



TRƯỜNG ĐẠI HỌC VINH
VIỆN KỸ THUẬT VÀ CÔNG NGHỆ

TÓM TẮT BÀI GIẢNG
KỸ THUẬT LẬP TRÌNH
(TÀI LIỆU LƯU HÀNH NỘI BỘ)

Giảng viên:

Đơn vị: Viện Kỹ thuật và Công nghệ

Email: sondt@vinhuni.edu.vn

Nghệ An, 2018



THÔNG TIN HỌC PHẦN

- Tên học phần: **Kỹ thuật lập trình**
- Mã học phần: **ELE20004**
- Khối kiến thức: Kiến thức cơ sở ngành
- Số tín chỉ: 05
 - Lý thuyết: 30 tiết; Thực hành: 15 tiết Đồ án: 15 tiết; Tự học: 150 tiết
- Vị trí học phần:
 - Học phần tiên quyết: Tin học nhóm ngành kỹ thuật (INF20004)
 - Học phần song hành: Không



MÔ TẢ HỌC PHẦN

Kỹ thuật lập trình là học phần đầu tiên về lập trình, dành cho các sinh viên chưa có kiến thức nào về lập trình trước đó.

Học phần cung cấp những kiến thức và kỹ năng căn bản về lập trình bao gồm hai phương pháp lập trình: lập trình có cấu trúc và lập trình hướng đối tượng. Bên cạnh đó sinh viên còn được củng cố về các kỹ năng về làm việc nhóm để viết chương trình phần mềm giải quyết các vấn đề trong thực tiễn.



MỤC TIÊU HỌC PHẦN

- **G1** - Sử dụng các thuật toán để giải quyết vấn đề;
- **G2** - Thực hiện lập trình bằng ngôn ngữ Python;
- **G3** - Nắm vững kiến thức về lập trình hướng đối tượng;
- **G4** - Thể hiện phong cách lập trình chuyên nghiệp;
- **G5** - Thực hiện giải quyết vấn đề bằng lập trình phần mềm;
- **G6** - Thể hiện kỹ năng làm việc nhóm và giao tiếp hiệu quả.



NỘI DUNG HỌC PHẦN

- **CHƯƠNG 1** - Giới thiệu về máy tính và lập trình
- **CHƯƠNG 2** - Ngôn ngữ lập trình Python
- **CHƯƠNG 3** - Lập trình hàm
- **CHƯƠNG 4** - Các kiểu dữ liệu có cấu trúc
- **CHƯƠNG 5** - Thiết kế module
- **CHƯƠNG 6** - Lập trình hướng đối tượng
- **CHƯƠNG 7** - Thao tác trên tập tin và thư mục
- **CHƯƠNG 8** - Lập trình giao diện



HÌNH THỨC ĐÁNH GIÁ

- Đánh giá quá trình:
 - Ý thức học tập **50%**
 - Hồ sơ học tập (bài tập) **10%**
 - Đánh giá giữa kỳ (thực hành) **20%**
 - Đánh giá cuối kỳ (thực hành) **20%**
- Đánh giá cuối kỳ:
 - Đồ án **50%**
 - **50%**



NGUỒN HỌC LIỆU

Giáo trình:

- [1]. Allen B. Downey, Think Python, O'Reilly Media, Inc, 2015.
- [2]. Võ Duy Tuấn, Python cơ bản, Ebook, 2016.

Tài liệu tham khảo:

- [3]. Magnus Lie Hetland, Beginning Python: From Novice to Professional, APress, 2008.
- [4]. Website: <https://docs.python.org/3.7/tutorial>.



QUY ĐỊNH HỌC PHẦN

- Sinh viên phải tham dự tối thiểu 80% thời gian lên lớp;
- Có đầy đủ học liệu theo yêu cầu của giảng viên;
- Tham gia tích cực hoạt động thảo luận theo nhóm;
- Báo cáo bài tập trên hệ thống Github đảm bảo chất lượng, đáp ứng đầy đủ nội dung yêu cầu của môn học.
- Thực hiện đồ án (project) và báo cáo đầy đủ nội dung theo quy định.



**TRƯỜNG ĐẠI HỌC VINH
VIỆN KỸ THUẬT VÀ CÔNG NGHỆ**

**GIỚI THIỆU VỀ GIT VÀ
HƯỚNG DẪN SỬ DỤNG HỆ THỐNG GITHUB**

Nghệ An, 2018



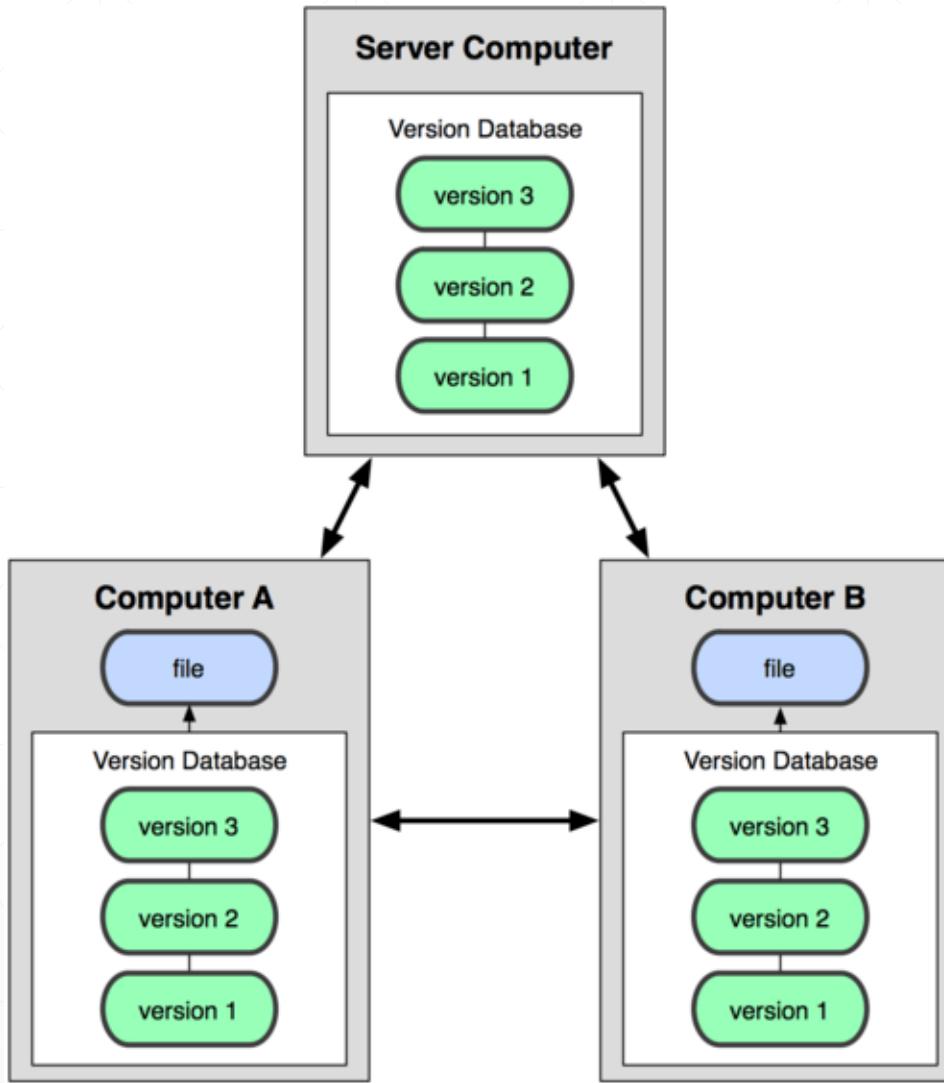
GIỚI THIỆU VỀ GIT

- Git là tên gọi của một Hệ thống quản lý phiên bản phân tán (Distributed Version Control System – DVCS) là một trong những hệ thống quản lý phiên bản phân tán phổ biến nhất hiện nay.
- Hệ thống giúp mỗi máy tính có thể lưu trữ nhiều phiên bản khác nhau của một mã nguồn được nhân bản (clone) từ một kho chứa mã nguồn (repository);



GIỚI THIỆU VỀ GIT

- Mỗi thay đổi vào mã nguồn trên máy tính sẽ có thể ủy thác (commit) rồi đưa lên máy chủ nơi đặt kho chứa chính.
- Và một máy tính khác (nếu họ có quyền truy cập) cũng có thể clone lại mã nguồn từ kho chứa hoặc clone lại một tập hợp các thay đổi mới nhất trên máy tính kia.
- Trong Git, thư mục làm việc trên máy tính gọi là Working Tree.



Mô hình hoạt động của Distributed Version Control System – DVCS



GIỚI THIỆU VỀ GIT

▪ Hiểu một cách đơn giản:

- Git sẽ giúp bạn lưu lại các phiên bản của những lần thay đổi vào mã nguồn và có thể dễ dàng khôi phục lại dễ dàng mà không cần copy lại mã nguồn rồi cất vào đâu đó.
- Một người khác có thể xem các thay đổi của bạn ở từng phiên bản, họ cũng có thể đổi chiếu các thay đổi của bạn rồi gộp phiên bản của bạn vào phiên bản của họ.
- Cuối cùng là tất cả có thể đưa các thay đổi vào mã nguồn của mình lên một kho chứa mã nguồn.



GIỚI THIỆU VỀ GITHUB

- GitHub là một dịch vụ cung cấp kho lưu trữ mã nguồn Git dựa trên nền web cho các dự án phát triển phần mềm.
- GitHub cung cấp cả phiên bản trả tiền lẫn miễn phí cho các tài khoản. Các dự án mã nguồn mở sẽ được cung cấp kho lưu trữ miễn phí.
- Tính đến tháng 4 năm 2016, GitHub có hơn 14 triệu người sử dụng với hơn 35 triệu kho mã nguồn, làm cho nó trở thành máy chủ chứa mã nguồn lớn trên thế giới.



GIỚI THIỆU VỀ GITHUB

- Github đã trở thành một yếu tố có sức ảnh hưởng trong cộng đồng phát triển mã nguồn mở.
- Nhiều nhà phát triển đã bắt đầu xem nó là một sự thay thế cho sơ yếu lý lịch và một số nhà tuyển dụng yêu cầu các ứng viên cung cấp một liên kết đến tài khoản Github để đánh giá ứng viên.
- Vào ngày 4 tháng 6 năm 2018, Microsoft đã thông báo việc đạt được thỏa thuận mua lại GitHub với giá 7.5 tỷ Đôla Mỹ.



HƯỚNG DẪN SỬ DỤNG GITHUB

▪ Tạo tài khoản

- Để bạn có thể lưu trữ mã nguồn và sử dụng các dịch vụ của GitHub, trước tiên, chúng ta phải tạo một tài khoản của GitHub.
- Để tạo tài khoản, đơn giản, trên trang web chính của GitHub, bạn bấm nút ***Sign up*** hoặc ***Sign up for GitHub***.

The screenshot shows the GitHub homepage. At the top, there is a navigation bar with links for Personal, Open source, Business, Explore, Pricing, Blog, and Support. There is also a search bar labeled "Search GitHub" and buttons for "Sign in" and "Sign up". The main background image features a blurred photo of people working at a computer and a GitHub logo figurine. The central text reads "How people build software" and "Millions of developers use GitHub to build personal projects, support their businesses, and work together on open source technologies.". To the right, there is a sign-up form with fields for "Pick a username", "Your email address", and "Create a password". A note below the password field says "Use at least one letter, one numeral, and seven characters.". A large green "Sign up for GitHub" button is present. Below it, a small note states: "By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy policy](#). We'll occasionally send you account related emails."



Join us at GitHub Universe

The event for people building the future of software,
September 13-15 in San Francisco. [Learn more](#)

Tài khoản của GitHub





HƯỚNG DẪN SỬ DỤNG GITHUB

▪ **Bước 1:**

- Tiếp theo, tại bước 1, bạn điền username, email và password.
- Password của bạn nên có ít nhất một ký tự in thường, một chữ số và gồm ít nhất 7 ký tự.
- Bấm Create an account để sang bước 2

The screenshot shows the GitHub 'Join GitHub' page. At the top, there are three steps: 'Step 1: Set up a personal account', 'Step 2: Choose your plan', and 'Step 3: Tailor your experience'. The first step is active. Below it, the 'Create your personal account' section contains fields for 'Username' (lvman), 'Email Address' (manleviethuy@gmail.com), and 'Password' (a redacted field). A note says, 'This will be your username — you can enter your organization's username next.' Another note under the email address says, 'You will occasionally receive account related emails. We promise not to share your email with anyone.' Below the password field is a note: 'Use at least one lowercase letter, one numeral, and seven characters.' At the bottom of the form, a note states: 'By clicking on "Create an account" below, you are agreeing to the [Terms of Service](#) and the [Privacy Policy](#)'. A green 'Create an account' button is at the bottom.

Join GitHub

The best way to design, build, and ship software.

Step 1:
Set up a personal account

Step 2:
Choose your plan

Step 3:
Tailor your experience

Create your personal account

Username

lvman

This will be your username — you can enter your organization's username next.

Email Address

manleviethuy@gmail.com

You will occasionally receive account related emails. We promise not to share your email with anyone.

Password

.....

Use at least one lowercase letter, one numeral, and seven characters.

By clicking on "Create an account" below, you are agreeing to the [Terms of Service](#) and the [Privacy Policy](#).

Create an account

Tài khoản của GitHub





HƯỚNG DẪN SỬ DỤNG GITHUB

▪ **Bước 2:**

- Tại bước 2, bạn sẽ lựa chọn loại tài khoản.
- Hiện tại, GitHub cung cấp hai loại tài khoản là: miễn phí cho việc lưu trữ các dự án mở (chọn Unlimited public repertoires for free) và có phí (7 đô/tháng) cho lưu trữ các dự án đóng (chọn Unlimited private repertoires for \$7/month).
- Bấm Continue để sang bước 3.

The screenshot shows the GitHub welcome screen for a new user named @lvman. At the top, there are navigation icons, a search bar, and links for Pull requests, Issues, and Gist. To the right are account settings and a green 'Code' icon.

Welcome to GitHub

You've taken your first step into a larger world, @lvman.

Completed: Set up a personal account

Step 2: Choose your plan

Step 3: Tailor your experience

Choose your personal plan

Unlimited public repositories for free.

Unlimited private repositories for \$7/month. [\(view in VND\)](#)

Don't worry, you can cancel or upgrade at any time.

Help me set up an organization next

Organizations are separate from personal accounts and are best suited for businesses who need to manage permissions for many employees. [Learn more about organizations.](#)

Both plans include:

- Collaborative code review
- Issue tracking
- Open source community
- Unlimited public repositories
- Join any organization

Continue

© 2016 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Help](#)

Contact GitHub API Training Shop Blog About

Tài khoản của GitHub





HƯỚNG DẪN SỬ DỤNG GITHUB

▪ **Bước 3:**

- Ở bước 3, bạn trả lời một số câu hỏi điều tra của GitHub, rồi bấm Submit.
- GitHub sẽ gửi cho bạn một email vào địa chỉ mail của bạn đã cung cấp cho GitHub để xác thực. Trong email đó, bạn bấm đường link Verify email address để xác thực. Lúc này, GitHub sẽ gửi email chúc mừng và cho phép bạn tạo repertoire (kho) đầu tiên trên GitHub.

A screenshot of a web browser displaying the GitHub welcome screen for a user named @lvman. The browser has a light gray header with standard navigation icons (back, forward, search, refresh) and a tab bar showing 'GitHub, Inc.' The main content area features a large 'Welcome to GitHub' heading and a sub-headline: 'You'll find endless opportunities to learn, code, and create, @lvman.' Below this, there are three progress steps: 'Completed Set up a personal account' (green checkmark icon), 'Step 2: Choose your plan' (blue gear icon), and 'Step 3: Tailor your experience' (blue gear icon). A section titled 'How would you describe your level of programming experience?' contains three radio buttons: 'Totally new to programming' (selected, blue outline), 'Somewhat experienced' (gray outline), and 'Very experienced' (gray outline). Another section, 'What do you plan to use GitHub for? (check all that apply)', includes several checkboxes: 'Project Management' (gray outline), 'Development' (gray outline), 'Research' (gray outline), 'School projects' (selected, blue outline), 'Design' (gray outline), and 'Other (please specify)' (gray outline). A third section, 'Which is closest to how you would describe yourself?', shows three radio buttons: 'I'm a hobbyist' (gray outline), 'I'm a student' (selected, blue outline), and 'I'm a professional' (gray outline). A fourth section, 'What are you interested in?', features a search input field containing 'c#' with an 'X' button to clear it, and a placeholder text: 'e.g. tutorials, android, ruby, web-development, machine-learning, open-source'. At the bottom of this section are two buttons: a green 'Submit' button and a blue 'skip this step' link.

