợp list thành 1 chuỗi

Pool cũng giống như Process, tức cho phép các tác vụ xử lý được thực hiện trên nhiều core CPU. Pool thường được sử dụng thay cho Process khi cần tập hợp kết quả các Process với nhau vào một biến chung của chương trình chính (điều mà Process không làm được như ở ví dụ minh họa phía trên)

## Làm việc với JSON (JavaScript Object Notation)

Trong Python, để chuyển đổi giữa Python Object thành JSON string và ngược lại, có thể sử dụng 2 hàm của thư viện json:

* json.loads : chuyển json string thành object
* json.dumps : chuyển object thành json string



Sự khác biệt giữa JSON và dictionary:

* JSON is a pure string written in a convention format, which does not have any characteristics of data structure. The rules of the string representation of Python's dict look similar to JSON, but the dict itself is a complete data structure that implements all of its own algorithms.
* Python's dictionary key can be any hash object, and JSON can only be a string.
* The Python dict string uses single quotation marks, and JSON enforces double quotation marks.
* You can nest tuple in Python dict. JSON can only use array.
* JSON key name must be a string, Python dict uses hashable.

Cấu trúc :

{

"Key": "Value",

"Key": "Value",

}

JSON Library in Python:

import json

|  |  |
| --- | --- |
| **Method** | **Description** |
| dumps() | encoding to JSON objects |
| dump() | encoded string writing on file |
| loads() | Decode the JSON string |
| load() | Decode while JSON file read |

Có 4 phương thức chính trong đó: thêm s tức -> string còn không có thì là dành cho file :v

## Python to JSON (Encoding)

|  |  |
| --- | --- |
| **Python** | **JSON** |
| dict | Object |
| list | Array |
| unicode | String |
| number - int, long | number – int |
| float | number – real |
| True | True |
| False | False |
| None | Null |

Chuyển đổi dữ liệu Python thành JSON được gọi là thao tác Mã hóa. Mã hóa được thực hiện với sự trợ giúp của phương thức thư viện JSON - dumps()

Phương thức dumps() chuyển đổi đối tượng dictionary của python thành định dạng dữ liệu chuỗi JSON.

import json

x = {

"name": "Ken",

"age": 45,

"married": True,

"children": ("Alice","Bob"),

"pets": ['Dog'],

"cars": [

{"model": "Audi A1", "mpg": 15.1},

{"model": "Zeep Compass", "mpg": 18.1}

]

}

# sorting result in asscending order by keys:

sorted\_string = json.dumps(x, indent=4, sort\_keys=True)

print(sorted\_string)

XML:

XML (Extensible Markup Language) là ngôn ngữ đánh dấu mở rộng. Đây là một dạng ngôn ngữ đánh dấu, có chức năng truyền dữ liệu và mô tả nhiều loại dữ liệu khác nhau.

* XML được dùng cho dữ liệu có cấu trúc.
* Về trực quan, XML khá giống với HTML.
* Tuy là văn bản, nhưng XML không phải dùng để đọc.
* XML thường rất dài.
* XML được coi là cầu nối đưa HTML tới XHTML và là nền tảng cho RDF và Web mã hóa.
* XML là một module.
* XML miễn phí bản quyền, platform độc lập và được hỗ trợ rất tốt.

1. Ưu điểm của XML

Ưu điểm lớn nhất của XML chính là sự độc lập. Điều này cũng rất dễ hiểu, vì XML được sử dụng để mô tả dữ liệu dưới dạng text, nên hầu hết các phần mềm hay các chương trình bình thường đều có thể đọc được chúng.

XML có thể đọc và phân tích nguồn dữ liệu khá dễ dàng nên nó được sử dụng với mục đích chính là trao đổi dữ liệu giữa các chương trình, các hệ thống khác nhau. XML được sử dụng cho Remote Procedure Calls nhằm phục vụ cho các dịch vụ của website.

2. Nhược điểm của XML

Tỷ lệ sai sót khi sử dụng XML để truyền dữ liệu khoảng từ 5-7%. Con số này tuy không quá cao, nhưng trên thực tế, người ta vẫn cần cân nhắc trước khi sử dụng nó để trao đổi thông tin.

