**PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ**

**I. Khái niệm và thuật ngữ liên quan đến phần mềm mã nguồn mở**

**1. Thuật ngữ liên quan đến mã nguồn mở :**

* **Giấy phép GPL (General Public License) :** là một trong những giấy phép được sử dụng rộng rãi trong việc phát triển phần mềm mã nguồn mở, định nghĩa các điều kiện cho việc sử dụng, sửa đổi và phân phối phần mềm.
* **Cộng đồng phát triển :** là một nhóm các nhà phát triển và người dùng đóng góp vào việc phát triển phần mềm mã nguồn mở.
* **Repository :** là một kho lưu trữ các mã nguồn phần mềm, cho phép người sử dụng truy cập, sao chép và sửa đổi mã nguồn.
* **Forking :** là quá trình sao chép mã nguồn phần mềm từ một kho chứa mã nguồn gốc, tạo ra một phiên bản độc lập mới của phần mềm.
* **Commit :** là hành động ghi nhận các thay đổi đã được thực hiện vào mã nguồn phần mềm. Các commit được lưu trữ trong repository và cho phép người khác có thể xem lại quá trình phát triển phần mềm.
* **Pull request :** là quá trình người sử dụng đề xuất thay đổi mã nguồn phần mềm, sau đó gửi cho các nhà phát triển chính để xem xét và chấp nhận thay đổi đó.

**1.1. Mã nguồn mở**

**a) Khái niệm**

Phần mềm mã nguồn mở (Open source software - OSS) là những phần mềm có mã nguồn (source) được công khai, cho phép bất cứ ai cũng có thể chỉnh sửa, thay đổi hay sử dụng mã nguồn này để phát triển ra các phần mềm khác. Không chỉ miễn phí về giá mua mà còn miễn phí về bản quyền, người dùng được tùy ý sao chép và công khai nghiên cứu, làm việc mà không cần phải xin phép ai, điều mà không được phép đối với phần mềm mã nguồn đóng (Phần mềm thương mại).

Tiện ích mà phần mềm mã nguồn mở mang lại chính là quyền tự do sử dụng chương trình cho mọi mục đích, quyền tự do để nghiên cứu cấu trúc của chương trình, chỉnh sữa phù hợp với nhu cầu, truy cập vào mã nguồn, quyền tự do phân phối lại các phiên bản cho nhiều người, quyền tự do cải tiến chương trình và phát hành những bản cải tiến vì mục đích công cộng.

***\* Sự khác biệt giữa phần mềm mã nguồn mở và các loại phần mềm khác :***

Để phân định giữa phần mềm mã nguồn mở với các loại phần mềm khác cần dựa trên tính công khai của mã nguồn do lập trình viên/đơn vị sáng tạo ra quy định.

Nếu phần mềm mã nguồn mở công khai bộ mã nguồn cho mọi người cùng phân tích, sao chép và chỉnh sửa thì phần mềm mã nguồn đóng (độc quyền) lại ngược lại. Những phần mềm này chỉ cho phép những người đã tạo ra mới có quyền kiểm soát, bao gồm các thác tác truy cập, tìm lỗi, chỉnh sửa hay nâng cấp. Để sử dụng phần mềm độc quyền, người dùng phải đồng ý cam kết không tác động lên phần mềm ngoài phạm vi cho phép.

Phần mềm mã nguồn mở không có bảo hành như mã nguồn đóng. Do đó, nếu gặp vấn đề kỹ thuật trong khi sử dụng cũng sẽ không được hỗ trợ khắc phụ. Phần mềm mã nguồn mở cũng phải đăng ký, được quy định tại các đơn vị quy chuẩn giấy phép mã nguồn mở phổ biến như ***Apache License, BSD license, GNU General Public License, GNU Lesser General Public License, MIT License,…***

**b) Ưu - nhược điểm**

❖ Ưu điểm :

**Khả năng quản trị cao :** Đây là lý do đầu tiên khiến phần mềm mã nguồn mở được ưa chuộng nhiều, nhờ chức năng quản lý, kiểm soát sản phẩm. Đặc biệt, họ có thể cho phép phần nào hoạt động, phần nào không. Còn trên phương diện khách hàng, mặc dù không biết nhiều về khái niệm về nó, nhưng họ lại được hưởng vô số lợi ích từ phần mềm này.

**Gia tăng tư duy sáng tạo :** Vì sở hữu thiết kế dưới dạng mở, nên nhiều người có thể nghiên cứu nhằm tạo ra những ứng dụng hoàn thiện hơn. Đây có thể xem như một sản phẩm thách thức nhiều lập trình viên trong quá trình thực hiện dự án của mình.

**Chức năng tiện ích :** Nó cho phép Developer cùng làm việc trên một phần mềm nguồn mở mà không cần xin phép từ các tác giả gốc. Vậy nên, họ có quyền sửa chữa, cập nhật, nâng cấp nó nhanh hơn các phần mềm có bản quyền. Nói cách khác, khi có cả một cộng đồng cùng tập trung "chăm sóc" một sản phẩm thì chắc chắn, nó sẽ cải tiến và sở hữu nhiều chức năng chuyên nghiệp hơn.

**Tính ổn định :** Thông thường, phần mềm mã nguồn mở sở hữu độ đáng tin cậy cao. Đó là kết luận từ những cuộc phân tích, đánh giá và so sánh với các ứng dụng nguồn đóng.

❖ Nhược điểm :

**Tốc độ :** Việc thiết kế web bằng mã nguồn mở tương tự như chúng ta sử dụng 1 phần chức năng nhưng lại phải đưa lên 3 đến 4 phần mã nguồn, vì lý do một phần chung có thể cải tạo ra vô số loại website khác nhau. Trong khi đó, trang web xây dựng bằng tay sẽ được lập trình viên hướng đến một mục đích cụ thể, giúp giảm thiểu dư thừa code, nặng nề cho website.

**Khả năng bảo mật kém :** Do được chia sẻ rộng rãi trên mạng, nên bất cứ ai truy cập Internet đều có khả năng download về để tìm hiểu chúng. Điều này đồng nghĩa với mọi hacker cũng dễ dàng xem, đánh cắp dữ liệu hay làm gián đoạn quá trình hoạt động của trang web.

**Dịch vụ hỗ trợ hạn hẹp :** Vì được viết sẵn bởi những Developer tình nguyện nước ngoài, nên công ty cung cấp không thể nắm rõ tất cả chi tiết của website. Do đó, nếu xảy ra tình trạng lỗi hoặc muốn nâng cấp thêm chức năng… thì đối với nhà cung cấp, đây là điều vô cùng khó khăn.

**Thiếu tính độc quyền :** Phương pháp xây dựng này được gọi là cài đặt web chứ không phải là lập trình hay thiết kế. Bởi không cần kiến thức chuyên môn bạn cũng có thể làm ra được. Do vậy, tình trạng nhiều người cùng sở hữu một kiểu dáng, chức năng và bố cục website là điều không thể tránh khỏi.

**c) Các ví dụ**

Các hệ điều hành Linux, Ubuntu…; Các ngôn ngữ lập trình Perl, Python…; Hệ quản trị máy chủ Web (Web Server) Apache TomCat; Các hệ quản trị cơ sở dữ liệu (CSDL) quan hệ MySQL, PostgreSQL…

Trong hoạt động thư viện cũng có nhiều phần mềm mã nguồn mở. Các phần mềm mã nguồn mở đã và đang được sử dụng ở nước ta là:

- Phần mềm tư liệu CDS/ISIS, CSD/ISIS for Windows.

- Các phần mềm quản lý các bộ sưu tập số Greenstone, Dspace.

- Phần mềm quản trị thư viện tích hợp Koha.

**1.2. Mã nguồn đóng**

**a) Khái niệm**

Mã nguồn đóng là một dạng phần mềm được tạo ra bởi một mã nguồn độc quyền, chúng được bảo vệ chặt chẽ bởi một đơn vị chủ quản. Chỉ có những người xây dựng nên mã nguồn đóng mới được phép truy cập vào để thực hiện các công việc như sao chép, chỉnh sửa hay thậm chí là thay đổi các tính năng có trong chương trình. Muốn sử dụng các chương trình mã nguồn đóng, bạn chỉ có một cách là mua bản quyền của nó từ đơn vị phân phối chính thức.

***Những lý do nên chọn sử dụng mã nguồn đóng :***

* Sản phẩm được xây dựng dựa trên mã nguồn đóng sẽ được kiểm soát về tốc độ cập nhật các phiên bản mới.
* Do được thiết kế riêng bởi một đơn vị nhất định, nên chương trình sẽ có cơ chế bảo mật riêng và luôn đảm bảo an toàn cho những dữ liệu lưu trữ bên trong nó.
* Thuộc phạm vị quản lý của một đơn vị chủ quản, hạn chế tốt được sự xâm nhập cũng như các cuộc tấn công DoS làm ảnh hưởng đến hoạt động.
* Chương trình sẽ được xây dựng và phát triển dựa trên những yêu cầu của người dùng, nên sẽ không xuất hiện các tính năng dư thừa.
* Trong quá trình sử dụng sản phẩm, quý khách hàng sẽ luôn nhận được sự hỗ trợ nhiệt tình từ đơn vị cung cấp.
* Thể hiện rõ được những nét riêng của doanh nghiệp thông qua sản phảm, việc này sẽ gia tăng được lợi thế cạnh tranh cho doanh nghiệp so với các đơn vị khác cùng lĩnh vực hoạt động.

***Đặc điểm khi sử dụng mã nguồn đóng :***

* Không có chi phí nhất định, mà nó phụ thuộc phần lớn vào mức độ phức tạp, cùng những tính năng có trong phần mềm.
* Chương trình được xây dựng dựa trên mã nguồn đóng chỉ có thể được chỉnh sửa bởi nhà cung cấp.
* Trong quá trình sử dụng, nếu có bất kỳ trục trặc nao xảy ra bạn sẽ nhanh chóng nhận được sự hỗ trợ nhiệt tình từ đội ngũ support của đơn vị cung cấp.
* Khi sử dụng chương trình mã nguồn đóng, bạn không thể thay đổi bất kỳ yếu tố nào hoặc xem các đoạn code bên trong mã nguồn. Mặc dù vậy, bạn sẽ sở hữu được cho mình một phần mềm mã nguồn đáng tin cậy, cùng cơ chế bảo mật cao hơn.
* Tính ứng dụng của mã nguồn đóng luôn là một lợi thế đặc biệt theo đánh giá của nhiều người đã từng có cơ hội sử dụng qua. Bởi tài liệu hướng dẫn dành cho mã nguồn mở luôn được viết một cách đầy đủ và khá chi tiết.
* Các chương trình sẽ luôn được update lên các phiên bản mới nhanh chóng, nhưng phải tuân thủ theo nguyên tắc của nhà phát triển đưa ra.