Tài khoản của GitHub



The screenshot shows the GitHub homepage. At the top, there's a navigation bar with icons for back, forward, search, and other GitHub features. The main title "GitHub" is centered above a search bar with the placeholder "Search GitHub". Below the search bar are links for "Pull requests", "Issues", and "Gist". On the right side of the header, there are buttons for creating a new repository ("+") and a pull request ("P"). The main content area has a light blue background with the text "Learn Git and GitHub without any code!". Below this, it says "Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request." There are two buttons: a green one labeled "Read the guide" and a white one labeled "Start a project". At the bottom of the page, there's a footer with copyright information ("© 2016 GitHub, Inc. Terms Privacy Security Status Help"), a GitHub logo, and links for "Contact GitHub API Training Shop Blog About".

Chọn Start a project để bắt đầu tạo một repertoire đầu tiên trên GitHub.





HƯỚNG DẪN SỬ DỤNG GITHUB

▪ GitHub Desktop và GitHub Extension for Visual Studio:

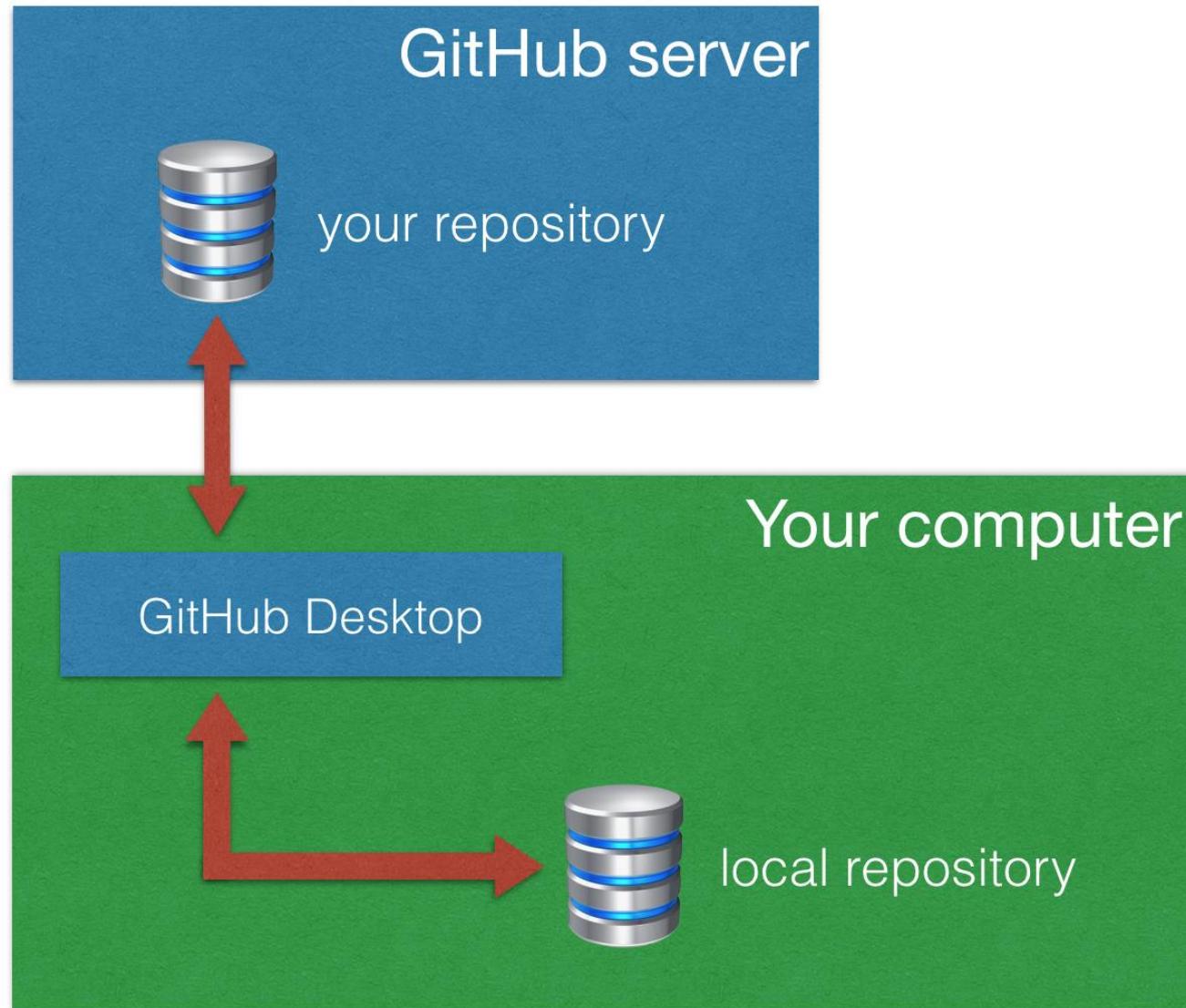
- Bên cạnh việc dùng web để quản lý các dự án, bạn có thể sử dụng phần mềm GitHub Desktop để đồng bộ với các dự án trên GitHub về máy của mình.
- Nếu bạn sử dụng Visual Studio 2015 trở lên, bạn có thể tải về GitHub Extension for Visual Studio. Ứng dụng này tích hợp trực tiếp vào giao diện Visual Studio và cũng hỗ trợ đầy đủ quy trình làm việc, quản lý dự án với GitHub ngay trên giao diện của Visual Studio.



HƯỚNG DẪN SỬ DỤNG GITHUB

▪ GitHub Desktop và GitHub Extension for Visual Studio:

- Một điểm cần lưu ý khi làm việc với GitHub Desktop và GitHub Extension (cũng như Git) đó là khi bạn sao chép (clone) một kho (repository) với GitHub thì kho đó sẽ được sao chép một bản vẽ trên máy của bạn và bản đó tồn tại độc lập với bản ở trên GitHub.
 - Thao tác này khác với tính năng đồng bộ thông thường mà chúng ta hay thấy. Do đó, mọi thao tác sửa đổi, lưu trạng thái (commit) sẽ chỉ được thực hiện ở máy của bạn. Chỉ đến khi bạn đồng bộ (publish hoặc sync) với máy chủ GitHub thì những sửa đổi và lần lưu trạng thái đó mới được chuyển lên GitHub.



GitHub Desktop.



QUY TRÌNH LÀM VIỆC VỚI GITHUB

1. Tạo một kho (repository) mới hoặc phân tách (fork) một kho đã có;
2. Tạo dự án trong tab Project, chuyển đổi các nhiệm vụ thành các vấn đề (issue);
3. Quản lý các vấn đề (issue), gán nhãn, gán mốc, gán người phụ trách;
4. Lựa chọn vấn đề (issue) cần giải quyết, kéo nhiệm vụ (task) liên quan sang cột In Progress;
5. Sao chép (clone) về máy tính hoặc đồng bộ (sync) lại kho;



QUY TRÌNH LÀM VIỆC VỚI GITHUB

6. (Nếu cần thiết) Tạo nhánh (branch) mới để giải quyết vấn đề
7. Chỉnh sửa và thực hiện lưu trạng thái (commit)
8. Tạo một yêu cầu gộp (pull request)
9. Kiểm duyệt yêu cầu gộp (review changes): chấp nhận gộp hoặc không chấp nhận gộp, đóng yêu cầu gộp
10. Trộn (merge) yêu cầu gộp vào nhánh chính
11. Xoá nhánh (branch) nếu muốn
12. Đóng vấn đề (issue) liên quan, kéo nhiệm vụ (task) liên quan sang cột Done
13. Quay lại bước 4.



**TRƯỜNG ĐẠI HỌC VINH
VIỆN KỸ THUẬT VÀ CÔNG NGHỆ**

CHƯƠNG 1

GIỚI THIỆU VỀ MÁY TÍNH VÀ LẬP TRÌNH

Nghệ An, 2018

Chương 1:

GIỚI THIỆU VỀ MÁY TÍNH VÀ LẬP TRÌNH



NỘI DUNG GIẢNG DẠY:

- 1.1. Phần cứng và phần mềm máy tính
- 1.2. Ngôn ngữ lập trình
- 1.3. Giải quyết vấn đề và phát triển phần mềm
- 1.4. Thuật toán



Chương 1:

GIỚI THIỆU VỀ MÁY TÍNH VÀ LẬP TRÌNH

NỘI DUNG GIẢNG DẠY:

1.1. Phần cứng và phần mềm máy tính

1.2. Ngôn ngữ lập trình

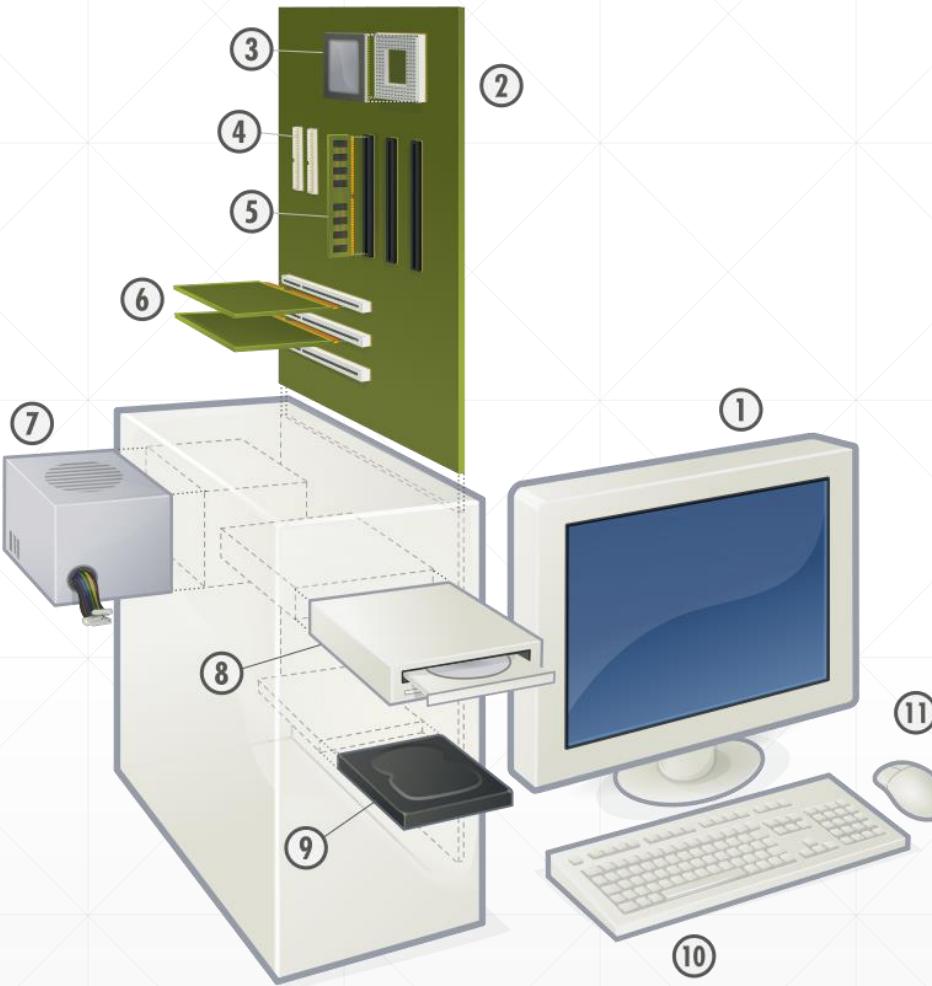
1.3. Giải quyết vấn đề và phát triển phần mềm

1.4. Thuật toán

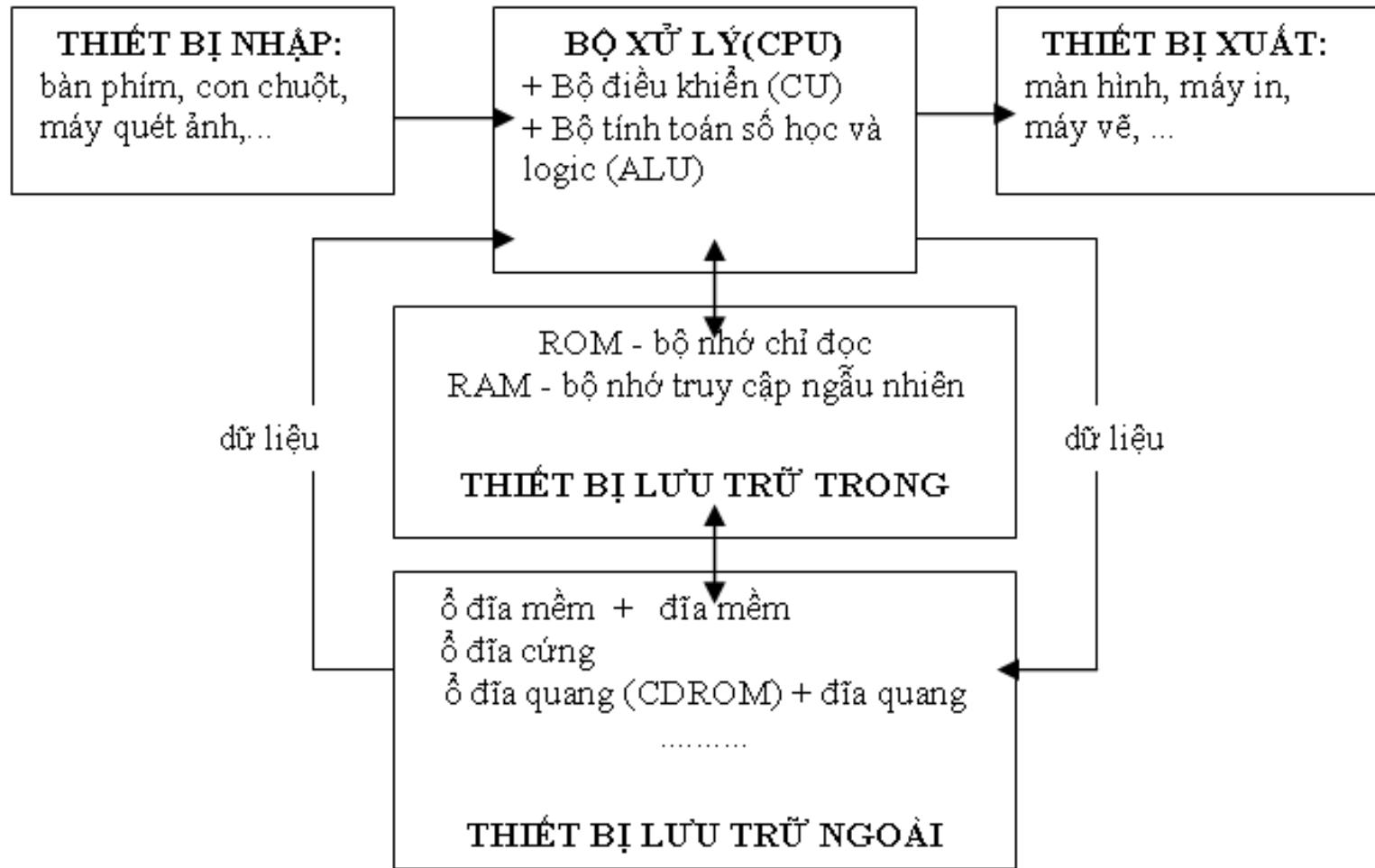


PHẦN CỨNG VÀ PHẦN MỀM MÁY TÍNH

Phần cứng (Hardware): là các cơ phận (vật lý) cụ thể của máy tính hay hệ thống máy tính như là màn hình, chuột, bàn phím, máy in, máy quét, vỏ máy tính, bộ nguồn, bộ vi xử lý CPU, bo mạch chủ, các loại dây nối, loa, ổ đĩa mềm, ổ đĩa cứng, ổ CDROM, ổ DVD, card đồ họa VGA, card wifi, card âm thanh, bộ phận tản nhiệt Cooler,...