Ví dụ: Decode , chuyển từ XML -> string

import xml.etree.ElementTree as ET

xml = """

   <students>

       <student>

           <name>Nguyễn Văn An</name>

           <address>Hà Nội</address>

       </student>

       <student>

           <name>Nguyễn Thị Bình</name>

           <address>TP.HCM</address>

       </student>

   </students>

"""

root = ET.fromstring(xml)

for student in root:

   name = student.find('name').text

   address = student.find('address').text

   print(name, address)

Encode , chuyển từ string -> XML

import xml.etree.cElementTree as ET

students = [

   { 'name' : 'Nguyễn Văn An', 'address' : 'Hà Nội'},

   { 'name' : 'Nguyễn Thị Bình', 'address' : 'TP.HCM'},

]

root = ET.Element("students")

for student in students:

   tag = ET.SubElement(root, "student")

   ET.SubElement(tag, "name").text = student['name']

   ET.SubElement(tag, "address").text = student['address']

tree = ET.ElementTree(root)

tree.write("output.xml", *encoding*="UTF-8")

## Làm việc với Base64

Base64 là một chương trình mã hóa chuỗi ký tự bằng cách dùng thay thế các ký tự trong bảng mã ASCII 8 bit thông dụng thành bảng mã 6 bit. Nó thường được sử dụng để mã hóa các tập tin đa phương tiện (hình ảnh, âm thanh, video,…). Ký tự 64 trong Base64 là đại diện cho 64 ký tự trong bãng mã ASCII. Base64 thường được sử dụng trong việc truyền tải email. Tuy nhiên, ngày nay người ta đã sử dụng nó vào việc truyền tải hình ảnh trên website.

## Mã hoá file theo chuẩn Base64.

Chuẩn Base64 là một tập hợp gồm các ký tự (theo đúng thứ tự) : từ A đến Z, từ a đến z, từ 0 đến 9, dấu +, dấu /  
Tổng cộng là 64 ký tự biểu diễn 64 giá trị từ 0 đến 63. Như vậy, ký tự từ A đến Z biểu diễn cho các giá trị từ 0 đến 25, từ a đến z biểu diễn cho giá trị từ 26 đến 51, từ 0 đến 9 biểu diễn cho giá trị từ 52 đến 61, dấu + biểu diễn cho giá trị 62, dấu / biểu diễn cho giá trị 63.  
  
Một ký tự biểu diễn theo mã ASCII sẽ dùng 8 bits. Một ký tự theo Base64 sẽ dùng 6 bits. Như vậy, một file ở dạng Base64 sẽ có kích thước lớn hơn khi ở dạng ASCII. Cụ thể, sẽ lớn gấp 4/3 lần (8 bits/6 bits).  
  
Để chuyển đổi file sang dạng Base64, ta thực hiện theo các bước như sau :  
1. Đọc nội dung file dưới dạng bit.  
2. Cứ 6 bits ta tách thành một nhóm để xử lý.  
3. Tra bảng mã Base64, mỗi nhóm 6 bits sẽ có giá trị tương ứng với một ký tự.  
4. Ghi ra file các ký tự đó.  
Vậy là xong.

Trong thực tế khi lưu file ảnh vào csdl hay file json thì người ta thường chuyển về định dạng Base64 rồi mới lắp vào, vì đơn giản trong file JSON không lưu trữ dấu :D

### Encode base64:

|  |
| --- |
| import base64  msg = 'Xin chào' msg\_bytes = msg.encode('utf-8') b64\_msg\_bytes = base64.b64encode(msg\_bytes) b64\_msg = b64\_msg\_bytes.decode()  print(b64\_msg) |

Hiểu như sau:

Chuỗi -> bytes -> b64\_bytes -> b64

Ta sẽ mã hóa chuỗi ký tự thường thành bytes ,sau đó lại mã hóa thành b64bytes , tiếp đến ta giải mã b64bytes trở thành chuỗi base64. Khi ấy chuỗi b64 này mới có thể lắp vào bên trong của csdl hay json được á :D