**b) Ưu - nhược điểm**

❖ Ưu điểm :

* Website được trang bị tính năng bảo mật khá cao, nhằm hạn chế sự tấn công DoS hay của các hacker, tin tặc, đảm bảo cho các dữ liệu lưu trữ trên hệ thống được an toàn.
* Sở hữu được những nét đặc trưng riêng của doanh nghiệp, dễ dàng hơn trong việc thu hút người dùng. Đồng thời, nâng cao sự chuyên nghiệp, cũng như giá trị hình ảnh của doanh nghiệp trong mắt các đối tác và khách hàng.
* Mang lại lợi nhuận ổn định cho các nhà phát triển, tạo nguồn động lực cho họ tạo ra thêm nhiều các website được ứng dụng những công nghệ hiện đại và tiên tiến.
* Thân thiện với mọi người dùng, đặc biệt còn cung cấp sẵn nhiều dịch vụ hỗ trợ và hướng dẫn sử dụng khác nhau, nhằm hỗ trợ nâng cao trải nghiệm người dùng.
* Các website sử dụng mã nguồn đóng sẽ luôn nhận được sự hỗ trợ tận tình và nhanh chóng từ nhà phát triển, khi có bất kỳ sự cố xảy ra liên quan đến yếu tố kỹ thuật.
* Dễ dàng tùy chỉnh hay thêm, bớt các tính năng trong website, để đáp ứng tối đa nhu cầu sử dụng.
* Các website sẽ liên tục được update lên các phiên bản tiên tiến hơn, mang đến cho người dùng những trải nghiệm vô cùng tuyệt vời.

❖ Nhược điểm :

* Chi phí xây dựng và phát triển tương đối cao.
* Việc sửa chữa hay nâng cấp phụ thuộc rất lớn vào đơn vị cung cấp, người bên ngoài không thể can thiệp vào source code, trừ khi nhận được sự cho phép.
* Do phải phụ thuộc vào đơn vị cung cấp, nên tiến độ thực hiện mỗi khi gặp sự cố có thể sẽ phải chờ đợi khá lâu hoặc có thể phải bỏ thêm một khoản chi phí thì mới được khắc phục.
* Một số trường hợp website của bạn sẽ bị giới hạn người dùng truy cập, gây ảnh hưởng không nhỏ đến hoạt động kinh doanh của doanh nghiệp.

**c) Các ví dụ**

* **Microsoft Windows :** Hệ điều hành phổ biến nhất trên thế giới được phát triển và phân phối bởi Microsoft Corp., với mã nguồn đóng.
* **Adobe Photoshop :** Phần mềm chỉnh sửa hình ảnh chuyên nghiệp, được phát triển và phân phối bởi Adobe Systems Inc., với mã nguồn đóng.
* **Oracle Database :** Phần mềm hệ thống quản trị cơ sở dữ liệu mạnh mẽ, được phát triển và phân phối bởi Oracle Corporation, với mã nguồn đóng.
* **SAP Business Suite :** Phần mềm quản lý doanh nghiệp và tài chính, được phát triển và phân phối bởi SAP SE, với mã nguồn đóng.
* **AutoCAD :** Phần mềm thiết kế vẽ kỹ thuật, được phát triển và phân phối bởi Autodesk, với mã nguồn đóng.

**2. Giấy phép GPL v 2.0, 3.0**

Giấy phép GPL (General Public License): là một trong những giấy phép được sử dụng rộng rãi trong việc phát triển phần mềm mã nguồn mở, định nghĩa các điều kiện cho việc sử dụng, sửa đổi và phân phối phần mềm.

**Sao chép phần mềm:**

* Được sao chép và phân phối chương trình, được yêu cầu trả phí cho việc phân phối đó .
* Khi sao chép và phân phối chương trình, phải đính kèm các thông báo về bản quyền gốc và không bảo hành( trừ trường hợp có văn bản thêm về quy định bảo hành).

**Phân phối phân mềm:**

* Được phân phối bản đã được thay đổi đó.
* Khi phân phối bản đã được thay đổi bởi bản thân , phải chú thích rõ đó là bản đã được thay đổi, các thành phần được thay đổi, và áp dụng giấy phép GNU cho bản đã được thay đổi đó.

**Tính phí phân phối cho người dùng:**

* Bạn có thể tính phí khi phân phối phần mềm cho người dùng tuy nhiên cần phải đưa cho họ bản sao của GNU-GPL để cho ho biết rằng phần mềm này được lấy từ đâu.

**Thay đổi phần mềm:**

* Được thay đổi chương trình để sử dụng cho mục đích cá nhân .
* Khi phát hành chương trình phải công khai mã nguồn của chương trình, đồng thời phải công bố mã nguồn của chương trình trong tối thiểu 3 năm mà không được đòi một khoản phí nào từ những người yêu cầu mã nguồn trừ chi phí vận chuyển hay tương đương.

**3. OSI ( Open Source Initiative - www.opensource.org )**

Open Source Initiative (OSI) là một tổ chức phi lợi nhuận dành riêng trong việc thúc đẩy phần mềm nguồn mở, hoạt động quan trọng nhất là xem xét phế duyệt giấy phép mã nguồn mở, họ muốn tạo ra một tổ chức để thúc đẩy và phối hợp sự phát triển của phần mềm mã nguồn mở và do đó thành lập OSI.

OSI đang tích cực tham gia vào việc xây dựng cộng đồng mã nguồn mở, vận động cộng đồng, giáo dục và nâng cao nhận thức về tầm quan trọng của phi phần mềm mã nguồn -proprietary hoặc mở. Để thiết lập một môi trường mã nguồn mở trên toàn thế giới, bảo tồn và hỗ trợ Định nghĩa mã nguồn mở OSI và cũng cung cấp các chứng nhận OSI-Certified Phần mềm nguồn mở chương trình. Để đạt được chứng nhận OSI này, phần mềm sẽ được phân phối sử dụng một giấy phép để đảm bảo quyền hợp pháp để tự do đọc, sử dụng, sửa đổi và tái phân phối phần mềm.

**4. Phần mềm tự do ( Libre sofware )**

**a) Khái niệm**

Phần mềm tự do (tiếng Anh: free software hay libre software) là phần mềm máy tính được phân phối theo các điều khoản cho phép người dùng chạy phần mềm cho bất kỳ mục đích nào cũng như nghiên cứu, thay đổi và phân phối phần mềm cũng như mọi phiên bản phù hợp. Có thể có hoặc không có mã nguồn, mang nghĩa tự do ( không hẳn miễn phí ), được sử dụng, sao chép, phân phối, sửa đổi ( nếu có mã nguồn ).

Phần mềm tự do tập trung vào vấn đề của sự tự do, chứ không phải là giá cả. Người dùng, độc lập hay hợp tác với các lập trình viên máy tính, được tự do làm những gì họ muốn với các bản sao của một phần mềm tự do (bao gồm cả lợi nhuận từ chúng) bất kể được trả bao nhiêu để có được chương trình. Các chương trình máy tính được coi là tự do nếu chúng cung cấp cho người dùng (không chỉ nhà phát triển) quyền kiểm soát cuối cùng đối với phần mềm và sau đó, trên các thiết bị của họ.

❖ Định nghĩa về 4 tự do :

* **Freedom 0:** Tự do ‘chạy’ chương trình cho bất kỳ mục đích nào.
* **Freedom 1:** Tự do ‘nghiên cứu’ cách thức chương trình hoạt động và thay đổi nó để khiến chương trình thực hiện những gì bạn muốn.
* **Freedom 2:** Tự do ‘phân phối lại’ ‘và tạo các bản sao để bạn có thể giúp đỡ hàng xóm của mình.
* **Freedom 3:** Quyền tự do ‘cải thiện’ chương trình và phát hành các cải tiến của bạn (và các phiên bản sửa đổi nói chung) cho công chúng, để toàn bộ cộng đồng được hưởng lợi.

**b) Ưu - nhược điểm**

❖ Ưu điểm :

* **Sự linh hoạt và tùy biến :** Phần mềm tự do cho phép người dùng thực hiện bất kỳ sửa đổi và tùy chỉnh nào vào mã nguồn của nó, giúp người dùng có thể tùy biến và thích nghi với nhu cầu của mình.
* **Không phải trả chi phí :** Phần mềm tự do được miễn phí để sử dụng, tái phân phối và sửa đổi, người dùng không phải trả cho bản quyền hoặc giấy phép sử dụng.
* **Tính đa dạng :** Có rất nhiều phần mềm tự do có thể được sử dụng cho nhiều mục đích khác nhau, từ hệ điều hành đến ứng dụng chuyên biệt.
* **Tự do phần mềm :** Phần mềm tự do đảm bảo tự do cho người dùng, cho phép họ kiểm soát, sử dụng và phân phối phần mềm một cách tự do.

❖ Nhược điểm :

* **Sự linh hoạt và tùy biến :** Phần mềm tự do cho phép người dùng thực hiện bất kỳ sửa đổi và tùy chỉnh nào vào mã nguồn của nó, giúp người dùng có thể tùy biến và thích nghi với nhu cầu của mình.
* **Không phải trả chi phí :** Phần mềm tự do được miễn phí để sử dụng, tái phân phối và sửa đổi, người dùng không phải trả cho bản quyền hoặc giấy phép sử dụng.
* **Tính đa dạng :** Có rất nhiều phần mềm tự do có thể được sử dụng cho nhiều mục đích khác nhau, từ hệ điều hành đến ứng dụng chuyên biệt.
* **Tự do phần mềm :** Phần mềm tự do đảm bảo tự do cho người dùng, cho phép họ kiểm soát, sử dụng và phân phối phần mềm một cách tự do.

**c) Các ví dụ**

* **Linux:** Hệ điều hành mã nguồn mở và tự do phổ biến nhất thế giới, được đóng góp và phát triển bởi hàng ngàn nhà phát triển trên toàn cầu.
* **Apache:** Phần mềm máy chủ web mã nguồn mở, điều khiển hơn nửa số trang web trên toàn thế giới.
* **LibreOffice:** Bộ ứng dụng văn phòng miễn phí và mở, bao gồm các ứng dụng như văn bản, bảng tính, trình chỉnh sửa slide và nhiều hơn nữa.
* **GIMP:** Phần mềm chỉnh sửa hình ảnh tự do chuyên nghiệp và mở.
* **Firefox:** Trình duyệt web miễn phí, an toàn và thân thiện với người dùng.
* **Blender:** Phần mềm đồ họa 3D tự do và chuyên nghiệp, được sử dụng trong các bộ phim và trò chơi hàng đầu.
* **OpenOffice:** Bộ ứng dụng văn phòng mở và miễn phí, hỗ trợ định dạng tài liệu Microsoft Office và nhiều hơn nữa.
* **Moodle:** Hệ thống quản lý học tập trực tuyến, miễn phí cho các giáo viên và trường học.
* **WordPress:** Hệ thống quản lý nội dung web miễn phí và mở, được sử dụng cho hầu hết các trang web và blog trên toàn thế giới.
* **Audacity:** Phần mềm chỉnh sửa âm thanh miễn phí và mở, được sử dụng để chỉnh sửa và phát triển các tệp âm thanh và video.