Các thành phần chính của máy tính cá nhân để bàn



Các thành phần chính của máy tính cá nhân để bàn



PHẦN CỨNG VÀ PHẦN MỀM MÁY TÍNH

Phần mềm máy tính (Computer Software) hay gọi tắt là **Phần mềm (Software)** là một tập hợp những câu lệnh hoặc chỉ thị (Instruction) được viết bằng một hoặc nhiều ngôn ngữ lập trình theo một trật tự xác định, và các dữ liệu hay tài liệu liên quan nhằm tự động thực hiện một số nhiệm vụ hay chức năng hoặc giải quyết một vấn đề cụ thể nào đó

Phần mềm thực hiện các chức năng của nó bằng cách gửi các chỉ thị trực tiếp đến phần cứng (hay phần cứng máy tính) hoặc bằng cách cung cấp dữ liệu để phục vụ các chương trình hay phần mềm khác.



PHẦN CỨNG VÀ PHẦN MỀM MÁY TÍNH

Phần mềm là một khái niệm trừu tượng, nó khác với phần cứng ở chỗ là "phần mềm không thể sờ hay đụng vào", và nó cần phải có phần cứng mới có thể thực thi được.

Chương trình máy tính thường được tạo ra bởi con người, những người này được gọi là **lập trình viên**, tuy nhiên cũng tồn tại những chương trình được sinh ra bởi các chương trình khác.

Chương 1: GIỚI THIỆU VỀ MÁY TÍNH VÀ LẬP TRÌNH



NỘI DUNG GIẢNG DẠY:

- 1.1. Phần cứng và phần mềm máy tính
- 1.2. Ngôn ngữ lập trình**
- 1.3. Giải quyết vấn đề và phát triển phần mềm
- 1.4. Thuật toán



NGÔN NGỮ LẬP TRÌNH

Ngôn ngữ lập trình là một tập con của ngôn ngữ máy tính, được thiết kế và chuẩn hóa để truyền các chỉ thị cho các máy có bộ xử lý (CPU), nói riêng là máy tính.

Ngôn ngữ lập trình được dùng để lập trình máy tính, tạo ra các chương trình máy nhằm mục đích điều khiển máy tính hoặc mô tả các thuật toán để người khác đọc hiểu.



NGÔN NGỮ LẬP TRÌNH

Một ngôn ngữ lập trình phải thỏa mãn được hai điều kiện cơ bản sau:

- Dễ hiểu và dễ sử dụng đối với người lập trình, để có thể dùng để giải quyết nhiều bài toán khác nhau.
- Miêu tả một cách đầy đủ và rõ ràng các tiến trình (tiếng Anh: process), để chạy được trên các hệ máy tính khác nhau.



NGÔN NGỮ LẬP TRÌNH

Trước đây, để tạo ra chương trình máy tính người ta phải làm việc trực tiếp với các con số 0 hoặc 1, hay còn gọi là ngôn ngữ máy. Công việc này vô cùng khó khăn, chiếm nhiều thời gian, công sức và đặc biệt dễ gây ra lỗi.

Để khắc phục nhược điểm này, người ta đề xuất ra hợp ngữ, một ngôn ngữ cho phép thay thế dãy 0 hoặc 1 này bởi các từ gợi nhớ. Tuy nhiên, cải tiến này vẫn còn chưa thật thích hợp với đa số người dùng.

Từ những năm 1950, người ta đã xây dựng những ngôn ngữ lập trình mà câu lệnh của nó gần với ngôn ngữ tự nhiên. Các ngôn ngữ này được gọi là ngôn ngữ lập trình bậc cao.



NGÔN NGỮ LẬP TRÌNH

Một tập hợp các chỉ thị được biểu thị qua ngôn ngữ lập trình nhằm mục đích thực hiện các thao tác máy tính nào đó được gọi là một **chương trình**. Khái niệm này còn có những tên khác như **chương trình máy tính** hay **chương trình điện toán**.

Khái niệm **lập trình** dùng để chỉ quá trình con người tạo ra chương trình máy tính thông qua ngôn ngữ lập trình. Người ta còn gọi đó là quá trình **mã hóa thông tin** tự nhiên thành ngôn ngữ máy.

Văn bản được viết bằng ngôn ngữ lập trình để tạo nên chương trình được gọi là **mã nguồn**.



NGÔN NGỮ LẬP TRÌNH

Thao tác chuyển đổi từ **mã nguồn** thành **chuỗi các chỉ thị máy tính** được thực hiện tương tự như việc chuyển đổi qua lại giữa các ngôn ngữ tự nhiên của con người.

Các thao tác này gọi là biên dịch, hay ngắn gọn hơn là dịch:

- Nếu quá trình dịch diễn ra đồng thời với quá trình thực thi, ta gọi đó là thông dịch.
- Nếu diễn ra trước, ta gọi đó là biên dịch.



NGÔN NGỮ LẬP TRÌNH

ĐẶC ĐIỂM CHUNG CỦA NGÔN NGỮ LẬP TRÌNH:

Mỗi ngôn ngữ lập trình có thể được xem như là một tập hợp của các chi tiết kỹ thuật chú trọng đến cú pháp, từ vựng, và ý nghĩa của ngôn ngữ. Những chi tiết kỹ thuật này thường bao gồm:

- Dữ liệu và cấu trúc dữ liệu;
- Câu lệnh và dòng điều khiển;
- Các tên và các tham số;
- Các cơ chế tham khảo và sự tái sử dụng.



NGÔN NGỮ LẬP TRÌNH

Đối với các ngôn ngữ phổ biến hoặc có lịch sử lâu dài, người ta thường tổ chức các hội thảo chuẩn hóa nhằm tạo ra và công bố các tiêu chuẩn chính thức cho ngôn ngữ đó, cũng như thảo luận về việc mở rộng, bổ sung cho các tiêu chuẩn trước đó.

Ví dụ: Với ngôn ngữ C++, hội đồng tiêu chuẩn ANSI C++ và ISO C++ đã tổ chức đến 13 cuộc hội thảo để điều chỉnh và nâng cấp ngôn ngữ này.



THẢO LUẬN NHÓM

NỘI DUNG:

1. Phân biệt giữa ngôn ngữ máy, hợp ngữ và ngôn ngữ lập trình cấp cao.
2. Giới thiệu về một số ngôn ngữ lập trình thông dụng hiện nay về các nội dung:
 - Lịch sử phát triển;
 - Đặc điểm;
 - Ứng dụng.



BÀI TẬP

NỘI DUNG:

1. Tạo tài khoản trên trang web <https://github.com/>.
2. Tải các tài liệu của học phần đã được giới thiệu.

CHUẨN BỊ CHO BUỔI HỌC TIẾP THEO:

1. Đọc các tài liệu về nội dung mục 1.3 và 1.4.
2. Nghiên cứu về khái niệm **mã giả**, cách sử dụng **mã giả**.



Chương 1: **GIỚI THIỆU VỀ MÁY TÍNH VÀ LẬP TRÌNH**

NỘI DUNG GIẢNG DẠY:

- 1.1. Phần cứng và phần mềm máy tính
- 1.2. Ngôn ngữ lập trình
- 1.3. Giải quyết vấn đề và phát triển phần mềm**
- 1.4. Thuật toán

GIẢI QUYẾT VẤN ĐỀ VÀ PHÁT TRIỂN PHẦN MỀM



CÁC MÔ HÌNH PHÁT TRIỂN SẢN PHẨM PHẦN MỀM

- Quá trình phát triển phần mềm là tập hợp các thao tác và các kết quả tương quan để sản xuất ra một sản phẩm phần mềm.
- Hầu hết các thao tác này được tiến hành bởi các kỹ sư phần mềm.
- Các công cụ hỗ trợ máy tính về kỹ thuật phần mềm có thể được dùng để giúp trong một số thao tác.

GIẢI QUYẾT VẤN ĐỀ VÀ PHÁT TRIỂN PHẦN MỀM



Có 4 thao tác nền tảng của hầu hết các quá trình phần mềm:

- + Đặc tả phần mềm: Các chức năng của phần mềm và điều kiện để nó hoạt động phải được định nghĩa.
- + Phát triển phần mềm: Để phần mềm đạt được đặc tả thì phải có quá trình phát triển này.
- + Đánh giá phần mềm: Phần mềm phải được đánh giá để chắc chắn rằng nó làm những gì mà khách hàng muốn.
- + Sự tiến hóa của phần mềm: Phần mềm phải tiến hóa để thỏa mãn sự thay đổi các yêu cầu của khách hàng.

GIẢI QUYẾT VẤN ĐỀ VÀ PHÁT TRIỂN PHẦN MỀM



Có nhiều loại mô hình phát triển phần mềm khác nhau:

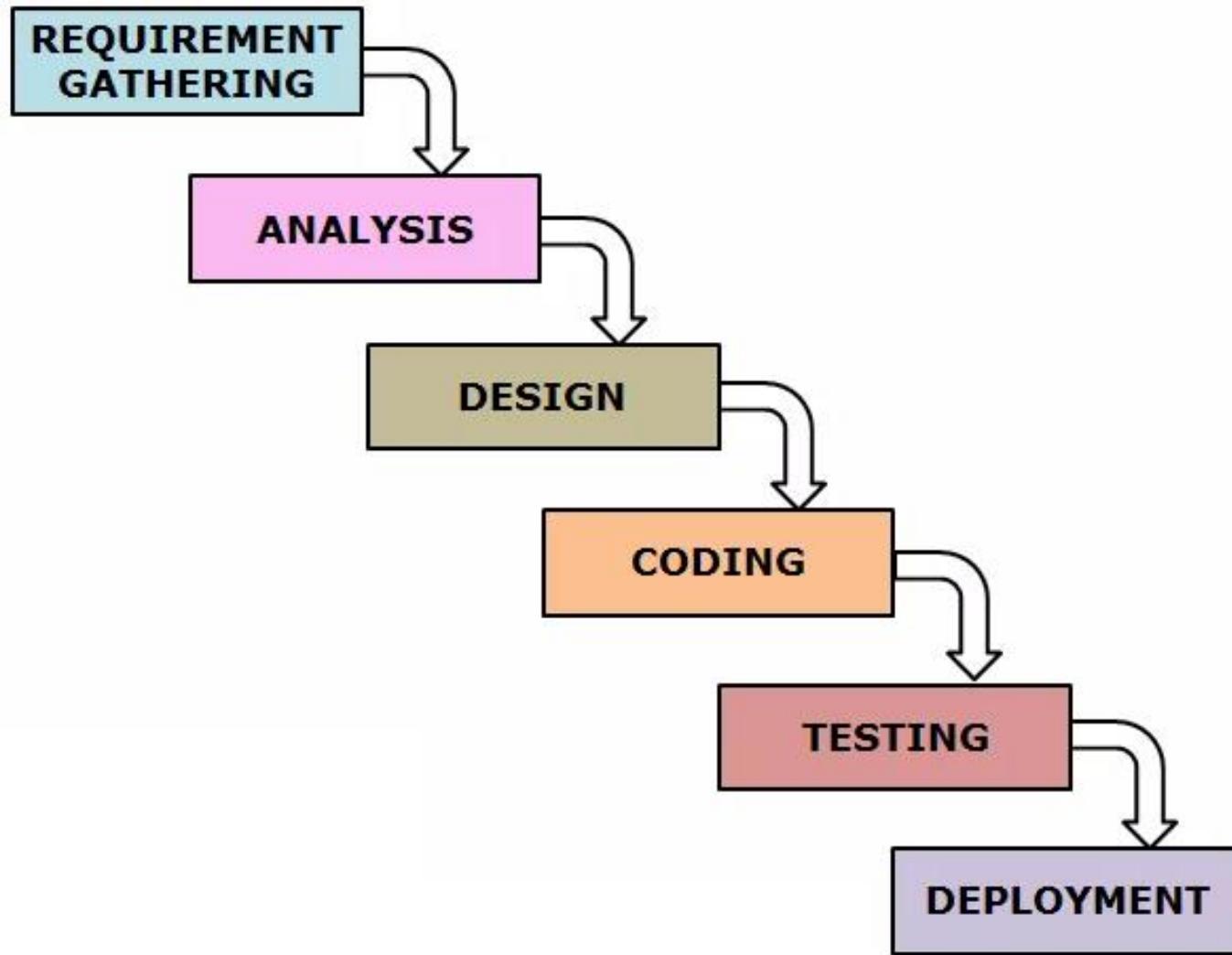
- Mô hình thác nước (Waterfall model)
- Mô hình xoắn ốc (Spiral model)
- Mô hình agile
- Mô hình tiếp cận lặp (Iterative model)
- Mô hình tăng trưởng (Incremental model)
- Mô hình chữ V (V model)
- Mô hình Scrum
- RAD model (Rapid Application Development)

GIẢI QUYẾT VẤN ĐỀ VÀ PHÁT TRIỂN PHẦN MỀM



MÔ HÌNH THÁC NƯỚC (WATERFALL MODEL):

- Đây được coi như là mô hình phát triển phần mềm đầu tiên được sử dụng.
- Mô hình này áp dụng tuần tự các giai đoạn của phát triển phần mềm.
- Đầu ra của giai đoạn trước là đầu vào của giai đoạn sau. Giai đoạn sau chỉ được thực hiện khi giai đoạn trước đã kết thúc. Đặc biệt không được quay lại giai đoạn trước để xử lý các yêu cầu khi muốn thay đổi.



Mô hình thác nước (Waterfall model)

GIẢI QUYẾT VẤN ĐỀ VÀ PHÁT TRIỂN PHẦN MỀM



MÔ HÌNH THÁC NƯỚC (WATERFALL MODEL):

• Ứng dụng

Mô hình thường được áp dụng cho các dự án phần mềm như sau:

- Các dự án nhỏ, ngắn hạn.
- Các dự án có ít thay đổi về yêu cầu và không có những yêu cầu không rõ ràng.

GIẢI QUYẾT VẤN ĐỀ VÀ PHÁT TRIỂN PHẦN MỀM



MÔ HÌNH THÁC NƯỚC (WATERFALL MODEL):

- **Ưu điểm**

- Dễ sử dụng, dễ tiếp cận, dễ quản lý.
- Sản phẩm phát triển theo các giai đoạn được xác định rõ ràng.
- Xác nhận ở từng giai đoạn, đảm bảo phát hiện sớm các lỗi.

GIẢI QUYẾT VẤN ĐỀ VÀ PHÁT TRIỂN PHẦN MỀM



MÔ HÌNH THÁC NƯỚC (WATERFALL MODEL):

- **Nhược điểm**

- Ít linh hoạt, phạm vi điều chỉnh hạn chế.
- Rất khó để đo lường sự phát triển trong từng giai đoạn.
- Mô hình không thích hợp với những dự án dài, đang diễn ra, hay những dự án phức tạp, có nhiều thay đổi về yêu cầu trong vòng đời phát triển.
- Khó quay lại khi giai đoạn nào đó đã kết thúc.