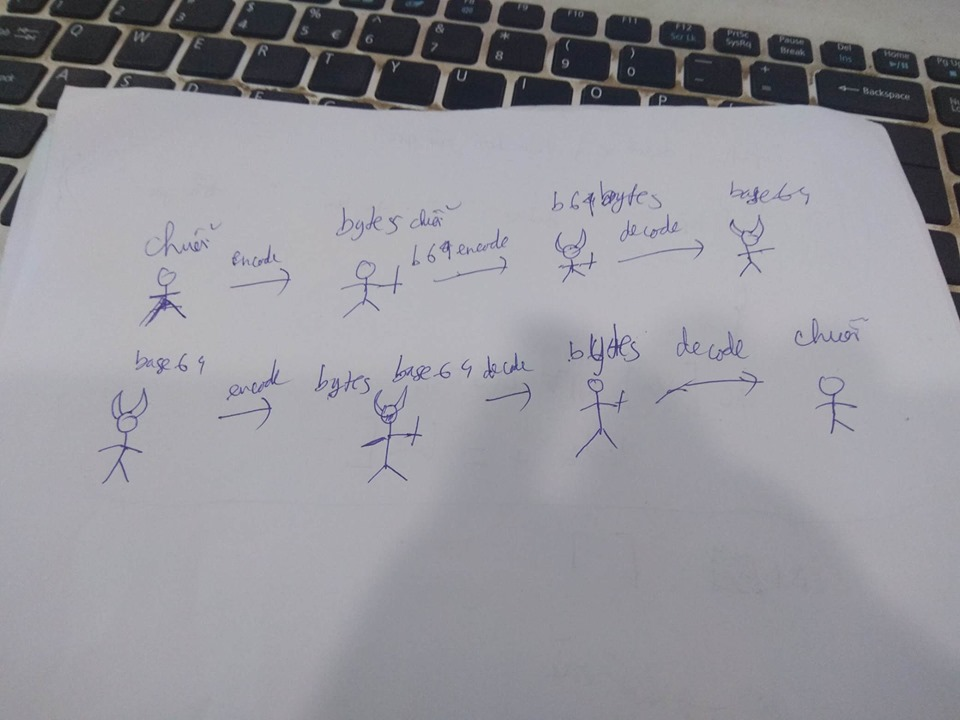
### Decode base64:

|  |
| --- |
| import base64  msg = 'WGluIGNow6Bv' msg\_bytes = msg.encode() decoded\_msg\_bytes = base64.b64decode(msg\_bytes) decoded\_msg = decoded\_msg\_bytes.decode('utf-8')  print(decoded\_msg) |

Hiểu như sau:

Ta có chuỗi base64 .

Điều đầu tiên là mã hóa thành dạng bytes , sau đó giải mã thành mã bytes , sau đó thì giải mã thành thông điệp ban đầu :D



Làm việc với csv

Để đọc, ghi file csv, có thể sử dụng thư viện csv của Python.

Đọc file csv:

|  |
| --- |
| import csv  with open('input.csv', encoding='utf-8') as fi:    reader = csv.reader(fi,delimiter=',')    rows = [row for row in reader]     for row in rows:    print(row) |

Ghi file csv:

|  |
| --- |
| with open('output.csv', 'w', newline='',  encoding='utf-8') as fo:    writer = csv.writer(fo, delimiter=',')    for row in rows:        writer.writerow(row) |

Project 1:

Hôm nay chúng ta đã học về JSON & Base64

Các bạn dành thời gian làm BTVN sau:

Danh sách học sinh của một lớp học được lấy về từ 1 web service

có dạng json như trong file students.json đính kèm, thông tin về mỗi học sinh gồm:

studentId : số hiệu học sinh

name : họ tên

address : đia chỉ

image: ảnh chân dung, mã hóa dạng base64

Viết chương trình đọc dữ liệu từ file students.json và xuất dữ liệu

ra dưới dạng csv với các cột : studentId, name, address.

Riêng trường image, tách lấy dữ liệu, giải mã và lưu thành file theo format {studentId}.jpg

Project 2:

Bài tập tổng hợp sau để ôn lại kiến thức đã học:

1.Tạo class mô tả đối tượng sản phẩm của một cửa hàng với các trường thông tin : Tên, mã, loại sản phẩm, đơn giá, tên file ảnh

2. Đọc vào 1 file csv chứa danh sách các sản phẩm của cửa hàng và chuyển danh sách sản phẩm sang dạng json (ảnh sản phẩm đọc vào từ file và chuyển sang dạng base64)

3. Tạo một bảng trong database để lưu trữ thông tin về các sản phẩm. Viết chương trình import dữ liệu từ file csv hoặc json để đưa vào database