**5. Phần mềm miễn phí ( Freeware )**

**a) Khái niệm**

Là phần mềm miễn phí có thể được sử dụng, sao chép, phân phối và sửa đổi mà không yêu cầu tính phí cho bất kỳ mục đích nào. Phần mềm miễn phí này có thể được tìm thấy trên internet hoặc bất kỳ đĩa mềm, ổ đĩa USB hoặc đĩa CD có chứa chúng. Một số phần mềm miễn phí có thể không cung cấp cho người dùng tất cả các tính năng hoặc chức năng của các phiên bản trả tiền và thường có các giới hạn sử dụng, nhưng vẫn được xem là miễn phí để sử dụng.

Phần mềm miễn phí có nghĩa là không có giấy phép trả tiền cần thiết để sử dụng ứng dụng, không có phí hoặc đóng góp cần thiết, không có giới hạn về số lần bạn có thể tải xuống hoặc mở chương trình và không có ngày hết hạn.

Tuy nhiên, phần mềm miễn phí vẫn có thể bị hạn chế theo một số cách. Phần mềm tự do, mặt khác, là hoàn toàn và hoàn toàn vô hiệu hóa các hạn chế và cho phép người dùng làm hoàn toàn bất cứ điều gì họ muốn với chương trình.

**b) Ưu - nhược điểm**

❖ Ưu điểm :

* **Miễn phí :** Đây là lợi ích lớn nhất của phần mềm miễn phí. Người dùng không phải trả tiền để sử dụng chúng.
* **Thuận tiện :** Phần mềm miễn phí thường có dung lượng nhỏ, dễ cài đặt và có sẵn trên internet. Người dùng không cần phải mất nhiều thời gian và tiền bạc để mua và tải xuống.
* **Đa dạng :** Có rất nhiều loại phần mềm miễn phí cho các ngành công nghiệp khác nhau, có nghĩa là người dùng có nhiều tùy chọn để lựa chọn.

❖ Nhược điểm :

* **Giới hạn tính năng:** Các phiên bản miễn phí của phần mềm thường có tính năng giới hạn so với phiên bản trả tiền.
* **Quảng cáo:** Một số phần mềm miễn phí có chứa quảng cáo, dẫn đến trải nghiệm người dùng không được tốt.
* **Không có hỗ trợ:** Không có hỗ trợ khách hàng miễn phí và các phiên bản miễn phí thường không được cập nhật thường xuyên như các phiên bản trả tiền.

**c) Các ví dụ**

* **7-Zip:** là một chương trình nén và giải nén dữ liệu miễn phí được phát triển cho Windows.
* **VLC Media Player:** là một phần mềm trình phát đa phương tiện miễn phí và mã nguồn mở.
* **LibreOffice:** là một bộ ứng dụng văn phòng miễn phí và mã nguồn mở có tính năng tương tự như Microsoft Office.
* **GIMP:** là một phần mềm chỉnh sửa ảnh miễn phí và mã nguồn mở, được sử dụng như một thay thế cho Adobe Photoshop.
* **Blender:** là một chương trình tạo và chỉnh sửa hình ảnh, đồ họa 3D và video miễn phí có các tính năng như dựng hình, ánh sáng, vật liệu và hơn thế nữa.
* **Mozilla Firefox:** là một trình duyệt web miễn phí và mã nguồn mở được sử dụng nhiều nhất trên thế giới.

**6. Phần mềm trả một phần ( Shareware )**

**a) Khái niệm**

Phần mềm trả một phần (Shareware) là loại phần mềm có bản quyền đặt giới hạn sử dụng nhưng cho phép người dùng sử dụng miễn phí trong một khoảng thời gian nhất định và sau đó yêu cầu người dùng trả tiền để tiếp tục sử dụng phần mềm. Loại phần mềm này thường được phân phối qua các kênh truyền thông như CD, DVD, tải xuống hoặc bản sao lưu. Người dùng có thể tải xuống chương trình và sử dụng miễn phí trong một khoảng thời gian nhất định trước khi quyết định mua bản quyền nếu họ muốn tiếp tục sử dụng tiếp.

**b) Ưu - nhược điểm**

❖ Ưu điểm :

* **Dễ sử dụng :** Phần mềm shareware có thể dễ dàng tải xuống và cài đặt so với phần mềm trả tiền khác.
* **Thử nghiệm miễn phí :** Người dùng có thể tải và sử dụng phần mềm một thời gian nhất định miễn phí để đánh giá tính năng.
* **Giá cả hợp lý :** Giá của phần mềm shareware thường rẻ hơn các phần mềm trả tiền khác.
* **Hỗ trợ khách hàng tốt :** Người dùng có thể nhận được hỗ trợ khách hàng từ nhà phát triển hoặc người bán phần mềm.

❖ Nhược điểm :

* **Hạn chế tính năng :** Ảnh hưởng đến trải nghiệm người dùng với các tính năng được giới hạn trong phần mềm tải về miễn phí.
* **Quảng bá :** Một số phần mềm shareware có thể có quảng bá và spam nhắm vào khách hàng.
* **Phiền toái :** Người dùng có thể mất nhiều thời gian để tìm kiếm bản phần mềm phù hợp và yêu cầu phải tìm bản quyền, trả tiền để sử dụng phần mềm nếu muốn tiếp tục sử dụng sau khi thời gian trải nghiệm miễn phí kết thúc.

**c) Các ví dụ**

* **WinRAR :** là một công cụ nén và giải nén tập tin dữ liệu. Bản trả phí của phần mềm này cung cấp một số tính năng nâng cao.
* **IDM - Internet Download Manager :** là một trình quản lý tải xuống được sử dụng phổ biến để tải xuống các tập tin từ Internet. Người dùng có thể thử nghiệm miễn phí trong một khoảng thời gian nhất định trước khi quyết định mua bản quyền.
* **Camtasia :** là một phần mềm chỉnh sửa video và quay màn hình. Người dùng có thể tải xuống phiên bản miễn phí thử nghiệm trong 30 ngày trước khi phải mua bản quyền để tiếp tục sử dụng phần mềm.
* **WinZip :** là một chương trình nén và giải nén dữ liệu. Bản phần mềm trả phí của nó cung cấp các tính năng nâng cao và hỗ trợ nén các tập tin lớn hơn.
* **Snagit :** là một công cụ chụp ảnh và quay màn hình. Bản trả phí của phần mềm này có các tính năng nâng cao như quay phim trên màn hình và chụp ảnh tùy chỉnh.

**7. Phần mềm thương mại ( Commercial software )**

**a) Khái niệm**

Phần mềm thương mại (Commercial software) là loại phần mềm được công ty hoặc nhà phát triển phân phối với mục đích hướng đến lợi nhuận kinh doanh. Phần mềm thương mại được bán với giá tiền hoặc phí sử dụng để đáp ứng cho việc phát triển, cập nhật và hỗ trợ phần mềm. Người mua phần mềm thương mại sẽ được cấp giấy phép sử dụng với các điều khoản được quy định trong hợp đồng giữa người bán và người mua. Loại phần mềm này được sử dụng phổ biến trong các doanh nghiệp và tổ chức.

**b) Ưu - nhược điểm**

❖ Ưu điểm :

* **Đảm bảo chất lượng vào sản phẩm :** Do phần mềm được phát triển và phân phối với mục đích hướng đến lợi nhuận kinh doanh, các nhà phát triển cần đảm bảo sản phẩm có chất lượng cao để thu hút người mua.
* **Hỗ trợ khách hàng tốt :** Do lợi nhuận được lấy từ phần mềm thương mại, các nhà phát triển phải cung cấp mức hỗ trợ tốt cho khách hàng, bao gồm cấp phép sử dụng và hỗ trợ kỹ thuật.
* **Dễ sử dụng :** Phần mềm thương mại thường được thiết kế với giao diện và trải nghiệm người dùng tốt để thuận tiện cho khách hàng sử dụng.
* **Tính ổn định** : Phần mềm thương mại thường được kiểm tra và đảm bảo tính ổn định trước khi phát hành cho người dùng.

❖ Nhược điểm :

* **Giá cả cao :** Giá của phần mềm thương mại thường cao hơn các loại phần mềm khác, làm cho nó khó tiếp cận với một số người dùng.
* **Hạn chế tính năng :** Một số phần mềm thương mại có thể bị giới hạn các tính năng nếu không mua bản quyền sử dụng.
* **Bảo mật :** Vì phần mềm thương mại thường chứa thông tin quan trọng, điều này có thể dẫn đến các vấn đề về bảo mật nếu không được quản lý đúng cách.
* **Phụ thuộc vào nhà phát triển :** Người dùng phụ thuộc vào nhà phát triển cho việc cập nhật và hỗ trợ phần mềm, nếu nhà phát triển không cung cấp đầy đủ những yêu cầu này thì khách hàng sẽ phải chịu thiệt hại.

**c) Các ví dụ**

* **Microsoft Office :** gồm các ứng dụng phổ biến như Microsoft Word, Excel, PowerPoint, Outlook, và những phần mềm khác, được sử dụng trong các tổ chức và doanh nghiệp.
* **Adobe Creative Suite :** bao gồm các phần mềm thiết kế đồ họa và xuất bản như Adobe Photoshop, Illustrator, InDesign, Acrobat và Premiere Pro, phổ biến trong các công ty thiết kế.
* **AutoCAD :** là một phần mềm vẽ kỹ thuật và thiết kế đồ họa phục vụ cho việc công nghiệp và xây dựng.
* **QuickBooks :** là một phần mềm kế toán, quản lý tài chính được sử dụng để quản lý doanh nghiệp nhỏ và vừa.
* **Salesforce :** là một phần mềm quản lý mối quan hệ khách hàng (CRM) được sử dụng trong nhiều tổ chức và doanh nghiệp để giúp họ quản lý thông tin khách hàng và dữ liệu kinh doanh.
* **Oracle :** là một hệ thống quản trị cơ sở dữ liệu được sử dụng để quản lý dữ liệu và cung cấp dịch vụ lưu trữ dữ liệu cho các doanh nghiệp.