GIẢI QUYẾT VẤN ĐỀ VÀ PHÁT TRIỂN PHẦN MỀM

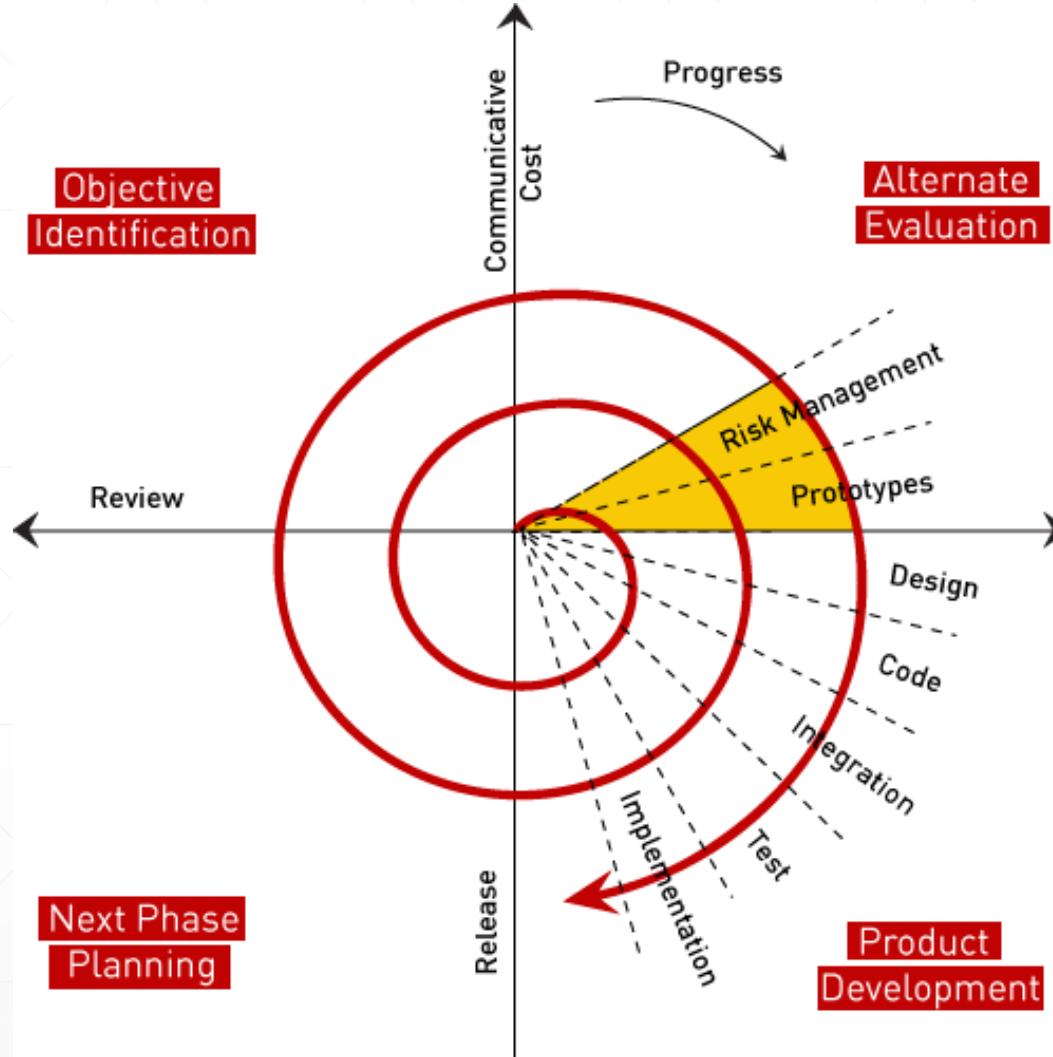


MÔ HÌNH XOĂN ỐC (SPIRAL MODEL):

Là mô hình kết hợp giữa các tính năng của mô hình prototyping và mô hình thác nước.

Mô hình xoắn ốc được ưa chuộng cho các dự án lớn, đắt tiền và phức tạp.

Mô hình này sử dụng những giai đoạn tương tự như mô hình thác nước, vẽ thứ tự, plan, đánh giá rủi ro, ...



Mô hình xoắn ốc (Spiral model)

GIẢI QUYẾT VẤN ĐỀ VÀ PHÁT TRIỂN PHẦN MỀM



MÔ HÌNH XOĂN ỐC (SPIRAL MODEL):

Ứng dụng

Mô hình này thường được sử dụng cho các ứng dụng lớn và các hệ thống được xây dựng theo các giai đoạn nhỏ hoặc theo các phân đoạn.

GIẢI QUYẾT VẤN ĐỀ VÀ PHÁT TRIỂN PHẦN MỀM



MÔ HÌNH XOẮN ỐC (SPIRAL MODEL):

Ưu điểm

- Tốt cho các hệ phần mềm quy mô lớn.
- Dễ kiểm soát các mạo hiểm ở từng mức tiến hóa.
- Đánh giá thực tế hơn như là một quy trình làm việc, bởi vì những vấn đề quan trọng đã được phát hiện sớm hơn.

GIẢI QUYẾT VẤN ĐỀ VÀ PHÁT TRIỂN PHẦN MỀM



MÔ HÌNH XOẮN ỐC (SPIRAL MODEL):

Nhược điểm

- Manager cần có kỹ năng tốt để quản lý dự án, đánh giá rủi ro kịp thời.
- Chi phí cao và mất nhiều thời gian để hoàn thành dự án.
- Phức tạp và không thích hợp với các dự án nhỏ và ít rủi ro.
- Yêu cầu thay đổi thường xuyên dẫn đến lặp vô hạn.
- Chưa được dùng rộng rãi.

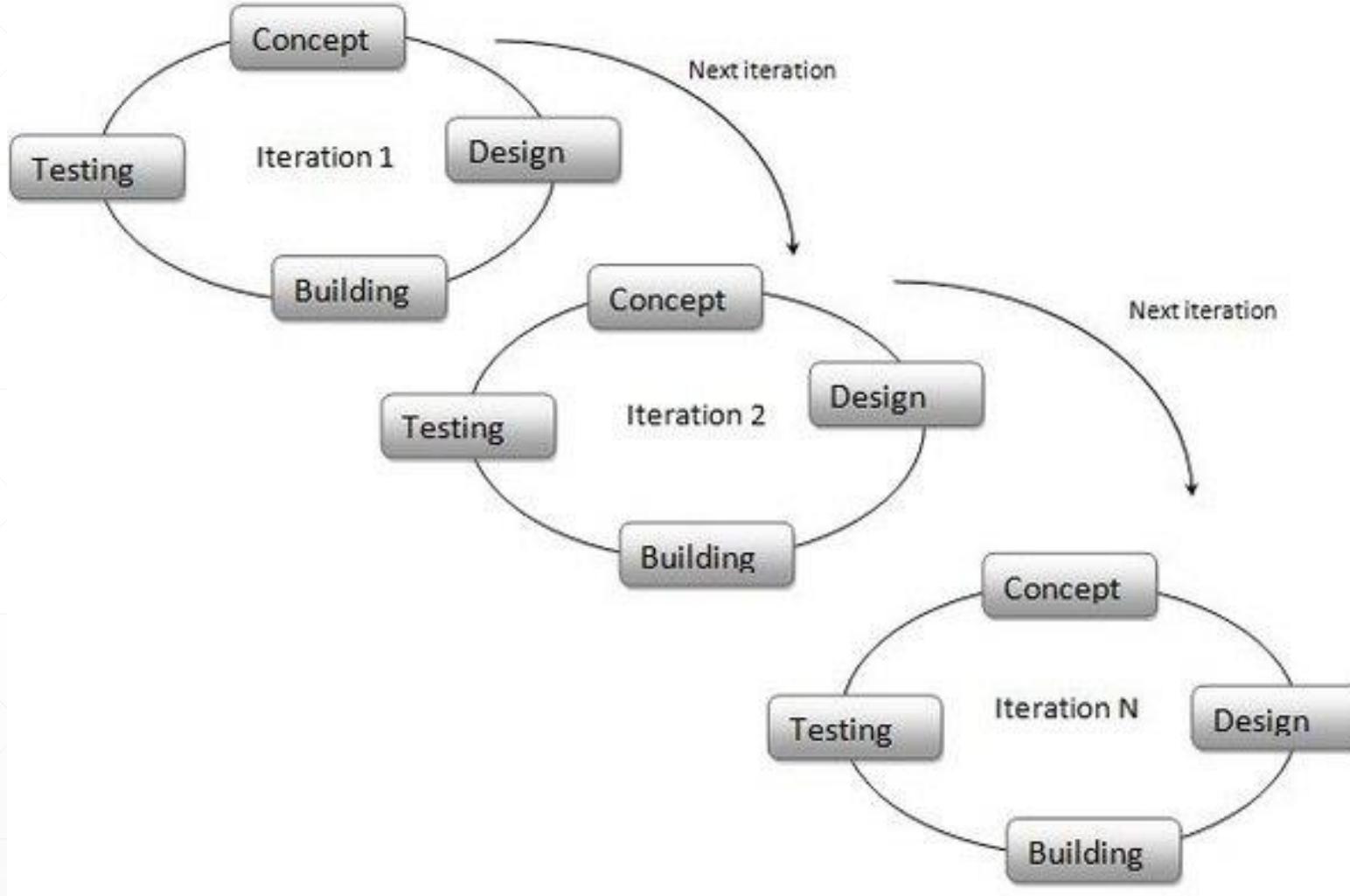
GIẢI QUYẾT VẤN ĐỀ VÀ PHÁT TRIỂN PHẦN MỀM



MÔ HÌNH AGILE:

Agile là một phương pháp phát triển phần mềm linh hoạt để làm sao đưa sản phẩm đến tay người dùng càng nhanh càng tốt và được xem như là sự cải tiến so với những mô hình cũ như mô hình "Thác nước (waterfall)" hay "CMMI".

Phương thức phát triển phần mềm Agile là một tập hợp các phương thức phát triển lặp và tăng dần trong đó các yêu cầu và giải pháp được phát triển thông qua sự liên kết cộng tác giữa các nhóm tự quản và liên chức năng.



Mô hình Agile

GIẢI QUYẾT VẤN ĐỀ VÀ PHÁT TRIỂN PHẦN MỀM



MÔ HÌNH AGILE: Ứng dụng

Có thể được sử dụng với bất kỳ loại hình dự án nào, nhưng cần sự tham gia và tính tương tác của khách hàng.

Sử dụng khi khách hàng yêu cầu chức năng sẵn sàng trong khoảng thời gian ngắn.

GIẢI QUYẾT VẤN ĐỀ VÀ PHÁT TRIỂN PHẦN MỀM



MÔ HÌNH AGILE:

Ưu điểm

- Tăng cường tinh thần làm việc nhóm và trao đổi công việc hiệu quả.
- Các chức năng được xây dựng nhanh chóng và rõ ràng, dễ quản lý.
- Dễ dàng bổ sung, thay đổi yêu cầu.
- Quy tắc tối thiểu, tài liệu dễ hiểu, dễ sử dụng.

GIẢI QUYẾT VẤN ĐỀ VÀ PHÁT TRIỂN PHẦN MỀM



MÔ HÌNH AGILE:

Nhược điểm

- Không thích hợp để xử lý các phụ thuộc phức tạp.
- Có nhiều rủi ro về tính bền vững, khả năng bảo trì và khả năng mở rộng.
- Cần một team có kinh nghiệm.
- Phụ thuộc rất nhiều vào sự tương tác rõ ràng của khách hàng.
- Chuyển giao công nghệ cho các thành viên mới trong nhóm có thể khá khó khăn do thiếu tài liệu.

Chương 1:

GIỚI THIỆU VỀ MÁY TÍNH VÀ LẬP TRÌNH



NỘI DUNG GIẢNG DẠY:

- 1.1. Phần cứng và phần mềm máy tính
- 1.2. Ngôn ngữ lập trình
- 1.3. Giải quyết vấn đề và phát triển phần mềm
- 1.4. Thuật toán**



THUẬT TOÁN

Thuật toán, còn gọi là **giải thuật**, là một tập hợp hữu hạn của các chỉ thị hay phương cách được định nghĩa rõ ràng cho việc hoàn tất một số sự việc từ một trạng thái ban đầu cho trước; khi các chỉ thị này được áp dụng triệt để thì sẽ dẫn đến kết quả sau cùng như đã dự đoán.

Nói cách khác, thuật toán là một bộ các qui tắc hay qui trình cụ thể nhằm giải quyết một vấn đề trong một số bước hữu hạn, hoặc nhằm cung cấp một kết quả từ một tập hợp của các dữ kiện đưa vào.



THUẬT TOÁN

Ví dụ: Thuật toán để giải phương trình bậc nhất $P(x)$: $ax + b = c$, (a, b, c là các số thực), trong tập hợp các số thực có thể là một bộ các bước sau đây:

1. Nếu $a = 0$

- $b = c$ thì $P(x)$ có nghiệm bất kì
- $b \neq c$ thì $P(c)$ vô nghiệm

2. Nếu $a \neq 0$

- $P(x)$ có duy nhất một nghiệm $x = (c - b)/a$



THUẬT TOÁN

Lưu ý:

Khi một thuật toán đã hình thành thì ta không xét đến việc chứng minh thuật toán đó mà chỉ chú trọng đến việc áp dụng các bước theo sự hướng dẫn sẽ có kết quả đúng.

Việc chứng minh tính đầy đủ và tính đúng của các thuật toán phải được tiến hành xong trước khi có thuật toán. Nói rõ hơn, thuật toán có thể chỉ là việc áp dụng các công thức hay qui tắc, qui trình đã được công nhận là đúng hay đã được chứng minh về mặt toán học.



THUẬT TOÁN

"Thuật toán" hiện nay thường được dùng để chỉ thuật toán giải quyết các vấn đề tin học. Hầu hết các thuật toán tin học đều có thể viết thành các chương trình máy tính mặc dù chúng thường có một vài hạn chế (khả năng của máy tính, khả năng của người lập trình).

Trong nhiều trường hợp, một chương trình khi thiết kế bị thắt bại là do lỗi ở các thuật toán mà người lập trình đưa vào là không chính xác, không đầy đủ, hay không ước định được trọn vẹn lời giải vấn đề.

Tuy nhiên cũng có một số bài toán mà hiện nay người ta chưa tìm được lời giải triệt để, những bài toán ấy gọi là những bài toán NP - không đầy đủ.



THUẬT TOÁN

Tính chất của thuật toán:

- Tính chính xác: để đảm bảo kết quả tính toán hay các thao tác mà máy tính thực hiện được là chính xác.
- Tính rõ ràng: Thuật toán phải được thể hiện bằng các câu lệnh minh bạch; các câu lệnh được sắp xếp theo thứ tự nhất định.
- Tính khách quan: Một thuật toán dù được viết bởi nhiều người trên nhiều máy tính vẫn phải cho kết quả như nhau.



THUẬT TOÁN

- Tính phổ dụng: Thuật toán không chỉ áp dụng cho một bài toán nhất định mà có thể áp dụng cho một lớp các bài toán có đầu vào tương tự nhau.
- Tính kết thúc: Thuật toán phải gồm một số hữu hạn các bước tính toán.



THUẬT TOÁN

Độ phức tạp thuật toán:

Thời gian mà máy tính khi thực hiện một thuật toán không chỉ phụ thuộc vào bản thân thuật toán đó, ngoài ra còn tùy thuộc từng máy tính. Để đánh giá hiệu quả của một thuật toán, có thể xét số các phép tính phải thực hiện khi thực hiện thuật toán này.

Thông thường số các phép tính được thực hiện phụ thuộc vào cỡ của bài toán, tức là độ lớn của đầu vào. Vì thế độ phức tạp thuật toán là một hàm phụ thuộc đầu vào. Tuy nhiên trong những ứng dụng thực tiễn, chúng ta không cần biết chính xác hàm này mà chỉ cần biết một ước lượng đủ tốt của chúng.



THẢO LUẬN NHÓM

NỘI DUNG:

1. Xác định vai trò của thuật toán trong lập trình.
2. Khái niệm về mã giả và cách sử dụng mã giả;
3. Ứng dụng viết mã giả để giải quyết các vấn đề:
 - Giải phương trình bậc nhất;
 - Giải phương trình bậc 2.



BÀI TẬP

NỘI DUNG:

1. Tìm hiểu về phần mềm Flowgorithm.
2. Tải và cài đặt phần mềm: Flowgorithm.
3. Sử dụng phần mềm Flowgorithm để thiết kế các thuật toán:
 - Giải phương trình bậc nhất;
 - Giải phương trình bậc 2;
 - Giải thuật tìm dãy số Fibonacci.

Báo cáo kết quả thực hiện lên github.com.