**8. Phần mềm miễn phí dạng dịch vụ ( Free and open-source software as a service - FOSSaaS )**

**a) Khái niệm**

Phần mềm miễn phí dạng dịch vụ là một loại phần mềm miễn phí được cung cấp trên nền tảng trực tuyến mà người dùng có thể truy cập thông qua trình duyệt web hoặc ứng dụng di động. FOSSaaS thường được phát hành theo các giấy phép mã nguồn mở và cho phép người dùng sử dụng, chỉnh sửa, sao chép và phân phối lại mà không cần trả bất kỳ phí sử dụng nào. FOSSaaS được yêu thích bởi vì nó không yêu cầu người dùng phải cài đặt phần mềm trên máy tính và có thể truy cập trực tiếp từ trình duyệt web hoặc ứng dụng di động. FOSSaaS có thể được sử dụng trong nhiều mục đích, bao gồm quản lý dự án, quản lý thông tin cá nhân, tư vấn tài chính, quản lý mối quan hệ khách hàng và nhiều hơn nữa.

**b) Ưu - nhược điểm**

❖ Ưu điểm :

* Phần mềm được cung cấp miễn phí và mã nguồn mở cho người dùng, giúp giảm thiểu chi phí mà người dùng phải chi trả cho giấy phép sử dụng phần mềm.
* FOSSaaS cho phép các nhà phát triển tùy chỉnh và sửa đổi phần mềm theo nhu cầu của họ, tạo ra các phiên bản mới và cải tiến phần mềm, từ đó giúp nâng cao tính năng và hiệu suất của nó.
* FOSSaaS thường có lượng người dùng và cộng đồng sử dụng lớn, giúp giải quyết các vấn đề và sự cố phát sinh nhanh chóng, đồng thời tạo ra các trang web và diễn đàn trợ giúp để cung cấp thông tin hữu ích cho người dùng.

❖ Nhược điểm :

* FOSSaaS thường có tính năng và hiệu suất khá giới hạn so với các phần mềm SaaS trả phí, bởi vì họ không có nguồn lực để phát triển và cập nhật phần mềm tốt hơn.
* Các tính năng và các ứng dụng không bao giờ được cung cấp với chất lượng cao như các sản phẩm dạng SaaS chuyên nghiệp.
* Không có sự đảm bảo về hiệu suất và tính bảo mật của FOSSaaS, làm cho nó không phù hợp cho các doanh nghiệp phải đối mặt với các tài sản quy mô lớn hoặc các lĩnh vực nhạy cảm.

**c) Các ví dụ**

* **Office Suite :** Cung cấp cho người dùng các ứng dụng văn phòng trực tuyến miễn phí, ví dụ như xử lý văn bản, trình chiếu, bảng tính và nhiều ứng dụng khác.
* **Bitwarden :** Là một trình quản lý mật khẩu trực tuyến mã nguồn mở, cho phép người dùng lưu trữ và quản lý thông tin đăng nhập của họ một cách an toàn.
* **WordPress :** Cung cấp nền tảng tạo và quản lý website miễn phí, mã nguồn mở và cung cấp hàng nghìn các plugin và chủ đề tùy chỉnh khác nhau.
* **Nextcloud :** Dịch vụ lưu trữ trực tuyến miễn phí, cung cấp cho người dùng các tính năng như lưu trữ tệp, chia sẻ tệp, liên lạc đồng nghiệp, và nhiều hơn nữa.
* **Turtl :** Là một ứng dụng ghi chú trực tuyến miễn phí, cung cấp cho người dùng khả năng tạo và chỉnh sửa các ghi chú cá nhân của họ và chia sẻ chúng với người dùng khác.

**9. Phần mềm dạng dịch vụ ( Software as a Service - SaaS )**

**a) Khái niệm**

SaaS là viết tắt của Software as a Service (Phần mềm dưới dạng Dịch vụ). Đây là một mô hình kinh doanh phần mềm, trong đó ứng dụng phần mềm được cung cấp cho người dùng thông qua các trang web và các ứng dụng trực tuyến. Thay vì phải cài đặt và sử dụng phần mềm trên máy tính của mình, người dùng có thể truy cập nó thông qua internet bằng cách đăng nhập vào tài khoản của mình trên trang web hoặc ứng dụng đó.

Với SaaS, người dùng có thể truy cập các phiên bản mới nhất của phần mềm và được cập nhật tự động. Họ chỉ cần trả tiền cho một khoản phí theo dịch vụ hoặc theo người dùng, thay vì phải mua giấy phép sử dụng phần mềm và trả phí bảo trì hằng năm.

**b) Ưu - nhược điểm**

❖ Ưu điểm :

* Không yêu cầu sử dụng tài nguyên phần cứng và không gian lưu trữ trên máy tính nên không phải đầu tư chi phí cao để mua phần cứng để lưu trữ và giám sát, quản lý.
* Người dùng có thể truy cập vào các tiện ích của phần mềm mọi lúc, mọi nơi thông qua internet và bất kỳ thiết bị nào cũng được hỗ trợ.
* Không cần phải mua giấy phép sử dụng phần mềm, quản lý và cập nhật phần mềm bởi vì tất cả đều được thực hiện bởi nhà cung cấp SaaS.
* Tính linh hoạt cao, người dùng có thể dễ dàng thay đổi giữa các dịch vụ khác nhau hoặc mở rộng quy mô sử dụng.

❖ Nhược điểm :

* Người dùng phải trả tiền cho việc sử dụng, điều này sẽ trở thành một khoản chi phí đối với những người dùng cá nhân hoặc những công ty nhỏ.
* Dữ liệu được lưu trữ ngoài đám mây và phụ thuộc vào kết nối internet, nếu có sự cố về kết nối internet hoặc bảo mật sẽ làm gián đoạn quá trình làm việc và lưu trữ dữ liệu.
* Vì những dịch vụ SaaS được điều chỉnh tốt nhất cho khách hàng là chung nên các kiểu tùy chỉnh lớn cần sự hỗ trợ để được giải quyết.

**c) Các ví dụ**

* **Google Workspace :** Cung cấp cho người dùng các ứng dụng văn phòng trực tuyến như Gmail, Google Drive, Google Docs, Google Sheets, Google Slides và Google Calendar.
* **Salesforce :** Là nền tảng quản lý quan hệ khách hàng trực tuyến, cung cấp cho các doanh nghiệp một bộ công cụ để quản lý khách hàng, tiếp thị, bán hàng và hỗ trợ.
* **Dropbox :** Là một dịch vụ lưu trữ trực tuyến, cho phép người dùng lưu và chia sẻ tệp trực tuyến với tính năng đồng bộ hóa lên đến nhiều thiết bị.
* **Slack :** Là một dịch vụ giao tiếp cộng tác nhóm trực tuyến, cung cấp cho người dùng một nền tảng để chia sẻ tệp tin và nhắn tin với đồng nghiệp.
* **Hubspot :** Là một nền tảng marketing tổng hợp, cung cấp cho doanh nghiệp rất nhiều công cụ bao gồm email marketing, quản lý khách hàng và marketing tổng quát.
* **Canva :** Là một ứng dụng thiết kế đồ họa trực tuyến, cung cấp cho người dùng các công cụ để tạo ra những tác phẩm đồ họa chuyên nghiệp và độc đáo.

**10. Mô hình phát triển OSS**

Mô hình phát triển OSS là một quy trình sản xuất phần mềm mã nguồn mở (open source software - OSS) được cộng đồng sử dụng và phát triển đồng thời. Mô hình này thường được sử dụng cho các dự án phát triển phần mềm mã nguồn mở như Linux, Apache, Firefox, MySQL, PHP, v.v.

***Mô hình phát triển OSS bao gồm các bước sau:***

1. Thu thập các yêu cầu và đề xuất tính năng mới từ người sử dụng và các nhà phát triển OSS trong cộng đồng.
2. Phân tích yêu cầu và đề xuất, và quyết định những tính năng nào sẽ được thêm vào trong phiên bản mới của phần mềm.
3. Phát triển mã nguồn của phần mềm bằng cách sử dụng các công nghệ và ngôn ngữ lập trình phù hợp.
4. Kiểm tra và thử nghiệm phần mềm bằng các phương pháp kiểm tra tự động và thử nghiệm thủ công.
5. Phát hành phiên bản mới của phần mềm với các tính năng mới được thêm vào, và giải quyết các lỗi được phát hiện bởi cộng đồng.
6. Xây dựng và bảo trì tài liệu hướng dẫn và hỗ trợ cho phần mềm để giúp người sử dụng và nhà phát triển OSS tiếp cận và hiểu về phần mềm.

Mô hình phát triển OSS có nhiều ưu điểm, bao gồm tính mở và linh hoạt, cho phép sự tham gia của cộng đồng, tăng tính đáng tin cậy của phần mềm và chi phí thấp hơn so với các phần mềm thương mại. Tuy nhiên, điều quan trọng là phải có một cộng đồng đông đảo và tích cực để phát triển phần mềm theo mô hình này.

**11. Ưu điểm và nhược điểm hệ thống mã nguồn mở**

❖ Ưu điểm :

* **Tính mở và linh hoạt :** mã nguồn được phát triển bởi một cộng đồng người dùng và phát hành dưới các giấy phép mã nguồn mở, cho phép các nhà phát triển tự do sửa đổi và tái sử dụng mã nguồn, và sử dụng trong các mục đích khác nhau.
* **Tính đáng tin cậy :** Do có sự kiểm tra và đóng góp từ cộng đồng người dùng sử dụng, nên hệ thống mở mã nguồn thường được xác định và sửa đổi nhanh chóng để giải quyết các lỗi và đảm bảo tính ổn định và độ tin cậy của phần mềm.
* **Chi phí thấp :** Do mã nguồn mở là miễn phí và có thể tái sử dụng cho nhiều mục đích khác nhau, vì vậy chi phí phát triển, triển khai và bảo trì có thể rẻ hơn nhiều so với phần mềm thương mại.
* **Tính tương thích :** Hệ thống mã nguồn mở thường có khả năng tương thích tốt hơn với các phần mềm khác hoặc các hệ thống khác, do được phát triển và sử dụng bởi cộng đồng người dùng.

❖ Nhược điểm :

* **Thiếu sự hỗ trợ :** Do không có một đơn vị cụ thể phát hành hoặc cung cấp phần mềm, nên việc tìm kiếm hỗ trợ hoặc tìm kiếm giải pháp khi phát sinh vấn đề có thể khó khăn hơn so với phần mềm thương mại.
* **Không đảm bảo tính bảo mật :** Mã nguồn mở thường có thể bị tấn công bởi các hacker và phần mềm độc hại, do nó dễ dàng truy cập và cách sử dụng phức tạp hơn so với phần mềm thương mại.
* **Không cập nhật thường xuyên :** Do được phát triển bởi cộng đồng người dùng, nên việc cập nhật phần mềm và các mô-đun liên quan có thể bị chậm trễ hoặc bị bỏ qua trong một số trường hợp.

**12. Cách khắc phục nhược điểm của hệ thống mã nguồn mở**

* Xây dựng cộng đồng lớn và tích cực: Cộng đồng người dùng và nhà phát triển mã nguồn mở càng lớn và tích cực, thì càng có nhiều động lực để viết và cập nhật mã nguồn. Việc phát triển một cộng đồng mang tính tích cực là yếu tố quan trọng đối với một dự án mã nguồn mở.
* Tạo ra hệ thống hỗ trợ như phần mềm thương mại: Thay vì chỉ dựa vào cộng đồng để hỗ trợ việc sử dụng và khắc phục lỗi, các nhà phát triển mã nguồn mở cần phải thiết kế và cung cấp các giải pháp hỗ trợ sử dụng và khắc phục lỗi cho người dùng, để tạo sự tin tưởng và thu hút người dùng.
* Quản lý mã nguồn mở tốt: Việc quản lý và duy trì phát triển mã nguồn mở là rất quan trọng. Các nhà phát triển cần đảm bảo mã nguồn của mình được theo dõi chặt chẽ, có cơ chế kiểm soát và đánh giá chất lượng mã nguồn, bảo vệ quyền sở hữu trí tuệ và tránh các lỗ hổng bảo mật.
* Tăng cường tính bảo mật: Các nhà phát triển mã nguồn mở cần tăng cường tính bảo mật cho phần mềm của mình, bao gồm sử dụng mã hóa, thiết lập và kiểm tra khả năng chống tấn công, cập nhật thường xuyên để khắc phục các lỗi bảo mật.
* Diversify nguồn tài trợ: Thay vì chỉ phải dựa vào tiền quà tặng các cá nhân ủng hộ, các nhà phát triển mã nguồn mở cần phải tìm nguồn tài trợ đa dạng để đủ sức phát triển theo thời gian.

**13. Giấy phép sử dụng phần mềm mã nguồn mở, lợi ích và các yếu tố tác động đến sự tự do phần mềm mã nguồn mở**

Giấy phép sử dụng phần mềm mã nguồn mở là một tài liệu pháp lý quy định quyền và nghĩa vụ của người dùng khi sử dụng phần mềm mã nguồn mở. Giấy phép này cũng được gọi là giấy phép mã nguồn mở hoặc giấy phép phần mềm tự do.

Giấy phép này định nghĩa các quyền và nghĩa vụ của người sử dụng phần mềm mã nguồn mở, cung cấp cho người sử dụng nhiều quyền hơn so với phần mềm thương mại có giá và đảm bảo tính mã nguồn mở của phần mềm.

**Một số giấy phép mã nguồn mở phổ biến nhất bao gồm:**

* **Giấy phép GPL (General Public License) :** Giấy phép này yêu cầu khi đóng góp và phân phối phần mềm, phải bảo đảm mã nguồn được công bố theo giấy phép GPL, bảo vệ quyền sở hữu trí tuệ của các tác giả ban đầu và khuyến khích tất cả người dùng đóng góp cho cộng đồng phát triển phần mềm mã nguồn mở.
* **Giấy phép LGPL (Lesser General Public License) :** Giấy phép này giống như GPL, nhưng cho phép các thư viện được sử dụng trong các ứng dụng thương mại.
* **Giấy phép Apache :** Giấy phép này cho phép người sử dụng sử dụng, sao chép, phân phối, sửa đổi và đóng góp cho mã nguồn. Giấy phép này vẫn bảo vệ quyền sở hữu trí tuệ của các tác giả ban đầu, nhưng cho phép các công ty thương mại tích hợp phần mềm mã nguồn mở vào sản phẩm của mình mà không phải công bố mã nguồn mở.
* **Giấy phép MIT :** Giấy phép này cho phép người sử dụng sử dụng, sao chép, phân phối, sửa đổi và đóng góp cho mã nguồn. Giấy phép này rất đơn giản và cho phép các công ty thương mại sử dụng phần mềm mã nguồn mở mà không cần công bố mã nguồn mở của họ.

❖ Lợi ích :

* **Khả năng sử dụng và phân phối :** Giấy phép mã nguồn mở cho phép người dùng sử dụng, sao chép, phân phối và thay đổi phần mềm, miễn là các điều khoản của giấy phép được tuân thủ. Điều này đảm bảo quyền tự do cho người dùng, cũng như khả năng phát triển và sử dụng phần mềm với mục đích thương mại.
* **Đảm bảo tính mã nguồn mở :** Giấy phép mã nguồn mở đảm bảo rằng mã nguồn của phần mềm mã nguồn mở sẽ luôn được công khai, vào bất kỳ thời điểm nào, người dùng có quyền sử dụng và thay đổi mã nguồn này.
* **Kiểm soát sở hữu trí tuệ :** Giấy phép mã nguồn mở định nghĩa các quyền và nghĩa vụ làm chủ của nhà phát triển phần mềm, bảo vệ quyền sở hữu trí tuệ của các tác giả ban đầu và đảm bảo rằng người sử dụng tuân thủ những quy định này.
* **Cảm hứng sáng tạo :** Giấy phép mã nguồn mở thúc đẩy sự cộng tác và đóng góp từ cộng đồng người dùng và nhà phát triển, tạo điều kiện để các học sinh, sinh viên và nhà nghiên cứu có thể tham gia vào các dự án phần mềm có tầm ảnh hưởng.
* **Tin cậy và hiệu quả :** Giấy phép mã nguồn mở cung cấp cho người dùng và doanh nghiệp một hệ sinh thái phần mềm tốt và đáng tin cậy, giúp tăng hiệu quả và giảm chi phí phát triển phần mềm.

❖ Các yếu tố tác động đến sự tự do phần mềm mã nguồn mở

* **Giấy phép sử dụng phần mềm :** Giấy phép mã nguồn mở quy định các điều kiện và giới hạn về sử dụng, sao chép, phân phối và thay đổi phần mềm mã nguồn mở. Các giấy phép này là yếu tố chính tác động đến sự tự do của phần mềm mã nguồn mở.
* **Sự phát triển và cộng tác từ cộng đồng :** Một trong những đặc điểm của phần mềm mã nguồn mở là sự phát triển và cộng tác từ cộng đồng người dùng rộng lớn. Vì vậy, sự tự do của phần mềm mã nguồn mở phụ thuộc vào mức độ sự tham gia và cộng tác của cộng đồng này.
* **Tính chất của mã nguồn mở :** Mã nguồn mở là yếu tố quan trọng cốt lõi của sự tự do của phần mềm mã nguồn mở. Vì mã nguồn luôn được công khai, người dùng có quyền sử dụng, sao chép, phân phối và thay đổi mã nguồn này.
* **Quyền sở hữu trí tuệ :** Quyền sở hữu trí tuệ của các nhà phát triển phần mềm mã nguồn mở cũng là yếu tố quan trọng tác động đến sự tự do của phần mềm mã nguồn mở. Các tác giả ban đầu đôi khi có quyền hạn chế việc sử dụng và phân phối phần mềm, do đó cần có các quy định và điều khoản rõ ràng về quyền sở hữu trí tuệ.
* **Điều kiện kinh doanh của các doanh nghiệp :** Nhiều doanh nghiệp sử dụng phần mềm mã nguồn mở và tích hợp vào các sản phẩm của họ. Tuy nhiên, việc kinh doanh và tài trợ phát triển của các doanh nghiệp này có thể tác động đến sự tự do của phần mềm mã nguồn mở.

**14. Các tiêu chí để phân loại phần mềm dựa trên giấy phép ( lisence )**

* **Giấy phép mã nguồn mở :** Đó là giấy phép cho phép người dùng sử dụng, sao chép, phân phối và sửa đổi mã nguồn của phần mềm. Giấy phép phổ biến bao gồm GNU GPL, MIT, BSD, Apache, Mozilla Public License.
* **Giấy phép bảo hộ sở hữu trí tuệ :** Đó là giấy phép mà các tác giả phần mềm cấp để bảo vệ quyền sở hữu trí tuệ của mình. Các giấy phép phổ biến bao gồm các dạng như GPL có điều khoản "copyleft", giấy phép bảo mật hình thức, và giấy phép thương mại.
* **Giấy phép công cộng :** Đó là giấy phép cho phép bất cứ ai sử dụng, sao chép hoặc phân phối phần mềm. Tuy nhiên, các tác giả phần mềm vẫn giữ quyền sở hữu trí tuệ của họ. Ví dụ về giấy phép công cộng bao gồm các giấy phép như Creative Commons.
* **Giấy phép thương mại :** Đó là giấy phép yêu cầu người dùng trả tiền để sử dụng phần mềm hoặc mua giấy phép từ các nhà phát triển phần mềm. Ví dụ về giấy phép thương mại là các giấy phép của Microsoft và Adobe.
* **Giấy phép tổ chức :** Đó là giấy phép cho phép các tổ chức sử dụng phần mềm, thường có giá cả thấp hơn so với giấy phép thương mại. Ví dụ về giấy phép tổ chức là các giấy phép của Red Hat và Suse.

**II. Bài tập**

**1. List**

Trong Python list là một dạng dữ liệu cho phép lưu trữ nhiều kiểu dữ liệu khác nhau và truy xuất các phần tử bên trong nó thông qua vị trí của phần tử đó trong list. List có thể được coi là một mảng tuần tự như trong các ngôn ngữ khác (như vector trong C++ và ArrayList trong Java). Lists không nhất thiết phải đồng nhất, điều này khiến nó trở thành một công cụ mạnh mẽ nhất trong Python. Một list đơn có thể bao gồm các loại Datatypes như Integers, Strings cũng như Objects. Lists có thể thay đổi được ngay cả sau khi được tạo.

List trong Python được sắp xếp theo thứ tự và có số lượng xác định. Các phần tử trong list được lập chỉ mục theo một trình tự xác định và việc lập chỉ mục của danh sách được thực hiện với 0 là chỉ số đầu tiên. Mỗi yếu tố trong trong list đều có vị trí xác định trong list, điều này cho phép sao chép các phần tử trong danh sách, với mỗi phần tử có vị trí và độ tin cậy riêng biệt

❖ Cách tạo List trong python

List trong Python có thể được tạo bằng cách đặt chuỗi bên trong dấu ngoặc vuông []. Không giống như Sets, List không cần chức năng built-in để thiết lập và bao gồm các yếu tố có thể thay đổi được.

***Ví dụ về cách tạo List trong Python***

# Creation of List

# Creating a List

List = []

print("Blank List: ")

print(List)

# Creating a List of numbers

List = [10, 20, 14]

print("\nList of numbers: ")

print(List)

# Creating a List of strings and accessing

# using index

List = ["Geeks", "For", "Geeks"]

print("\nList Items: ")

print(List[0])

print(List[2])

# Creating a Multi-Dimensional List

# (By Nesting a list inside a List)

List = [['Geeks', 'For'] , ['Geeks']]

print("\nMulti-Dimensional List: ")

print(List)

**Output**

Blank List:

[]

List of numbers:

[10, 20, 14]

List Items

Geeks

Geeks

Multi-Dimensional List:

[['Geeks', 'For'], ['Geeks']]

**❖ Thêm phần tử vào List**

Các phần tử có thể được thêm vào list trong Python bằng cách sử dụng append() chức năng cài sẵn. Chỉ một phần tử có thể được thêm vào danh sách bằng cách sử dụng phương thức append(). Để thêm nhiều phần tử bằng append (), bạn phải sử dụng các vòng lặp. Các bộ giá trị cũng có thể được thêm vào List bằng cách sử dụng các phương thức nối tiếp vì các bộ giá trị là bất biến. Không giống như Set, List cũng có thể được thêm vào danh sách hiện có bằng phương thức append().