BÀI TẬP

CHUẨN BỊ CHO BUỔI HỌC TIẾP THEO:

1. Đọc các tài liệu về nội dung mục 2.1; 2.2 và 2.3.
2. Nghiên cứu về ngôn ngữ lập trình Python và đặc điểm của ngôn ngữ lập trình Python.



**TRƯỜNG ĐẠI HỌC VINH
VIỆN KỸ THUẬT VÀ CÔNG NGHỆ**

CHƯƠNG 2

NGÔN NGỮ LẬP TRÌNH PYTHON

Nghệ An, 2018

Chương 2:

NGÔN NGỮ LẬP TRÌNH PYTHON



NỘI DUNG GIẢNG DẠY:

- 2.1. Một số đặc điểm của ngôn ngữ lập trình Python
- 2.2. Môi trường lập trình và thực thi Python
- 2.3. Cú pháp cơ bản của Python
- 2.4. Biến và các toán tử trong Python
- 2.5. Các cấu trúc điều khiển

Chương 2:

NGÔN NGỮ LẬP TRÌNH PYTHON



NỘI DUNG GIẢNG DẠY:

2.1. Một số đặc điểm của ngôn ngữ lập trình Python

2.2. Môi trường lập trình và thực thi Python

2.3. Cú pháp cơ bản của Python

2.4. Biến và các toán tử trong Python

2.5. Các cấu trúc điều khiển

ĐẶC ĐIỂM CỦA NGÔN NGỮ LẬP TRÌNH PYTHON



Python là gì?

Python là một ngôn ngữ lập trình thông dịch (interpreted), hướng đối tượng (object-oriented), và là một ngôn ngữ bậc cao (high-level) ngữ nghĩa động (dynamic semantics).

Python hỗ trợ các module và gói (packages), khuyến khích chương trình module hóa và tái sử dụng mã.

Trình thông dịch Python và thư viện chuẩn mở rộng có sẵn dưới dạng mã nguồn hoặc dạng nhị phân miễn phí cho tất cả các nền tảng chính và có thể được phân phối tự do.

ĐẶC ĐIỂM CỦA NGÔN NGỮ LẬP TRÌNH PYTHON



Các đặc điểm của Python:

- Ngữ pháp đơn giản, dễ đọc.
- Vừa hướng thủ tục (procedural-oriented), vừa hướng đối tượng (object-oriented)
- Hỗ trợ module và hỗ trợ gói (package)
- Xử lý lỗi bằng ngoại lệ (Exception)
- Kiểu dữ liệu động ở mức cao.
- Có các bộ thư viện chuẩn và các module ngoài, đáp ứng tất cả các nhu cầu lập trình.

ĐẶC ĐIỂM CỦA NGÔN NGỮ LẬP TRÌNH PYTHON



- Có khả năng tương tác với các module khác viết trên C/C++ (Hoặc Java cho Jython, hoặc .Net cho IronPython).
- Có thể nhúng vào ứng dụng như một giao tiếp kịch bản (scripting interface).

ĐẶC ĐIỂM CỦA NGÔN NGỮ LẬP TRÌNH PYTHON



Lịch sử của Python:

Python đã được hình thành vào cuối những năm 1980, và việc thực hiện nó vào tháng 12 năm 1989 bởi Guido van Rossum tại Centrum Wiskunde & Informatica (CWI) ở Hà Lan như là một kế thừa cho ngôn ngữ ABC (tự lấy cảm hứng từ SETL) có khả năng xử lý ngoại lệ và giao tiếp với hệ điều hành Amoeba.

Van Rossum là tác giả chính của Python, và vai trò trung tâm của ông trong việc quyết định hướng phát triển của Python.

ĐẶC ĐIỂM CỦA NGÔN NGỮ LẬP TRÌNH PYTHON



Về nguồn gốc của Python, **Van Rossum** đã viết vào năm 1996:

Hơn sáu năm trước, vào tháng 12 năm 1989, tôi đã tìm kiếm một dự án lập trình "sở thích" mà nó đã chiếm đóng tâm trí tôi trong suốt tuần lễ Giáng sinh. Văn phòng của tôi ... sẽ đóng cửa, nhưng tôi đã có một máy tính ở nhà, và không có nhiều thứ khác trên tay. Tôi quyết định viết một bộ thông dịch (interpreter) cho ngôn ngữ kịch bản mới mà tôi đã từng nghĩ đến: một hậu duệ của ABC có thể hấp dẫn các hacker Unix/C. Tôi đã chọn Python như là một tiêu đề làm việc cho dự án.



ĐẶC ĐIỂM CỦA NGÔN NGỮ LẬP TRÌNH PYTHON



- Python 2.0 đã được phát hành vào ngày 16 tháng 10 năm 2000 và có nhiều tính năng mới, bao gồm bộ thu gom rác theo chu kỳ (cycle-detecting garbage) và hỗ trợ Unicode. Với việc phát hành này quá trình phát triển đã được thay đổi và trở nên minh bạch hơn và cộng đồng hậu thuẫn.
- Python 3.0 được phát hành năm 2008, sau một thời gian dài thử nghiệm.
- Cho tới năm 2018, Python đang có phiên bản 3.7.

Language Rank	Types	Spectrum Ranking
1. Python		100.0
2. C++		99.7
3. Java		97.5
4. C		96.7
5. C#		89.4
6. PHP		84.9
7. R		82.9
8. JavaScript		82.6
9. Go		76.4
10. Assembly		74.1

The 2018 Top Programming Languages

Chương 2: NGÔN NGỮ LẬP TRÌNH PYTHON



NỘI DUNG GIẢNG DẠY:

2.1. Một số đặc điểm của ngôn ngữ lập trình Python

2.2. Môi trường lập trình và thực thi Python

2.3. Cú pháp cơ bản của Python

2.4. Biến và các toán tử trong Python

2.5. Các cấu trúc điều khiển



MÔI TRƯỜNG LẬP TRÌNH VÀ THỰC THI PYTHON

Python là một ngôn ngữ lập trình dạng thông dịch, do đó có ưu điểm tiết kiệm thời gian phát triển ứng dụng vì không cần phải thực hiện biên dịch và liên kết.

Trình thông dịch có thể được sử dụng để chạy file script, hoặc cũng có thể được sử dụng theo cách tương tác.

Ở chế độ tương tác, ta có thể nhập vào từng biểu thức rồi go Enter, và kết quả thực thi sẽ được hiển thị ngay lập tức. Đặc điểm này rất hữu ích cho người mới học, giúp họ nghiên cứu tính năng của ngôn ngữ; hoặc để các lập trình viên chạy thử mã lệnh trong suốt quá trình phát triển phần mềm.



MÔI TRƯỜNG LẬP TRÌNH VÀ THỰC THI PYTHON

Tùy vào hệ điều hành, việc cài đặt Python có những sự khác nhau tương đối:

- Người dùng Windows có thể tự tải về và chạy file cài đặt trên máy tính với một vài bước cài đặt đơn giản.
- Người dùng Linux và MacOS X có thể dùng các bộ Python đã được cài đặt sẵn theo hệ điều hành (Python là thành phần có sẵn trong Linux và MacOS X).

Trên mỗi nền tảng có một dạng khác nhau của Python. Từ máy tính, tới điện thoại, web, game... tuy nhiên chúng vẫn có những đặc điểm phổ biến giống nhau về mặt cơ bản.



MÔI TRƯỜNG LẬP TRÌNH VÀ THỰC THI PYTHON

Quá trình thực thi của Python

Tùy thuộc vào chiều hướng bạn nhìn vào mà Python được chia làm hai dạng: từ phía người lập trình và từ phía trình thông dịch Python.

- Từ phía người lập trình:

Ở dạng đơn giản, Python chỉ là 1 file văn bản với những câu lệnh Python được viết bằng bất cứ trình soạn thảo nào, miễn là đúng cú pháp quy định và lưu dưới dạng *.py.

Việc chạy cũng vô cùng đơn giản, chỉ đơn giản là thực thi toàn bộ các câu lệnh trong file *.py lần lượt từ trên xuống dưới.



MÔI TRƯỜNG LẬP TRÌNH VÀ THỰC THI PYTHON

- **Từ phía trình biên dịch Python:** dưới góc nhìn của trình biên dịch, quá trình thực thi file Python trải qua ba bước:

B1 - *Viết code Python:* phần này đã giải thích ở trước.

B2 - *Biên dịch sang bytecode:* các câu lệnh sẽ được Python biên dịch sang một dạng thấp hơn đó là dạng bytecode. Thông thường thì quá trình này được thực thi nội tại và không thể thấy (trong một số trường hợp, có thể thấy một file với phần mở rộng là *.pyc).

Để tăng tốc quá trình làm việc, Python sẽ lưu cái file bytecode này và ở lần thực thi tiếp theo, nó sẽ chạy trực tiếp cái file này thay vì phải biên dịch lại.



MÔI TRƯỜNG LẬP TRÌNH VÀ THỰC THI PYTHON

B3 - Máy ảo Python:

Sau khi biên dịch ra bytecode, các file sẽ được thực thi bởi một máy ảo Python (Python Virtual Machine).

Máy ảo Python đơn giản là một vòng lặp lớn mà ở đó bytecode sẽ được thực thi lần lượt. Đây mới là phần thực sự thi hành các lệnh của chương trình. Về mặt kỹ thuật thì đây là bước cuối cùng trong quá trình thông dịch Python.



GIỚI THIỆU PYCHARM IDE

PyCharm được phát triển bởi Jet Brains, cung cấp cho người dùng bản Community miễn phí, dùng thử 30 ngày cho phiên bản chuyên nghiệp, \$213 – \$690 phí đăng ký hàng năm.

Khả năng hỗ trợ code toàn diện và phân tích làm cho PyCharm là IDE tốt nhất cho các lập trình Python tất cả các cấp độ.

PyCharm cũng hỗ trợ các ngôn ngữ khác và hoạt động trên nhiều nền tảng, vì vậy thực tế bất cứ ai cũng có thể sử dụng nó.



GIỚI THIỆU PYCHARM IDE

A detailed screenshot of the PyCharm interface. The code editor shows a Python script named "blueprintexample.py" with the following content:

```
from flask import Flask
from simple_page.simple_page import simple_page

app = Flask(__name__)
app.register_blueprint(simple_page)
# Blueprint can be registered many times
app.register_blueprint(simple_page, url_prefix='/pages')

if __name__ == '__main__':
    app.run(host='0.0.0.0')
```

The PyCharm interface includes several toolbars and panels: a top menu bar with File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help; a Project panel on the left showing a "blueprintexample" project with files like ".vagrant", "simple_page", "blueprintexample.py", "test_blueprintexample.py", and "Vagrantfile"; a Structure panel below it; a Variables panel in the bottom left showing variables like "app", "simple_page", and "Flask"; a Watches panel in the bottom right showing "No watches"; and a Debug toolbar at the bottom with buttons for Debugger, Console, Run, Stop, and Step. The status bar at the very bottom shows tabs for Run, Debug, TODO, Python Console, Terminal, and Event Log.

Chương 2: NGÔN NGỮ LẬP TRÌNH PYTHON



NỘI DUNG GIẢNG DẠY:

- 2.1. Một số đặc điểm của ngôn ngữ lập trình Python
- 2.2. Môi trường lập trình và thực thi Python
- 2.3. Cú pháp cơ bản của Python**
- 2.4. Biến và các toán tử trong Python
- 2.5. Các cấu trúc điều khiển



CÚ PHÁP CƠ BẢN CỦA PYTHON

• **Định danh (identifier) trong Python**

- Một định danh (identifier) trong Python là một tên được sử dụng để nhận diện một biến, một hàm, một lớp, hoặc một đối tượng.
- Một định danh bắt đầu với một chữ cái từ A tới Z hoặc từ a tới z hoặc một dấu gạch dưới (_) được sau bởi không hoặc nhiều ký tự, dấu gạch dưới hoặc các chữ số (từ 0 tới 9).
- Python không hỗ trợ các punctuation char chẵng hạn như @, \$ và % bên trong các định danh.
- Python là ngôn ngữ lập trình phân biệt chữ hoa - chữ thường.



CÚ PHÁP CƠ BẢN CỦA PYTHON

Một số qui tắc nên được sử dụng trong khi đặt tên các định danh:

- Một định danh là một dãy ký tự hoặc chữ số.
- Không có ký tự đặc biệt nào được sử dụng (ngoại trừ dấu gạch dưới) như một định danh.
- Ký tự đầu tiên có thể là chữ cái, dấu gạch dưới, nhưng không được sử dụng chữ số làm ký tự đầu tiên.
- Từ khóa không nên được sử dụng như là một tên định danh (*phản tiếp theo sẽ trình bày về các từ khóa này*).



CÚ PHÁP CƠ BẢN CỦA PYTHON

- Tên lớp bắt đầu với một chữ cái hoa. Tất cả định danh khác bắt đầu với một chữ cái thường.
- Bắt đầu một định danh với một dấu gạch dưới đơn chỉ rằng định danh đó là private.
- Bắt đầu một định danh với hai dấu gạch dưới chỉ rằng định danh đó thực sự là private.
- Nếu định danh cũng kết thúc với hai dấu gạch dưới, thì định danh này là một tên đặc biệt được định nghĩa bởi ngôn ngữ (ví dụ như `__init__` chẳng hạn).



CÚ PHÁP CƠ BẢN CỦA PYTHON

• Các từ khóa trong Python:

Đây là các từ dành riêng và không thể sử dụng chúng như là các hằng, biến hoặc cho bất kỳ tên định danh nào.

Tất cả từ khóa trong Python là chỉ ở dạng chữ thường.

and	exec	not
assert	finally	or
break	for	pass
class	from	print



CÚ PHÁP CƠ BẢN CỦA PYTHON

- **Các từ khóa trong Python:**

continue	global	raise
def	if	return
del	import	try
elif	in	while
else	is	with
except	lambda	yield



CÚ PHÁP CƠ BẢN CỦA PYTHON

• Dòng lệnh và độ thut dòng lệnh trong Python

- Python không cung cấp các dấu ngoặc ôm ({}) để chỉ các khối code cho định nghĩa lớp hoặc hàm hoặc điều khiển luồng.
- Các khối code được nhận biết bởi độ thut dòng code (indentation) trong Python và đây là điều bắt buộc.
- Số khoảng trắng trong độ thut dòng là biến đổi, nhưng tất cả các lệnh bên trong khối phải được thut cùng một số lượng khoảng trắng như nhau.



CÚ PHÁP CƠ BẢN CỦA PYTHON

- Trong Python thì tất cả các dòng liên tiếp nhau mà được thụt đầu dòng với cùng lượng khoảng trắng như nhau sẽ tạo nên một khối.

Ví dụ:

```
if True:  
    print "True"  
  
else:  
    print "False"
```

Khối sau sẽ tạo ra một lỗi:

```
if True:  
    print "Answer"  
    print "True"  
  
else:  
    print "Answer"  
    print "False"
```



CÚ PHÁP CƠ BẢN CỦA PYTHON

• Các lệnh trên nhiều dòng trong Python

Các lệnh trong Python có một nét đặc trưng là kết thúc với một newline (dòng mới). Tuy nhiên, Python cho phép sử dụng ký tự \ để chỉ rõ sự liên tục dòng.

Ví dụ:

```
total = item_one + \
        item_two + \
        item_three
```



CÚ PHÁP CƠ BẢN CỦA PYTHON

• Trích dẫn trong Python

Python chấp nhận trích dẫn đơn ('), kép ("") và trích dẫn tam (""" hoặc """") để biểu thị các hàng chuỗi, miễn là các trích dẫn này có cùng kiểu mở và đóng.

Trích dẫn tam được sử dụng để trải rộng chuỗi được trích dẫn qua nhiều dòng.

```
word = 'word'  
sentence = "This is a sentence."  
paragraph = """This is a paragraph. It is  
made up of multiple lines and sentences."""
```



CÚ PHÁP CƠ BẢN CỦA PYTHON

• Comment trong Python

Python hỗ trợ hai kiểu comment đó là comment đơn dòng và đa dòng.

Trong Python, một dấu #, mà không ở bên trong một hằng chuỗi nào, bắt đầu một comment đơn dòng. Tất cả ký tự ở sau dấu # và kéo dài cho đến hết dòng đó thì được coi là một comment và được bỏ qua bởi trình thông dịch.

```
# First comment  
print "Hello, Python!" # second comment
```



CÚ PHÁP CƠ BẢN CỦA PYTHON

• Comment trong Python

Python cũng hỗ trợ kiểu comment thứ hai, đó là kiểu comment đa dòng được cho bên trong các trích dẫn tam.

```
#single line comment  
print "Hello Python"  
"""This is  
multiline comment"""
```



CÚ PHÁP CƠ BẢN CỦA PYTHON

• Sử dụng dòng trống trong Python

- Một dòng mà chỉ chứa các khoảng trống trắng whitespace, có thể với một comment, thì được xem như là một dòng trống và Python hoàn toàn bỏ qua nó.
- Trong một phiên thông dịch trong chế độ tương tác, bạn phải nhập một dòng trống để kết thúc một lệnh đa dòng.



CÚ PHÁP CƠ BẢN CỦA PYTHON

- **Các lệnh đa dòng trên một dòng đơn trong Python**

- Dấu chấm phẩy (;) cho phép xuất hiện nhiều lệnh trên một dòng đơn.
 - Tất cả các lệnh được cung cấp này không bắt đầu một khối code mới.

```
import sys; x = 'foo'; sys.stdout.write(x + '\n')
```



CÚ PHÁP CƠ BẢN CỦA PYTHON

- **Các nhóm lệnh đa dòng trong Python**

- Một nhóm các lệnh đơn, mà tạo một khối code đơn, **được gọi là suite** trong Python.
 - Các lệnh phức hợp như if, while, def, và class cần một dòng header và một suite.
 - Các dòng header bắt đầu lệnh (với từ khóa) và kết thúc với một dấu hai chấm (:) và được sau bởi một hoặc nhiều dòng để tạo nên một suite.



CÚ PHÁP CƠ BẢN CỦA PYTHON

- Các nhóm lệnh đa dòng trong Python

Ví dụ:

```
if expression :
```

suite

```
elif expression :
```

suite

```
else :
```

suite



THẢO LUẬN NHÓM

NỘI DUNG:

Đặc điểm của ngôn ngữ lập trình Python:

1. Phân biệt về ngôn ngữ lập trình biên dịch và ngôn ngữ lập trình thông dịch.
2. So sánh cú pháp của ngôn ngữ lập trình Python với một số ngôn ngữ lập trình thông dụng khác như: C/C++; Java; Pascal.



BÀI TẬP

NỘI DUNG:

1. Cài đặt môi trường lập trình Python trên máy tính cá nhân
2. Viết chương trình hiện thị các nội dung:
“Hello, world!”;
“Họ và tên, Mã số sinh viên, ngành học”
Kết quả code báo cáo trên Github.



BÀI TẬP

CHUẨN BỊ CHO BUỔI HỌC TIẾP THEO:

1. Đọc các tài liệu về nội dung mục 2.4; và 2.5.
2. Tìm hiểu về khái niệm biến và toán tử, các loại toán tử trong Python.
3. Nghiên cứu về câu lệnh và các cấu trúc điều khiển trong ngôn ngữ lập trình Python.

Chương 2:

NGÔN NGỮ LẬP TRÌNH PYTHON



NỘI DUNG GIẢNG DẠY:

- 2.1. Một số đặc điểm của ngôn ngữ lập trình Python
- 2.2. Môi trường lập trình và thực thi Python
- 2.3. Cú pháp cơ bản của Python
- 2.4. Biến và các toán tử trong Python**
- 2.5. Các cấu trúc điều khiển



BIẾN VÀ CÁC TOÁN TỬ TRONG PYTHON

• Biến

- Biến là các vị trí bộ nhớ được dành riêng để lưu trữ dữ liệu. Một khi một biến đã được lưu trữ, nghĩa là một khoảng không gian đã được cấp phát trong bộ nhớ đó.

- Dựa trên kiểu dữ liệu của một biến, trình thông dịch cấp phát bộ nhớ và quyết định những gì có thể được lưu trữ trong khu nhớ dành riêng đó.

- Bằng việc gán các kiểu dữ liệu khác nhau cho các biến, chúng ta có thể lưu trữ số nguyên, thập phân hoặc ký tự trong các biến này.



BIẾN VÀ CÁC TOÁN TỬ TRONG PYTHON

- **Gán các giá trị cho biến trong Python**

- Trong Python, chúng ta không cần khai báo biến một cách tường minh.
- Khi gán bất cứ giá trị nào cho biến thì biến đó được khai báo một cách tự động. Phép gán được thực hiện bởi toán tử =.
- Toán hạng trái của toán tử = là tên biến và toán hạng phải là giá trị được lưu trữ trong biến.



BIẾN VÀ CÁC TOÁN TỬ TRONG PYTHON

- Gán các giá trị cho biến trong Python

Ví dụ:

```
a = 20          # Một phép gán số nguyên  
b = 100.0       # Một số thực  
ten = "Hoang"   # Một chuỗi
```



BIẾN VÀ CÁC TOÁN TỬ TRONG PYTHON

- **Phép đa gán (multiple assignment) trong Python**

- Python cho phép bạn gán một giá trị đơn cho một số biến một cách đồng thời. Python hỗ trợ hai kiểu đa gán sau:
 - Gán giá trị đơn cho nhiều biến.

```
a = b = c = 1
```

```
- Gán nhiều giá trị cho nhiều biến
```

```
a,b,c=5,10,15
```

Trong trường hợp này, các giá trị sẽ được gán theo thứ tự mà các biến xuất hiện.



BIẾN VÀ CÁC TOÁN TỬ TRONG PYTHON

- **Các kiểu dữ liệu chuẩn trong Python**

- Dữ liệu mà được lưu trữ trong bộ nhớ có thể có nhiều kiểu khác nhau.

Ví dụ: Lương của công nhân được lưu trữ dưới dạng một giá trị số còn địa chỉ của họ được lưu trữ dưới dạng các ký tự chữ - số.

Python có nhiều kiểu dữ liệu chuẩn được sử dụng để xác định các hành động có thể xảy ra trên chúng và phương thức lưu trữ cho mỗi kiểu.



BIẾN VÀ CÁC TOÁN TỬ TRONG PYTHON

- **Các kiểu dữ liệu chuẩn trong Python**

Python có 5 kiểu dữ liệu chuẩn là:

- | | | |
|---------------|-------------------|-------------|
| - Kiểu Number | - Kiểu String | - Kiểu List |
| - Kiểu Tuple | - Kiểu Dictionary | |

Ngoài kiểu Number và kiểu String mà có thể đã được làm quen với các ngôn ngữ lập trình khác thì ở trong Python còn xuất hiện thêm ba kiểu dữ liệu đó là List, Tuple và Dictionary.

Chúng ta sẽ tìm hiểu chi tiết từng kiểu dữ liệu trong một chương riêng.



BIẾN VÀ CÁC TOÁN TỬ TRONG PYTHON

• Chuyển đổi kiểu trong Python

- Khi cần thực hiện một số phép chuyển đổi kiểu để thỏa mãn hàm hoặc phương thức nào đó, ... Để thực hiện điều này, chúng ta sử dụng tên kiểu như là một hàm.
- Các hàm này trả về một đối tượng mới biểu diễn giá trị đã được chuyển đổi.

Hàm	Miêu tả
int(x [,base])	Chuyển đổi x thành một số nguyên. Tham số base xác định cơ sở nếu x là một chuỗi



BIẾN VÀ CÁC TOÁN TỬ TRONG PYTHON

Hàm	Miêu tả
long(x [,base])	Chuyển đổi x thành một long int. Tham số base xác định cơ sở nếu x là một chuỗi
float(x)	Chuyển đổi x thành một số thực
complex(real [,imag])	Chuyển đổi x thành một số phức
str(x)	Chuyển đổi x thành một chuỗi



BIẾN VÀ CÁC TOÁN TỬ TRONG PYTHON

- **Toán tử trong Python**

- Toán tử là các biểu tượng cụ thể mà thực hiện một số hoạt động trên một số giá trị và cho ra một kết quả.

Ví dụ:

Biểu thức $2 + 3 = 5$, thì 2 và 3 được gọi là các toán hạng và dấu $+$ được gọi là toán tử.



BIẾN VÀ CÁC TOÁN TỬ TRONG PYTHON

- **Các loại toán tử trong Python**

Python hỗ trợ các loại toán tử sau:

- Toán tử số học: // , + , - , * , / , % , **;
- Toán tử quan hệ (toán tử so sánh): < , > , <= , <= , == , != , <>;
- Toán tử gán: = , /= , += , -= , *= , %= , **= , //=;
- Toán tử logic: and , or , not;
- Toán tử membership: in , not in;
- Toán tử identify: is , is not;
- Toán tử thao tác bit: & , | , ^ , ~ , << , >>;



BIẾN VÀ CÁC TOÁN TỬ TRONG PYTHON

• Các loại toán tử trong Python

Python hỗ trợ các loại toán tử sau:

- Toán tử số học: // , + , - , * , / , % , **;
- Toán tử quan hệ (toán tử so sánh): < , > , <= , <= , == , != , <>;
- Toán tử gán: = , /= , += , -= , *= , %= , **= , //=;
- Toán tử logic: and , or , not;
- Toán tử membership: in , not in;
- Toán tử identify: is , is not;
- Toán tử thao tác bit: & , | , ^ , ~ , << , >>;



THẢO LUẬN NHÓM

NỘI DUNG:

Xác định thứ tự ưu tiên của các toán tử trong Python để mang lại kết quả như mong muốn trong quá trình làm việc.

Chương 2: NGÔN NGỮ LẬP TRÌNH PYTHON



NỘI DUNG GIẢNG DẠY:

- 2.1. Một số đặc điểm của ngôn ngữ lập trình Python
- 2.2. Môi trường lập trình và thực thi Python
- 2.3. Cú pháp cơ bản của Python
- 2.4. Biến và các toán tử trong Python
- 2.5. Các cấu trúc điều khiển**



CÁC CẤU TRÚC ĐIỀU KHIỂN

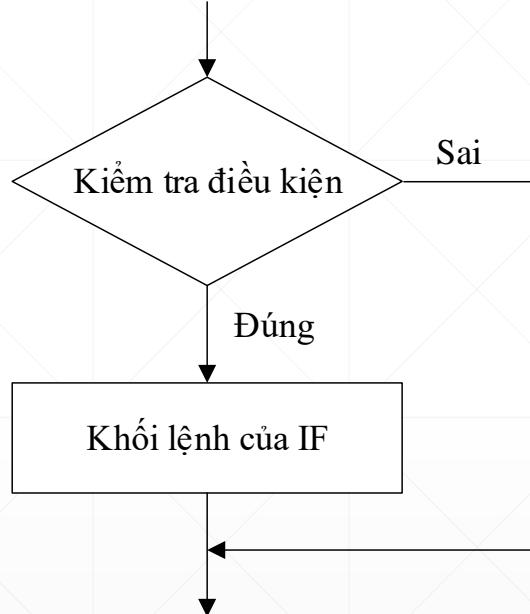
• Lệnh if

```
if điều kiện  
    khối lệnh
```

- Đánh giá điều kiện và sẽ thực hiện các lệnh khi điều kiện là True. Nếu điều kiện False thì lệnh sẽ không được thực hiện.
- Trong Python, khối lệnh của lệnh if được viết thụt lề vào trong. Khối lệnh của if bắt đầu với một khoảng thụt lề và dòng không thụt lề đầu tiên sẽ được hiểu là kết thúc lệnh if.

CÁC CẤU TRÚC ĐIỀU KHIỂN

- Cấu trúc lệnh if



num = 3

if num > 0:

print(num, "là số dương.")

num = -1

if num > 0:

print(num, "là số dương.")



CÁC CẤU TRÚC ĐIỀU KHIỂN

• Lệnh if...else

if điều kiện:

 Khối lệnh của if

else:

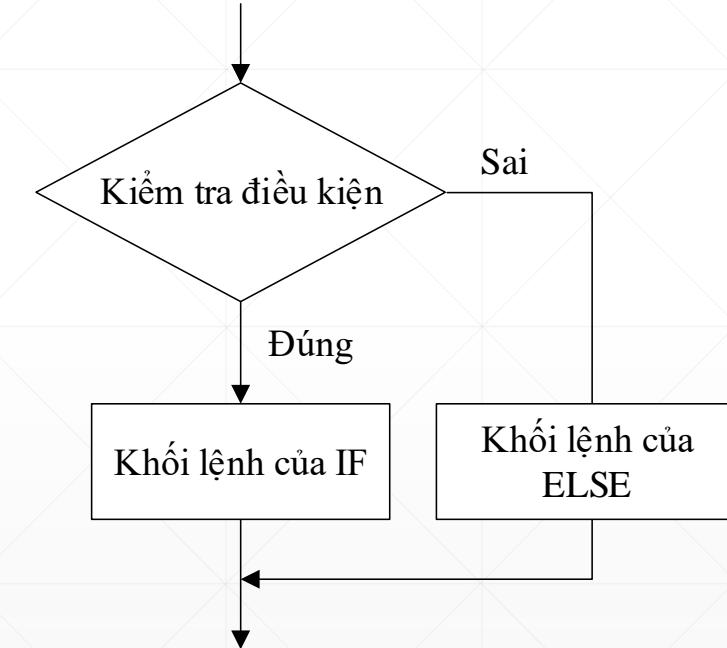
 Khối lệnh của else

- Lệnh if...else kiểm tra điều kiện và thực thi khối lệnh if nếu điều kiện đúng. Nếu điều kiện sai, khối lệnh của else sẽ được thực hiện.
- Thụt đầu dòng được sử dụng để tách các khối lệnh.

CÁC CẤU TRÚC ĐIỀU KHIỂN

- Cấu trúc if...else

```
if num >= 0:  
  
    print("So duong hoac bang 0")  
  
else:  
  
    print("So am")
```





CÁC CẤU TRÚC ĐIỀU KHIỂN

• Lệnh if...elif...else

if điều kiện:

 Khối lệnh của if

elif test expression:

 Khối lệnh của elif

else:

 Khối lệnh của else

Lệnh if ... elif ... elif ... là sự thay thế cho câu lệnh switch hay case trong các ngôn ngữ lập trình khác.



CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Lệnh if...elif...else**

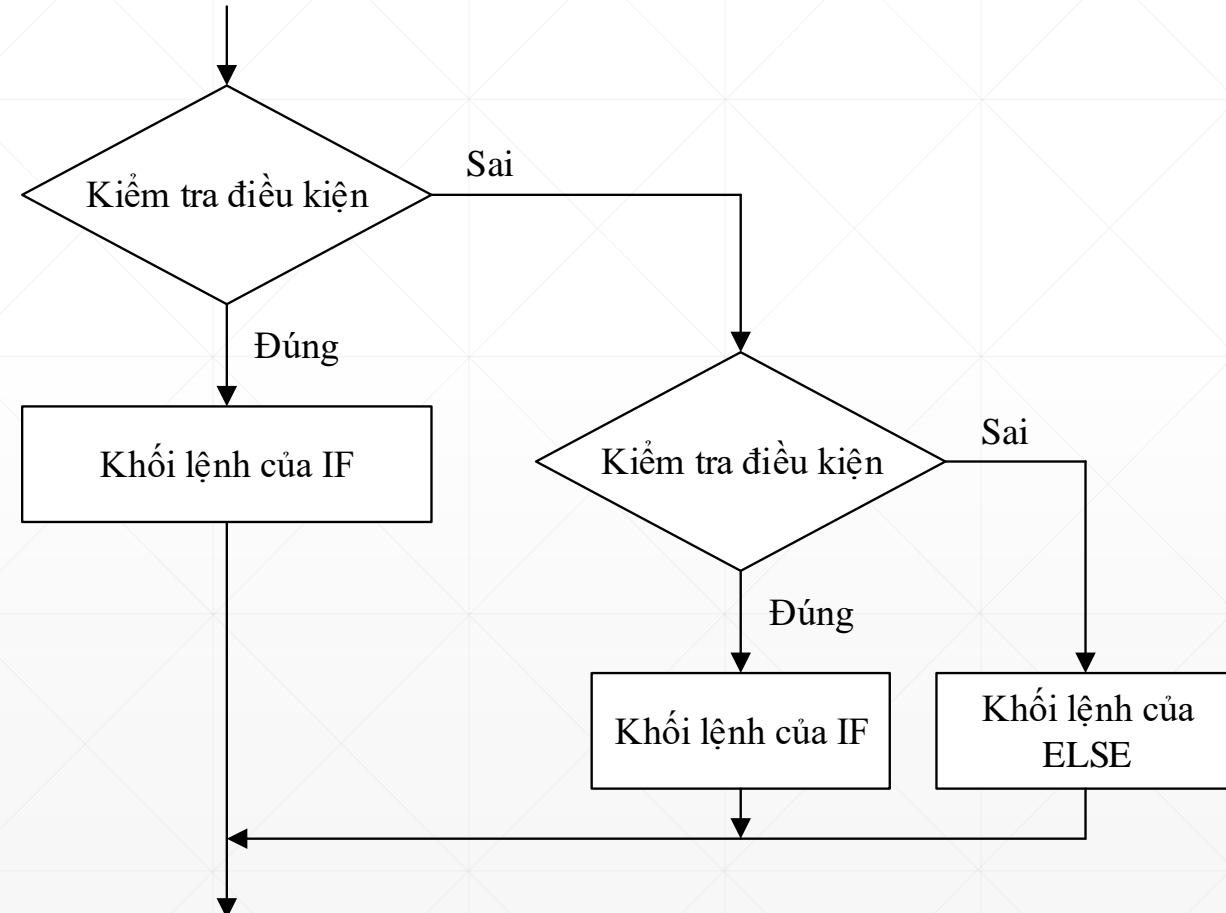
- elif là viết gọn của else if, nó cho phép chúng ta kiểm tra nhiều điều kiện.
- Nếu điều kiện là sai, nó sẽ kiểm tra điều kiện của khối elif tiếp theo và cứ như vậy cho đến hết.
- Nếu tất cả các điều kiện đều sai nó sẽ thực thi khối lệnh của else.
- Chỉ một khối lệnh trong if...elif...else được thực hiện theo điều kiện.
- Có thể không có hoặc có nhiều elif, phần else là tùy chọn.

CÁC CẤU TRÚC ĐIỀU KHIỂN

- Lệnh if...elif...else**

```
x = int(input("Nhap mot so: "))

if x < 0:
    print('So am')
elif x == 0:
    print('So 0')
elif x == 1:
    print('So 1')
else:
    print('So duong')
```





CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Lệnh if lồng nhau trong Python**

- Có thể viết lệnh if...elif...else trong một khối lệnh if...elif...else khác, và tạo thành lệnh if lồng nhau.
- Không giới hạn số lệnh được lồng vào lệnh khác.
- Thụt đầu dòng là cách duy nhất để nhận diện mức độ lồng, do đó nó có thể gây rối, nhầm lẫn, nên hạn chế sử dụng nếu có thể.



CÁC CẤU TRÚC ĐIỀU KHIỂN

- Lệnh if lồng nhau trong Python

```
num = float(input("Nhập một số: "))

if num >= 0:
    if num == 0:
        print("Số Không")
    else:
        print("Số dương")
else:
    print("Số âm")
```



CÁC CẤU TRÚC ĐIỀU KHIỂN

- Vòng lặp for trong Python

```
for bien_lap in chuoi_lap:
```

Khối lệnh của for

- chuoi_lap là chuỗi cần lặp, bien_lap là biến nhận giá trị của từng mục bên trong chuoi_lap trên mỗi lần lặp.

- Vòng lặp sẽ tiếp tục cho đến khi nó lặp tới mục cuối cùng trong chuỗi.

Khối lệnh của for được thuật lề để phân biệt với phần còn lại của code.



CÁC CẤU TRÚC ĐIỀU KHIỂN

- Vòng lặp for trong Python

```
for chu in 'kythuatlaptrinh':  
    print('Chữ cái hiện tại:', chu)  
  
#Lặp từ trong chuỗi  
chuoi = ['bố','mẹ','em']  
for tu in chuoi:  
    print('Anh yêu', tu)
```



CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Kết hợp for với else**

Khối lệnh của else sẽ được thực thi khi các mục trong chuỗi đã được lặp hết.

```
B = [0, 2, 4, 5]
for b in B:
    print(b)
else:
    print("Đã in hết số.")
```



CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Kết hợp for với else**

Lệnh break có thể được sử dụng để dừng vòng lặp for, lúc này phần else sẽ bị bỏ qua. Hay nói cách khác, phần else trong for sẽ chạy khi không có break nào được thực thi.

- **Hàm range()**

Có thể sử dụng hàm range() để tạo ra một dãy số. Ví dụ, range(100) sẽ tạo một dãy số từ 0 đến 99 (100 số).

Hàm range(số bắt đầu, số kết thúc, khoảng cách giữa hai số) được sử dụng để tạo dãy số tùy chỉnh. Nếu không đặt khoảng cách giữa hai số thì Python sẽ hiểu mặc định nó bằng 1.



BÀI TẬP

NỘI DUNG:

Viết chương trình tìm các số nguyên tố trong khoảng từ 0 đến 100 sử dụng vòng lặp for.



CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Vòng lặp while trong Python**

while điều_kiện_kiểm_tra:

 Khối lệnh của while

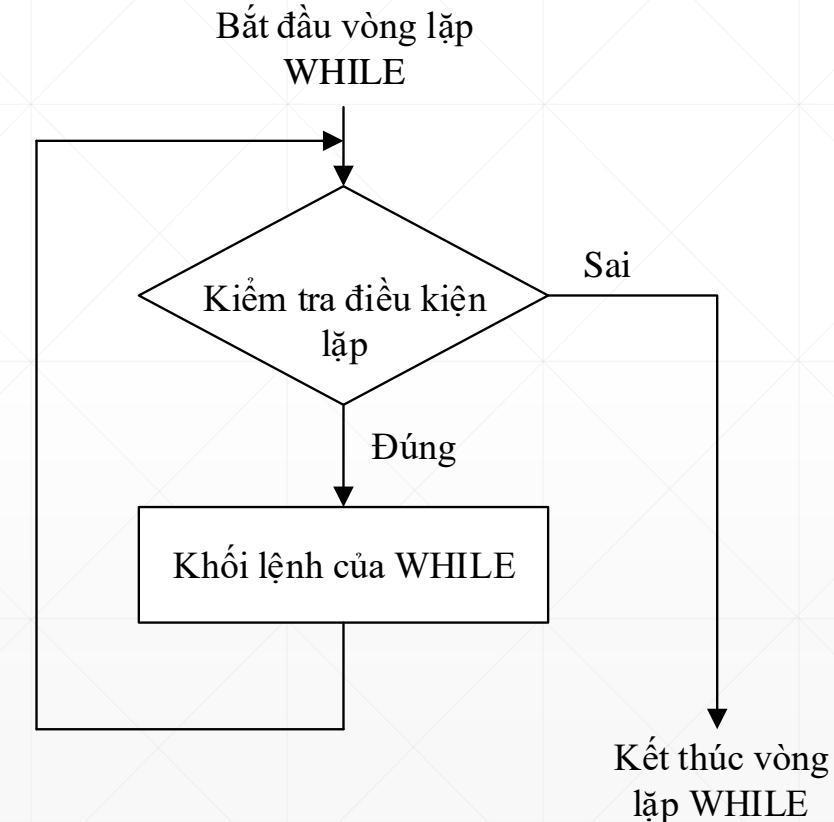
- Trong vòng lặp while, điều_kiện_kiểm_tra sẽ được kiểm tra đầu tiên.
- Khối lệnh của vòng lặp chỉ được nạp vào nếu điều_kiện_kiểm_tra là True.
- Sau một lần lặp, điều_kiện_kiểm_tra sẽ được kiểm tra lại.
- Quá trình này tiếp tục cho đến khi điều_kiện_kiểm_tra là False.

CÁC CẤU TRÚC ĐIỀU KHIỂN

• Vòng lặp while trong Python

Giống như if hay vòng lặp for, khôi lệnh của while cũng được xác định thông qua thuật lề.

Khôi lệnh bắt đầu với thuật lề đầu tiên và kết thúc với dòng không thuật lề đầu tiên liền sau khôi.





CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Vòng lặp while trong Python**

Trong Python mọi giá trị khác 0 đều là True, None và 0 được hiểu là False.

Đặc điểm này của while có thể dẫn đến trường hợp là while có thể không chạy vì ngay lần lặp đầu tiên điều_kiện_kiểm_tra đã False.

→ Khi đó, khôi lệnh của while sẽ bị bỏ qua và phần code ngay sau đó sẽ được thực thi.



CÁC CẤU TRÚC ĐIỀU KHIỂN

- Vòng lặp while trong Python

```
n = int(input("Nhập n: "))      # Nhập số n tùy ý  
tong = 0                         # Khai báo và gán giá trị cho tong  
i = 1                           # Khai báo và gán giá trị cho biến đếm i  
  
while i <= n:  
    tong = tong + i  
    i = i+1                      # Cập nhật biến đếm  
  
print("Tổng là", tong)
```



CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Kết hợp while với else**

Khối lệnh của else sẽ được thực hiện khi điều kiện của while là False.

```
dem = 0
while dem < 3:
    print("Đang ở trong vòng lặp while")
    dem = dem + 1
else:
    print("Đang ở trong else")
```



BÀI TẬP

NỘI DUNG:

Viết chương trình tìm các số nguyên tố trong khoảng từ 0 đến 100 sử dụng vòng lặp while.



CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Lệnh break trong Python**

Lệnh break kết thúc vòng lặp chứa nó và truyền điều khiển đến lệnh tiếp theo sau khối code của vòng lặp đó.

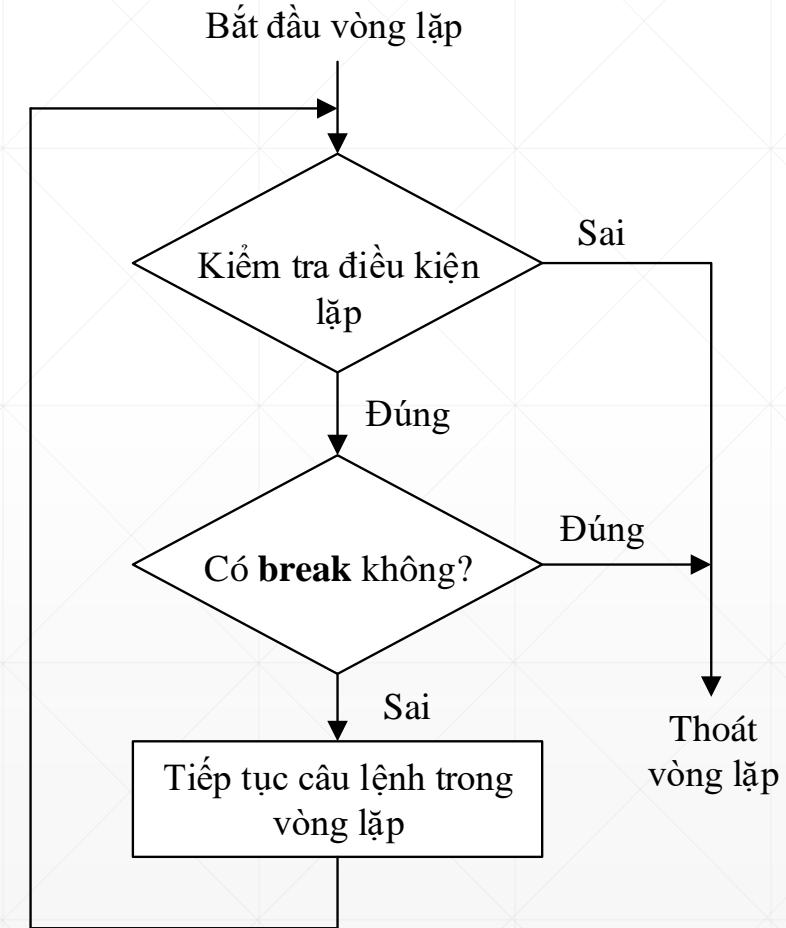
Nếu lệnh break ở trong một vòng lặp lồng nhau (vòng lặp bên trong một vòng lặp khác), break sẽ chấm dứt vòng lặp trong cùng.

```
break
```

CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Lệnh break trong vòng lặp for trong Python**

```
for var in sequence:  
    #khởi code bên trong vòng lặp for  
    if dieu_kien:  
        break  
    #code khác bên trong vòng lặp for  
    #code bên ngoài vòng lặp for
```

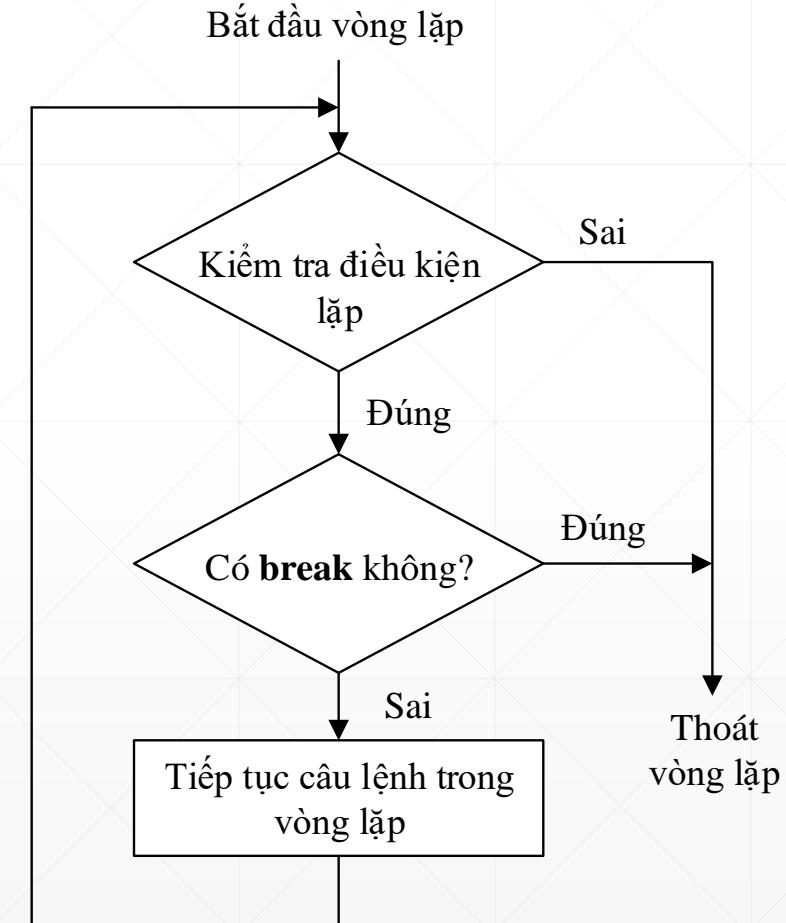


CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Lệnh break trong vòng lặp while**

Python

```
while dieu_kien_kiem_tra:  
    #code bên trong vòng lặp while  
    if dieu_kien:  
        break  
    #code khác bên trong vòng lặp while  
    #code bên ngoài vòng lặp while
```





CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Lệnh continue trong Python**

Lệnh continue được sử dụng để bỏ qua phần còn lại của code bên trong vòng lặp, áp dụng cho lần lặp hiện tại.