**Ví dụ thêm phần tử vào List**

# Addition of elements in a List

# Creating a List

List = []

print("Initial blank List: ")

print(List)

# Addition of Elements

# in the List

List.append(1)

List.append(2)

List.append(4)

print("\nList after Addition of Three elements: ")

print(List)

# Adding elements to the List

# using Iterator

for i in range(1, 4):

    List.append(i)

print("\nList after Addition of elements from 1-3: ")

print(List)

# Adding Tuples to the List

List.append((5, 6))

print("\nList after Addition of a Tuple: ")

print(List)

# Addition of List to a List

List2 = ['For', 'Geeks']

List.append(List2)

print("\nList after Addition of a List: ")

print(List)

**Output**

Initial blank List:

[]

List after Addition of Three elements:

[1, 2, 4]

List after Addition of elements from 1-3:

[1, 2, 4, 1, 2, 3]

List after Addition of a Tuple:

[1, 2, 4, 1, 2, 3, (5, 6)]

List after Addition of a List:

[1, 2, 4, 1, 2, 3, (5, 6), ['For', 'Geeks']]

**❖ Tạo list với nhiều phần tử riêng biệt hoặc trùng lặp**

Một**list trong Python** có thể chứa các giá trị trùng lặp với các giá trị riêng biệt, do đó các giá trị khác biệt hoặc trùng lặp này có thể được chuyển thành một chuỗi tại thời điểm tạo danh sách. Để tạo list với nhiều phần tử trùng lặp, sử dụng lệnh sau:

# Creating a List with

# the use of Numbers

# (Having duplicate values)

List = [1, 2, 4, 4, 3, 3, 3, 6, 5]

print("\nList with the use of Numbers: ")

print(List)

# Creating a List with

# mixed type of values

# (Having numbers and strings)

List = [1, 2, 'Geeks', 4, 'For', 6, 'Geeks']

print("\nList with the use of Mixed Values: ")

print(List)

**❖ Xóa các phần tử trong List**

Để xóa một phần tử trong**list của Python**, bạn có thể sử dụng câu lệnh del nếu bạn biết chính xác các phần tử mà bạn định xóa hoặc sử dụng phương thức remove () nếu bạn không biết. Phương thức remove() chỉ loại bỏ một phần tử tại một thời điểm, để loại bỏ phạm vi phần tử bạn phải sử dụng trình lặp. Phương thức remove () này sẽ loại bỏ các mục nhất định. Ví dụ:

list1 - [‘physics’, ‘chemistry’, 1997, 2000];

print list1

del list1[2];

print “After deleting value at index 2:”

print list1

*Khi đoạn code trên được thực thi, kết quả sẽ như sau:*

[‘physics’, ‘chemistry’, 1997, 2000]

After deleting value at index 2:

[‘physics’, ‘chemistry’, 2000]

**❖ Truy cập các phần tử trong List**

Để truy cập các mục trong danh sach tham chiếu đến số chỉ mục. Sử dụng toán tử chỉ mục [] để truy cập một mục trong**list của Python**. Chỉ mục phải là một số nguyên. List được kiểm tra được truy cập bằng cách sử dụng lập chỉ mục lồng nhau.

# accessing of element from list

# Creating a List with

# the use of multiple values

List = ["Geeks", "For", "Geeks"]

# accessing a element from the

# list using index number

print("Accessing a element from the list")

print(List[0])

print(List[2])

# Creating a Multi-Dimensional List

# (By Nesting a list inside a List)

List = [['Geeks', 'For'] , ['Geeks']]

# accessing an element from the

# Multi-Dimensional List using

# index number

print("Acessing a element from a Multi-Dimensional list")

print(List[0][1])

print(List[1][0])

**Output**

Accessing a element from the list

Geeks

Geeks

Acessing a element from a Multi-Dimensional list

For

Geeks

**2. Tuple**

Tuple là kiểu dữ liệu chuỗi bao gồm nhiều các thành phần có kiểu dữ liệu khác nhau, nhưng nó là không thể thay đổi được. Nói một các khác, tuple là một object Collection phân cách nhau bằng dấu phẩy. Tuple nhanh hơn List cũng ở chỗ này, do tĩnh (static) không thể thay đổi được.

# Nhiều loại tuple

# Empty tuple

tuple\_rong = ()

print(tuple\_rong)

# Tuple bao gồm toàn số

tuple\_toan\_so = (1, 2, 3)

print(tuple\_toan\_so)

# Tuple nhiều loại dữ liệu

tuple\_nhieu\_loai\_du\_lieu = (true, "Kieu Chuoi", 5.2)

print(tuple\_nhieu\_loai\_du\_lieu)

# Tuple kết hợp nhiều loại

tuple\_hon\_hop = ("hon hop", [true, false, 6], (9, 2, 3))

print(tuple\_hon\_hop)

**❖ Sử dụng Tuple**

Tuple cũng hỗ trợ 2 phương thức chính là count và index. Do tính chất immutable nên chắc chắn không thể có các method add và remove.

tuple\_phuong\_thuc = ('t', 'u', 'p', 'l', 'e', 'e', )

print(tuple\_phuong\_thuc.count('e')) # Output: 2

print(tuple\_phuong\_thuc.index('l')) # Output: 3

**❖ Xóa Tuple**

# Xóa tuple như thế nào

xoa\_tuple = ('x', 'o', 'a', 't', 'a', 't', 'c', 'a')

# Tuple không hỗ trợ xóa phần tử do là immutable

del xoa\_tuple[3]

# Nhưng có thể xóa toàn bộ tuple

del xoa\_tuple

**❖ Negative index ( index âm )**

# Index âm thì tuple sẽ index từ sau ra trước

tuple\_index\_am = ('t', 'o', 'p', 'd', 'e', 'v')

# Output: 'v'

print(tuple\_index\_am[-1])

# Output: 't'

print(tuple\_index\_am[-6])

**❖ Slicing**

Tuple cũng hỗ trợ cắt các giá trị trong tuple, lưu ý là lấy giá trị để sử dụng chứ về mặt bản chất là không thay đổi được tuple đã định nghĩa ban đầu.

# Truy cập tuple với slicing

my\_tuple = ('t','r','u','y','c','a','p','i','z')

# Phần tử 1 cho tới 4, có tổng 3

# Output: ('r', 'u', 'y')

print(my\_tuple[1:4])

**❖ Tuple empty** : *tuple\_empty = ()*

Có điều empty thì dễ, nhưng có 1 thì nhớ thêm dấu phẩy (comma) vào nha anh em. Mỗi dấu phẩy thôi nhưng mà quan trọng nha. Không add thì python không hiểu là Tuple đâu á.

# Nếu chưa có sẵn các giá trị nhưng vẫn muốn sử dụng tuple, ta có thể thoải mái định nghĩa tuple empty

tuple\_empty = ("kien dep trai",)

❖ Tuple mmix data types

# Với tuple mixed thoải mái nha anh em

my\_information = tuple("Kien",27,True,"xau",7,"trai",False)

**3. List và Tuple**

**❖ Cú pháp**

# List lưu nhiều giá trị trong một biến duy nhất

tuple\_1 = ["Hoc", "Python", "Khong", "Khó"].

# Tuple cũng lưu nhiều giá trị trong duy nhất 1 biến tuple\_2

tuple\_2 = ("Hoc", "Python", "Khong", "Khó")

Điểm khác biệt rõ ràng dễ nhận ra nhất là cú pháp: với tuple thì ta tạo ra danh sách các phần tử và đóng gói lại trong dấu ngoặc (), trong khi đó list thì định nghĩa các item và sau đó đóng mở trong dấu ngoặc vuông []

**❖ Truy cập**

# Cả List và Tuple đều access bằng index

>>>ten = ("Nguyen","Le","Trung","Kien")

>>>ten[2]

'Le'

>>>ngon\_ngu\_lap\_trinh = ["Python","JavaScript","Java","C"]

>>>ngon\_ngu\_lap\_trinh[0]

'Python'

**❖ Unpacking**

# Khai báo tuple trước (không đẹp xấu gì nữa), tên bỏ đi 1 chữ lót, tính Le là chữ lót nha

>>>ten\_toi = ("Nguyen","Le","Kien")

# Giờ muốn lấy họ, tên lót và tên ra thành các biến, 3 biến phía dưới nha

>>>ho,ten\_lot,ten = ten\_toi

>>>ho

'Nguyen'

>>>ten\_lot

'Le'

>>>ten

'Kien'

**❖ Điểm khác nhau giữa List và Tuple**

**- Mutable và immutabe (thay đổi và không thay đổi)**

Đầu tiên cũng là các khác biệt rõ ràng nhất. List thì có thể thay đổi được, nhưng tuple thì không. Một khi đã khai báo tuple thì không thể thay đổi giá trị trong đó.

# List có thể thay đổi được phần tử

list\_co\_the\_doi = [4, 2, 3]

list\_co\_the\_doi[1] = 9

print(list\_co\_the\_doi) = [9, 2, 3]

# Tuple thì cần nha, không cần Python không hiểu

tuple\_khong\_doi = (4, 2, 3, [6, 5])

# Lỗi, không thể gán giá trị cho tuple

# tuple\_khong\_doi[1] = 9

**- Tốc độ**

Do tuple là immutable (không thể thay đổi), nên tốc độ duyệt hoặc lấy theo index của tuple rất nhanh, hơn List.



**3. Dictionary**

Dictionary trong Python là một tập hợp các cặp key và value không có thứ tự. Nó là một container mà chứa dữ liệu, được bao quanh bởi các dấu ngoặc móc đơn {}. Mỗi cặp key-value được xem như là một item. Key mà đã truyền cho item đó phải là duy nhất, trong khi đó value có thể là bất kỳ kiểu giá trị nào. Key phải là một kiểu dữ liệu không thay đổi (immutable) như chuỗi, số hoặc tuple.

Key và value được phân biệt riêng rẽ bởi một dấu hai chấm (:). Các item phân biệt nhau bởi một dấu phảy (,). Các item khác nhau được bao quanh bên trong một cặp dấu ngoặc móc đơn tạo nên một Dictionary trong Python

Ví dụ:

data={100:'Hoang' ,101:'Nam' ,102:'Binh'}

print data

Kết quả là:

>>>

{100: 'Hoang', 101: 'Nam', 102: 'Binh'}

>>>

**Các thuộc tính của key trong Dictionary**

Không có hạn chế nào với các value trong Dictionary, tuy nhiên với key thì bạn cần chú ý các điểm sau:

(a) Nhiều hơn một entry cho mỗi key là không được phép. Nghĩa là không cho phép bản sao các key được xuất hiện. Khi bắt gặp nhiều bản sao key trong phép gán, thì phép gán cuối cùng được thực hiện. Ví dụ:

dict = {'Ten': 'Hoang', 'Tuoi': 7, 'Ten': 'Nam'};

print "dict['Ten']: ", dict['Ten']

**Kết quả là:**

dict['Ten']: Nam

(b) Key phải là immutable. Nghĩa là bạn chỉ có thể sử dụng chuỗi, số hoặc tuple làm key của Dictionary. Dưới đây là ví dụ đơn giản:

dict = {['Ten']: 'Hoang', 'Tuoi': 7};

print "dict['Ten']: ", dict['Ten']

**Khi code trên được thực thi sẽ cho kết quả:**

Traceback (most recent call last):

File "test.py", line 3, in <module>

dict = {['Ten']: 'Hoang', 'Tuoi': 7};

TypeError: list objects are unhashable

**❖ Truy cập các giá trị trong Dictionary trong Python**

Khi chỉ mục không được định nghĩa với Dictionary, thì các giá trị trong Dictionary có thể được truy cập thông qua các key của chúng.

**Cú pháp: <ten\_dictionary>[key]**

**Ví dụ:**

data1={'Id':100, 'Ten':'Thanh', 'Nghenghiep':'Developer'}

data2={'Id':101, 'Ten':'Chinh', 'Nghenghiep':'Trainer'}

print "Id cua nhan vien dau tien la",data1['Id']

print "Id cua nhan vien thu hai la",data2['Id']

print "Ten cua nhan vien dau tien la:",data1['Ten']

print "Nghe nghiep cua nhan vien thu hai la:",data2['Nghenghiep']

**Kết quả là:**

**>>>**

Id cua nhan vien dau tien la 100

Id cua nhan vien thu hai la 101

Ten cua nhan vien dau tien la is Thanh

Nghe nghiep cua nhan vien thu hai la Trainer

**>>>**

**❖ Cập nhật Dictionary trong Python**

Item (cặp key-value) có thể được cập nhật. Bạn cập nhật một Dictionary bằng cách thêm một entry mới hoặc một cặp key-value mới, sửa đổi một entry đã tồn tại, hoặc xóa một entry đang tồn tại như trong ví dụ đơn giản sau:

data1={'Id':100, 'Ten':'Thanh', 'Nghenghiep':'Developer'}

data2={'Id':101, 'Ten':'Chinh', 'Nghenghiep':'Trainer'}

data1['Nghenghiep']='Manager'

data2['Mucluong']=17000000

data1['Mucluong']=12000000

print data1

print data2

**Khi code trên được thực thi sẽ cho kết quả:**

**>>>**

{'Mucluong': 12000000, 'Nghenghiep': 'Manager','Id': 100, 'Ten': 'Thanh'}

{'Mucluong': 17000000, 'Nghenghiep': 'Trainer', 'Id': 101, 'Ten': 'Chinh'}

**>>>**

**❖ Xóa phần tử từ Dictionary trong Python**

Với Dictionary, bạn có thể xóa một phần tử đơn hoặc xóa toàn bộ nội dung của Dictionary đó. Bạn sử dụng lệnh del để thực hiện các hoạt động này.

**Cú pháp để xóa một item từ Dictionary:**

del ten\_dictionary[key]

Để xóa cả Dictionary, bạn sử dụng cú pháp:

del ten\_dictionary

Ví dụ:

data={100:'Hoang', 101:'Thanh', 102:'Nam'}

del data[102]

print data

del data

|  |  |
| --- | --- |
| **Hàm** | **Mô tả** |
| Hàm cmp(dict1, dict2) | So sánh các phần tử của cả hai dict |
| Hàm len(dict) | Độ dài của dict. Nó sẽ là số item trong Dictionary này |
| Hàm str(dict) | Tạo ra một biểu diễn chuỗi có thể in được của một dict |
| Hàm type(variable) | Trả về kiểu của biến đã truyền. Nếu biến đã truyền là Dictionary, thì nó sẽ trả về một kiểu Dictionary |

|  |  |
| --- | --- |
| **Phương thức** | **Mô tả** |
| Phương thức dict.clear() | Xóa tất cả phần tử của dict |
| Phương thức dict.copy() | Trả về bản sao của dict |
| Phương thức fromkeys(seq,value1)/ fromkeys(seq) | Được sử dụng để tạo một Dictionary mới từ dãy seq và value1. Trong đó dãy seq tạo nên các key và tất cả các key chia sẻ các giá trị từ value1. Trong trường hợp value1 không được cung cấp thì value của các key được thiết lập là None |
| Phương thức dict.get(key, default=None) | Trả về giá trị của key đã cho. Nếu key không có mặt thì phương thức này trả về None |
| Phương thức dict.has\_key(key) | Trả về true nếu key là có mặt trong Dictionary, nếu không là false |
| Phương thức dict.items() | Trả về tất cả các cặp (key-value) của một Dictionary |
| Phương thức dict.keys() | Trả về tất cả các key của một Dictionary |
| Phương thức dict.setdefault(key, default=None) | Tương tự get(), nhưng sẽ thiết lập dict[key]=default nếu key là không tồn tại trong dict |
| Phương thức dict.update(dict2) | Được sử dụng để thêm các item của dictionary 2 vào Dictionary đầu tiên |
| Phương thức dict.values() | Trả về tất cả các value của một Dictionary |

**4. Iterators**

Iterator ở khắp mọi nơi trong Python, bên trong các vòng lặp, comprehension, generator…

Nó đơn giản chỉ là các đối tượng cho phép ta lấy từng phần tử, bất cứ khi nào bạn sử dụng vòng lặp hay các kĩ thuật để có được giá trị một nhóm phần tử ở một thời điểm nào đó.

Về mặt kỹ thuật, Iterator trong Python phải thực hiện hai phương thức đặc biệt là \_\_iter\_\_() và \_\_next\_\_(), gọi chung là giao thức iterator (Iterator Protocol)

* Phương thức \_\_iter\_\_ trả về chính đối tượng iterator. Phương thức này được yêu cầu cài đặt cho cả đối tượng "iterable" và iterator để có thể sử dụng các câu lệnh for và in.
* Phương thức \_\_next\_\_ trả về phần tử tiếp theo. Nếu không còn phần tử nào nữa thì sẽ có lỗi StopIteration xảy ra.
* **Iterable object** là một đối tượng sau khi sử dụng các phương thức sẽ trả về một iterator, ví dụ như Chuỗi, List, Tuple.
* **Iter()** là một hàm dựng sẵn trong Python nhận đầu vào là một đối tượng iterable và trả về kết quả là một iterator.

# Khai bao mot list

my\_list = [4, 7, 0, 3]

# lay mot iterator bang cach su dung iter()

my\_iter = iter(my\_list)

## su dung next()

#prints 4

print(next(my\_iter))

#prints 7

print(next(my\_iter))

## next(obj) chinh la obj.\_\_next\_\_()

#prints 0

print(my\_iter.\_\_next\_\_())

#prints 3

print(my\_iter.\_\_next\_\_())

## Xay ra loi StopIteration vi het gia tri tra ve

next(my\_iter)

Chạy chương trình, kết quả trả về là:

4

7

0

3

Traceback (most recent call last):

File "<stdin>", line 24, in <module>

next(my\_iter)

StopIteration

Một cách tương tự trả về kết quả này là ta sử dụng [**vòng lặp *for***](https://quantrimang.com/hoc/vong-lap-for-trong-python-142234)**.**

>>> for element in my\_list:

... print(element)

...

4

7

0

3

Chúng ta có thể tự xây dựng iterator là một class. Xây dựng Iterator rất dễ dàng trong Python, chúng ta chỉ cần thực hiện các phương thức \_\_iter\_\_() và \_\_next\_\_().

class PowTwo:

def \_\_init\_\_(self, max = 0):

self.max = max

def \_\_iter\_\_(self):

self.n = 0

return self

def \_\_next\_\_(self):

if self.n <= self.max:

result = 2 \*\* self.n

self.n += 1

return result

else:

raise StopIteration

Phương thức \_\_iter\_\_ sẽ làm đối tượng trở thành đối tượng iterable.

Giá trị trả về của \_\_iter\_\_ là một iterator. Nó cần có phương thức \_\_next\_\_ và trả về StopIteration nếu không còn phần thử nào nữa

**Iterator lặp vô hạn**

Không phải tất cả các đối tượng iterator đều sẽ được gọi hết các phần tử và kết thúc khi không còn phần tử. Có một số trường hợp iterator sẽ lặp vô hạn. Ví dụ như sau:

>>> int()

0

>>> inf = iter(int,1)

>>> next(inf)

0

>>> next(inf)

0

Lists, tuples, dictionaries và sets đều là những đối tượng iterable - có thể lặp. Chúng là các container có thể lặp mà bạn có thể lấy một iterator bên trong.

Tất cả những đối tượng có phương thức iter() đều được dùng để có iterator.

Ví dụ:

Trả về một iterator từ tuple và print từng giá trị:

mytuple = ("apple", "banana", "cherry")

myit = iter(mytuple)

print(next(myit))

print(next(myit))

print(next(myit))

Thậm chí các chuỗi là đối tượng có thể lặp và trả về một iterator. Ví dụ:

mystr = "banana"

myit = iter(mystr)

print(next(myit))

print(next(myit))

print(next(myit))

print(next(myit))

print(next(myit))

print(next(myit))

**StopIteration**

Để tránh lặp lại mãi mãi, bạn có thể dùng lệnh **StopIteration**. Trong phương thức **\_next\_()**, bạn có thể thêm điều kiện kết thúc để phát sinh lỗi khi việc lặp lại đạt đủ số lần nhất định.

Ví dụ: Dừng sau 20 lần lặp.

class MyNumbers:

def \_\_iter\_\_(self):

self.a = 1

return self

def \_\_next\_\_(self):

if self.a <= 20:

x = self.a

self.a += 1

return x

else:

raise StopIteration

myclass = MyNumbers()

myiter = iter(myclass)

for x in myiter:

print(x)

**5. List comprihension**

List-comprehension là một cú pháp cho phép lập trình viên nhanh chóng tạo ra một biến dữ liệu list mới từ một list cũ hoặc vòng lặp dạng in-line, kết hợp với các điều kiện cho trước.

Về bản chất, lập trình viên hoàn toàn có thể tạo ra một list mới bằng cách sử dụng vòng lặp for/while thông thường.

Cú pháp cơ bản:

# basic looping

***for*** item ***in*** iterable:

expression

# list comprehension

[expression ***for*** item ***in*** iterable]

Ví dụ: Tạo ra một list danh sách các số từ 0 đến 99

-----# basic looping

my\_list **=** []

***for*** number ***in*** ***range***(100):

my\_list.append(number)

-----# list comprehension

my\_comprehension\_list **=** [number ***for*** number ***in*** ***range***(100)]

Ví dụ: Cho một list gồm các số nguyên, hãy tạo ra một list mới với các phần tử là bình phương của phần tử trong list cũ

old\_list **=** [1, 2, 3, 4, 5]

------# basic looping

new\_list\_1 **=** []

***for*** number ***in*** old\_list:

new\_list\_1.append(number **\*** number)

------# list comprehension

new\_list\_2 **=** [number **\*** number **for** number **in** old\_list]

**Một vài ứng dụng trong List-comprehension**

**❖ Lọc dữ liệu ( filter )**

# basic looping with a conditional statement

**for** item **in** iterable:

if some\_condition:

expression

# list comprehension with a conditional statement

[expression **for** item **in** iterable **if** some\_condition]

Ví dụ: Cho một list các số nguyên, tạo ra một list mới chỉ chứa các số chia hết cho 3.

old\_list **=** [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

# basic looping

new\_list\_1 **=** []

**for** number **in** old\_list:

**if** number % 3 == 0:

new\_list\_1.append(number)

# list comprehension

new\_list\_2 **=** [number **for** number **in** old\_list **if** number % 3 **==** 0]

Ngoài cách đặt filter ở cuối, chúng ta cũng có thể đặt filter ở phía trước.

Ví dụ: Cho một list các số nguyên, tạo ra một list mới chỉ chứa các số chia hết cho 2.

old\_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

# list comprehension

new\_list = [(number **if** number % 2 == 0 **else** 0) **for** number **in** old\_list]

print(new\_list)

**❖ Lọc dữ liệu với biểu thức so sánh bên ngoài**

Phần điều kiện filter ở some\_condition ở thể không phải là so sánh trực tiếp mà có gọi đến một function bên ngoài và trả về kết quả so sánh.

Ví dụ: Cho danh sách tên các con vật, tạo ra một danh sách mới chứa tên các con vật có 4 chân.

pets = ['dog', 'cat', 'bird', 'fish', 'mouse', 'chicken']

def has\_four\_legs(pet):

return pet in ['dog', 'cat', 'mouse']

new\_pets = [pet for pet in pets if has\_four\_legs(pet)]

print(new\_pets)

**❖ Sử dụng kết hợp Warlus Operator**

Python có cú pháp khá thú vị là Walrus Operator (:= ) dùng để gán chung dữ liệu ngay trong các vòng lặp, câu lệnh so sánh, ...

Ví dụ: Cho 1 chuỗi, lấy ra ngẫu nhiên 10 ký tự trong chuỗi, sau đó tạo ra một list, nếu ký tự lấy ra không phải là nguyên âm thì thực hiện viết hoa ký tự đó và đưa vào list.

import random

letters = list('this is a sample about list-comprehension and walrus operator')

consonants = [letter.upper() for \_ in range(0, 10) if (letter := random.choice(letters)) not in 'aeoui']

print(consonants)

**5. Generators**

Generator là cách đơn giản được dùng để có thể tạo ra iterator. Bạn có thể đơn hiểu đơn giản Generator là gì như sau: Generator là một trong những hàm trả về một đối tượng (iterator) mà ở đó chúng ta có thể lặp lại (một giá trị ở một thời điểm).

Ngoài ra, chúng cũng có nhiệm vụ tạo ra đối tượng theo kiểu danh sách, tuy nhiên bạn chỉ có thể phê duyệt nó qua những phần tử generator một lần duy nhất. Bởi vì, Generator không lưu các dữ liệu trong bộ nhớ nên khi lặp lại thì chúng sẽ tạo ra phần tử tiếp theo trong dãy rồi trả về phần tử đó.

**Để tạo generator trong Python,** bạn sử dụng từ khóa def giống như khi định nghĩa một hàm. Trong generator, dùng câu lệnh yield để trả về các phần tử thay vì câu lệnh return như bình thường.

Nếu một hàm chứa ít nhất một yield (có thể có nhiều yield và thêm cả return) thì chắc chắn đây là một hàm generator. Trong trường hợp này, cả yield và return sẽ trả về các giá trị từ hàm.

Điều khác biệt ở đây là return sẽ chấm dứt hoàn toàn một hàm, còn yield sẽ chỉ tạm dừng các trạng thái bên trong hàm và sau đó vẫn có thể tiếp tục khi được gọi trong các lần sau.

**Sự khác biệt giữa Generator và hàm thông thường**

Đây là một số khác biệt giữa hàm generator và [hàm thông thường](https://quantrimang.com/hoc/cac-ham-trong-python-143771):

* Hàm generator chứa một hoặc nhiều câu lệnh yield.
* Khi được gọi, generator trả về một đối tượng (iterator) nhưng không bắt đầu thực thi ngay lập tức.
* Các phương thức như \_\_iter \_\_() và \_\_next\_\_() được triển khai tự động. Vì vậy, chúng ta có thể lặp qua các mục bằng cách sử dụng next().
* Yield sẽ tạm dừng hàm, các biến cục bộ và trạng thái của chúng được ghi nhớ giữa các lệnh gọi liên tiếp. Mỗi lần lệnh yield được chạy, nó sẽ sinh ra một giá trị mới.
* Cuối cùng, khi hàm kết thúc, StopIteration sẽ xảy ra nếu tiếp tục gọi hàm.

Dưới đây là một ví dụ để minh họa tất cả các điểm đã nêu ở trên. Chúng ta có một hàm generator có tên my\_gen() với một số câu lệnh yield.

Để khởi động lại quá trình, tạo một đối tượng generator khác bằng cách sử dụng đối tượng như

a = my\_gen().

***Lưu ý: Có thể sử dụng generator trực tiếp cho các vòng lặp for.***

Vòng lặp for lấy một iterator và lặp lại nó bằng hàm next(), tự động kết thúc khi StopIteration xảy ra.

# Hàm generator đơn giản

def my\_gen():

n = 1

print('Doan text nay duoc in dau tien')

# Hàm Generator chứa câu lệnh yield

yield n

n += 1

print('Doan text nay duoc in thu hai')

yield n

n += 1

print('Doan text nay duoc in cuoi cung')

yield n

# Sử dụng vòng lặp for

for item in my\_gen():

print(item)

Chạy chương trình, kết quả trả về là :

Doan text nay duoc in dau tien

1

Doan text nay duoc in thu hai

2

Doan text nay duoc in cuoi cung

3

**❖ Generator với các vòng lặp python**

Ví dụ về generator đảo ngược chuỗi.

def rev\_str(my\_str):

length = len(my\_str)

for i in range(length - 1,-1,-1):

yield my\_str[i]

# Vòng lặp for đảo ngược chuỗi

# Viết bởi Quantrimang.com

# Output:

# o

# l

# l

# e

# h

for char in rev\_str("hello"):

print(char)

**❖ Biểu thức Generator**

Generator có thể dễ dàng được tạo ra khi sử dụng biểu thức generator.

Giống như Lambda tạo một hàm vô danh trong Python, generator cũng tạo một biểu thức generator vô danh. Cú pháp tương tự như cú pháp của list comprehension, nhưng dấu ngoặc vuông được thay thế bằng dấu ngoặc tròn.

List comprehension thì trả về một list, còn biểu thức generator trả về một generator tại một thời điểm khi được yêu cầu. Vì lý do này, biểu thức generator sử dụng ít bộ nhớ hơn, đem lại hiệu quả hiệu suất hơn so với list comprehension tương đương.

# Khởi tạo danh sách

my\_list = [1, 3, 6, 10]

# bình phương mỗi phần tử bằng cách sử dụng list comprehension

# Output: [1, 9, 36, 100]

[x\*\*2 for x in my\_list]

# kết quả tương tự khi sử dụng biểu thức generator

# Output: <generator object <genexpr> at 0x0000000002EBDAF8>

(x\*\*2 for x in my\_list)

Có thể thấy ở ví dụ trên, biểu thức generator không tạo ra kết quả cần thiết ngay lập tức mà trả về đối tượng generator, cứ mỗi lần lặp thì chúng sẽ tạo phần tử tiếp theo trong dãy và trả về phần tử đó.

# Khởi tạo danh sách

my\_list = [1, 3, 6, 10]

a = (x\*\*2 for x in my\_list)

# Output: 1

print(next(a))

# Output: 9

print(next(a))

# Output: 36

print(next(a))

# Output: 100

print(next(a))

# Output: StopIteration

next(a)

Biểu thức generator sử dụng bên trong các hàm thì có thể bỏ qua các dấu ngoặc tròn.

>>> sum(x\*\*2 for x in my\_list)

146

>>> max(x\*\*2 for x in my\_list)

100

**Tại sao nên sử dụng Generator ?**

**1. Đơn giản hóa code, dễ triển khai**

Generator có thể giúp code được triển khai rõ ràng và ngắn gọn hơn so với lớp iterator tương tự. Để minh họa cho việc này, chúng ta sẽ lấy một ví dụ cụ thể.

class PowTwo:

def \_\_init\_\_(self, max = 0):

self.max = max

def \_\_iter\_\_(self):

self.n = 0

return self

def \_\_next\_\_(self):

if self.n > self.max:

raise StopIteration

result = 2 \*\* self.n

self.n += 1

return result

**Đoạn code này khá dài. Bây giờ hãy thử sử dụng hàm generator.**

def PowTwoGen(max = 0):

n = 0

while n < max:

yield 2 \*\* n

n += 1

**2. Sử dụng ít bộ nhớ**

Một hàm thông thường khi trả về list sẽ lưu toàn bộ list trong bộ nhớ. Trong phần lớn các trường hợp, điều đó là không hay khi phải sử dụng đến dung lượng bộ nhớ lớn đến vậy.

Generator sẽ sử dụng ít bộ nhớ hơn vì chúng chỉ thực sự tạo kết quả khi được gọi tới, sinh ra một phần tử tại một thời điểm, đem lại hiệu quả nếu chúng ta không có nhu cầu duyệt nó quá nhiều lần.

**3. Tạo ra các list vô hạn**

Một hàm thông thường khi trả về list sẽ lưu toàn bộ list trong bộ nhớ. Trong phần lớn các trường hợp, điều đó là không hay khi phải sử dụng đến dung lượng bộ nhớ lớn đến vậy.

Generator sẽ sử dụng ít bộ nhớ hơn vì chúng chỉ thực sự tạo kết quả khi được gọi tới, sinh ra một phần tử tại một thời điểm, đem lại hiệu quả nếu chúng ta không có nhu cầu duyệt nó quá nhiều lần.

def all\_even():

n = 0

while True:

yield n

n += 2