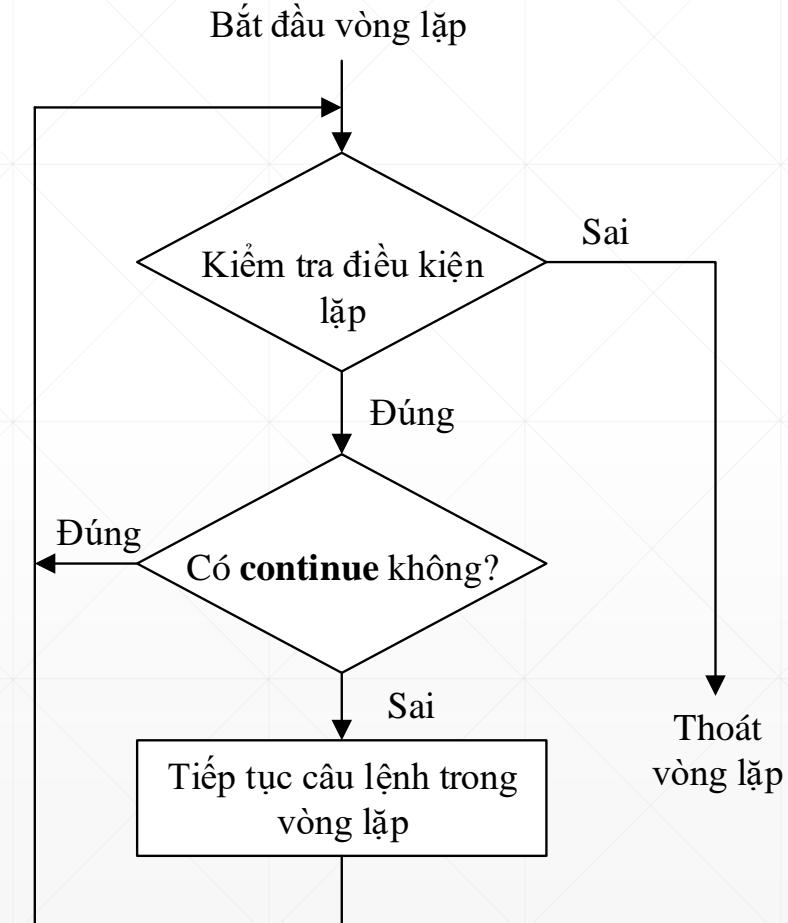
→ Nghĩa là vòng lặp không chấm dứt, nó sẽ tiếp tục với lần lặp kế tiếp.

```
continue
```

CÁC CẤU TRÚC ĐIỀU KHIỂN

- Lệnh continue trong vòng lặp for

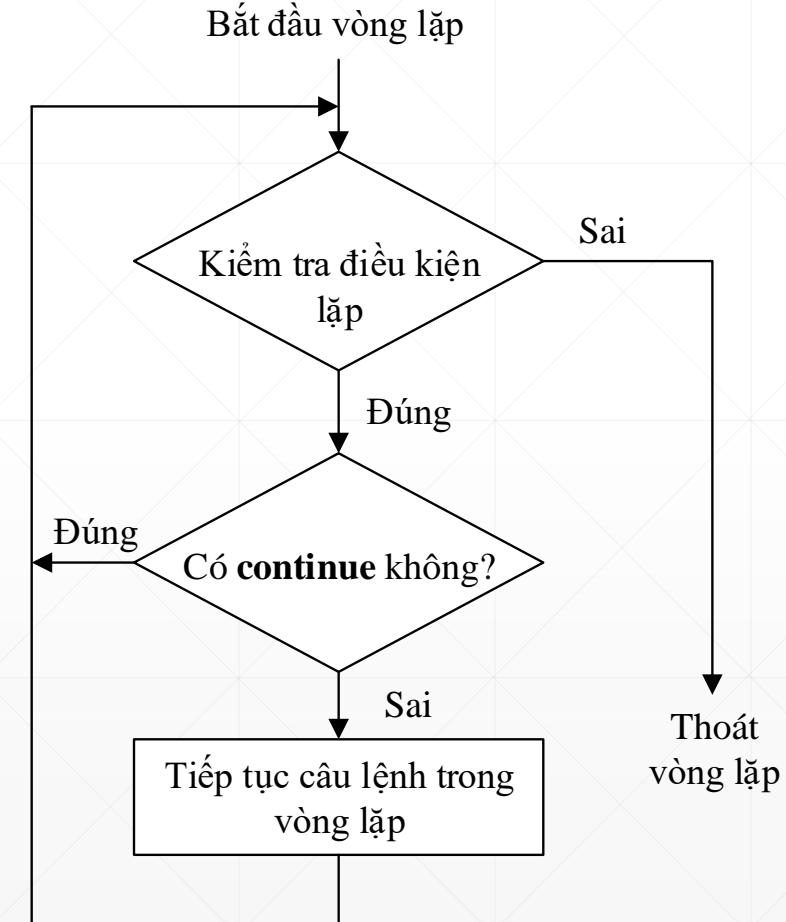
```
for var in sequence:  
    #khởi code bên trong vòng lặp for  
    if dieu_kien:  
        continue  
    #code khác bên trong vòng lặp for  
    #code bên ngoài vòng lặp for
```



CÁC CẤU TRÚC ĐIỀU KHIỂN

- Lệnh **continue** trong **vòng lặp while**

```
while dieu_kien_kiem_tra:  
    #code bên trong vòng lặp while  
    if dieu_kien:  
        continue  
    #code khác bên trong vòng lặp while  
    #code bên ngoài vòng lặp while
```





BÀI TẬP

NỘI DUNG:

1. Viết chương trình tìm tất cả các số chia hết cho 7 nhưng không phải bội số của 5, nằm trong đoạn 2000 và 3200 (tính cả 2000 và 3200). Các số thu được sẽ được in thành chuỗi trên một dòng, cách nhau bằng dấu phẩy.
2. Viết chương trình tìm dãy số Fibonacci trong khoảng từ 0 đến 1000.



BÀI TẬP

CHUẨN BỊ CHO BUỔI HỌC TIẾP THEO:

1. Đọc các tài liệu về nội dung mục 3.1; 3.2; và 3.3.
2. Tìm hiểu về khái niệm hàm trong lập trình Python.
3. Nghiên cứu về các phương thức truyền tham số trong lập trình hàm.



**TRƯỜNG ĐẠI HỌC VINH
VIỆN KỸ THUẬT VÀ CÔNG NGHỆ**

CHƯƠNG 3

LẬP TRÌNH HÀM TRONG PYTHON

Nghệ An, 2018

Chương 3:

LẬP TRÌNH HÀM TRONG PYTHON



NỘI DUNG GIẢNG DẠY:

- 3.1. Định nghĩa hàm trong Python
- 3.2. Các loại hàm trong Python
- 3.3. Tham số của hàm
- 3.4. Hàm vô danh
- 3.5. Các loại biến trong Python

Chương 3:

LẬP TRÌNH HÀM TRONG PYTHON



NỘI DUNG GIẢNG DẠY:

3.1. Định nghĩa hàm trong Python

3.2. Các loại hàm trong Python

3.3. Tham số của hàm

3.4. Hàm vô danh

3.5. Các loại biến trong Python



ĐỊNH NGHĨA HÀM TRONG PYTHON

- Trong Python, hàm là một nhóm các lệnh có liên quan đến nhau được dùng để thực hiện một tác vụ, nhiệm vụ cụ thể nào đó.
- Hàm giúp chia chương trình Python thành những khối/phần/môđun nhỏ hơn.
- Khi chương trình Python quá lớn, hoặc cần mở rộng, thì các hàm giúp chương trình có tổ chức và dễ quản lý hơn.
- Hàm còn có một tác dụng vô cùng quan trọng nữa là tránh việc phải lặp lại code để thực thi những tác vụ tương tự nhau, giúp code gọn hơn và có thể tái sử dụng.



ĐỊNH NGHĨA HÀM TRONG PYTHON

- Cú pháp của hàm Python

def *ten_ham*(các tham số/đối số):

"""Chuỗi văn bản để mô tả cho hàm (docstring)"""

Các câu lệnh

Về cơ bản, một định nghĩa hàm Python sẽ bao gồm các thành phần sau:

1. Từ khóa **def**: Đánh dấu sự bắt đầu của tiêu đề hàm.
2. *ten_ham*: Là định danh duy nhất dành cho hàm. Việc đặt tên hàm phải tuân thủ theo quy tắc viết tên và định danh trong Python.



ĐỊNH NGHĨA HÀM TRONG PYTHON

3. **Các tham số/đối số:** Chúng ta truyền giá trị cho hàm thông qua các tham số này. Chúng là tùy chọn.
4. **Dấu hai chấm (:) :** Đánh dấu sự kết thúc của tiêu đề hàm.
5. **docstring:** Chuỗi văn bản tùy chọn để mô tả chức năng của hàm.
6. **Các câu lệnh:** Một hoặc nhiều lệnh Python hợp lệ tạo thành khôi lệnh. Các lệnh này phải có cùng một mức thụt đầu dòng (thường là 4 khoảng trắng).
7. **Lệnh return:** Lệnh này là tùy chọn, dùng khi cần trả về giá trị từ hàm.

ĐỊNH NGHĨA HÀM TRONG PYTHON



- **Cách thức làm việc của hàm trong Python**

```
def ten_ham ():
```

• • • • • ←

• • •

• • • •

ten_ham ();

• • •

• • • •



ĐỊNH NGHĨA HÀM TRONG PYTHON

- **Ví dụ về hàm Python**

Dưới đây là một định nghĩa hàm đơn giản, gồm tên hàm, tham số của hàm, mô tả hàm và một câu lệnh:

```
def chao(ten):
    """Hàm này dùng để
    chào một người được truyền
    vào như một tham số"""
    print("Xin chào, " + ten + "!")
```



ĐỊNH NGHĨA HÀM TRONG PYTHON

- **Gọi hàm trong Python**

Khi một hàm đã được định nghĩa, bạn có thể gọi nó từ một hàm khác, chương trình khác hoặc thậm chí tại dấu nhắc lệnh.

Để gọi hàm chúng ta chỉ cần nhập tên hàm với những tham số thích hợp là được.

Ví dụ để gọi hàm chao() vừa định nghĩa bên trên, ta gõ lệnh sau ngay tại dấu nhắc:

```
>>> chao ("Sinh vien lop Ky thuatt lap trinh")
```



ĐỊNH NGHĨA HÀM TRONG PYTHON

• Docstring trong Python

Chuỗi đầu tiên ngay sau tiêu đề hàm được gọi là docstring (documentation string), nó được dùng để giải thích chức năng cho hàm.

Mặc dù docstring là không bắt buộc, nhưng việc giải thích ngắn gọn về chức năng của hàm sẽ giúp người dùng sau, thậm chí là bản thân người viết chương trình, khi gọi hàm có thể hiểu ngay hàm sẽ làm gì mà không cần phải tìm lại định nghĩa hàm để xem xét.



ĐỊNH NGHĨA HÀM TRONG PYTHON

- **Lệnh return trong hàm Python**

Lệnh return thường được dùng để thoát hàm và trả về nơi mà tại đó hàm được gọi.

Cú pháp của lệnh return:

```
return [danh_sach_bieu_thuc]
```

- Lệnh này có thể chứa biểu thức được tính toán và giá trị trả về.
- Nếu không có biểu thức nào trong câu lệnh hoặc không có lệnh return trong hàm thì hàm sẽ trả về None.



ĐỊNH NGHĨA HÀM TRONG PYTHON

- Ví dụ về lệnh return:

```
def gia_tri_tuyet_doi(so):
    """Hàm này trả về giá trị tuyệt đối
    của một số nhập vào"""
    if so >= 0:
        return so
    else:
        return -so
```



ĐỊNH NGHĨA HÀM TRONG PYTHON

- **Phạm vi và thời gian tồn tại của các biến**

- Phạm vi của biến là đoạn chương trình mà ở đó biến được thừa nhận. Các tham số và biến được xác định bên trong một hàm không thể "nhìn thấy" từ bên ngoài. Do đó, những biến và tham số này chỉ có phạm vi trong hàm.

- Thời gian tồn tại của biến là khoảng thời gian mà biến đó xuất hiện trong bộ nhớ. Khi hàm được thực thi thì biến sẽ tồn tại.

- Biến bị hủy khi chúng ta thoát khỏi hàm. Hàm không nhớ giá trị của biến trong những lần gọi hàm trước đó.



ĐỊNH NGHĨA HÀM TRONG PYTHON

- Ví dụ:

```
def ham_in():
    x = 15
    print("Giá trị bên trong hàm:",x)
    x = 30
ham_in()
print("Giá trị bên ngoài hàm:",x)
```

Chương 3: LẬP TRÌNH HÀM TRONG PYTHON



NỘI DUNG GIẢNG DẠY:

3.1. Định nghĩa hàm trong Python

3.2. Các loại hàm trong Python

3.3. Tham số của hàm

3.4. Hàm vô danh

3.5. Các loại biến trong Python



CÁC LOẠI HÀM TRONG PYTHON

Về cơ bản, Python có 2 loại hàm chính:

- Hàm được tích hợp sẵn trong Python: là các hàm có sẵn trong trình thông dịch của Python.
- Hàm do người dùng định nghĩa: định nghĩa để thực hiện một số công việc cụ thể.



CÁC LOẠI HÀM TRONG PYTHON

• Hàm được tích hợp sẵn trong Python

Trong phiên bản Python 3.6 có 68 hàm Python được tích hợp sẵn.

Hàm	Mô tả
abs()	Trả về giá trị tuyệt đối của một số
all()	Trả về True khi tất cả các phần tử trong iterable là đúng
any()	Kiểm tra bất kỳ phần tử nào của iterable là True
ascii()	Tả về string chứa đại diện (representation) có thể in
bin()	Chuyển đổi số nguyên sang chuỗi nhị phân
bool()	Chuyển một giá trị sang Boolean



CÁC LOẠI HÀM TRONG PYTHON

• Hàm được tích hợp sẵn trong Python

Hàm	Mô tả
bytearray()	Trả về mảng kích thước byte được cấp
bytes()	Trả về đối tượng byte không đổi
callable()	Kiểm tra xem đối tượng có thể gọi hay không
chr()	Trả về một ký tự (một chuỗi) từ Integer
classmethod()	Trả về một class method cho hàm
compile()	Trả về đối tượng code Python
complex()	Tạo một số phức



CÁC LOẠI HÀM TRONG PYTHON

- **Hàm được tích hợp sẵn trong Python**

Nếu muốn biết hàm này cụ thể làm gì, có đối số nào, chúng ta chỉ cần nhập lệnh:

```
print(ten_ham.__doc__)
```

→ Python sẽ giải thích khá đầy đủ về hàm.



CÁC LOẠI HÀM TRONG PYTHON

- **Hàm do người dùng định nghĩa:**

Việc định nghĩa hàm và gọi hàm đã được đề cập đến trong bài định nghĩa hàm Python (mục 3.1).

- Nếu ta sử dụng những hàm được người dùng khác viết dưới dạng thư viện, thì những hàm này gọi là hàm thư viện (library function). Như vậy, hàm ta tự định nghĩa có thể trở thành một hàm thư viện đối với người dùng nào đó.



CÁC LOẠI HÀM TRONG PYTHON

- Ưu điểm khi sử dụng hàm Python do người dùng định nghĩa**

- Hàm do người dùng định nghĩa giúp phân tích một chương trình lớn thành những phần nhỏ, khiến chương trình dễ hiểu, dễ duy trì và gỡ lỗi hơn.

- Khi một đoạn code bị lặp lại trong chương trình, thì có thể sử dụng hàm để gom đoạn code này lại và chạy khi cần bằng cách gọi hàm.

- Các lập trình viên cùng làm việc trong một dự án lớn, có thể phân chia công việc cho nhau bằng cách tạo các hàm khác nhau.

Chương 3: LẬP TRÌNH HÀM TRONG PYTHON



NỘI DUNG GIẢNG DẠY:

- 3.1. Định nghĩa hàm trong Python
- 3.2. Các loại hàm trong Python
- 3.3. Tham số của hàm**
- 3.4. Hàm vô danh
- 3.5. Các loại biến trong Python



THAM SỐ CỦA HÀM

Hàm có 0, 1 hoặc nhiều tham số. Ngăn cách nhau bởi dấu phẩy.

Tham số có 4 loại:

- Tham số bắt buộc
- Tham số có mặc định (Default parameter)
- Tham số có độ dài biến (Variable-Length Parameter)
- Tham số từ khóa (Keyword Parameter)



THAM SỐ CỦA HÀM

- **Tham số bắt buộc**

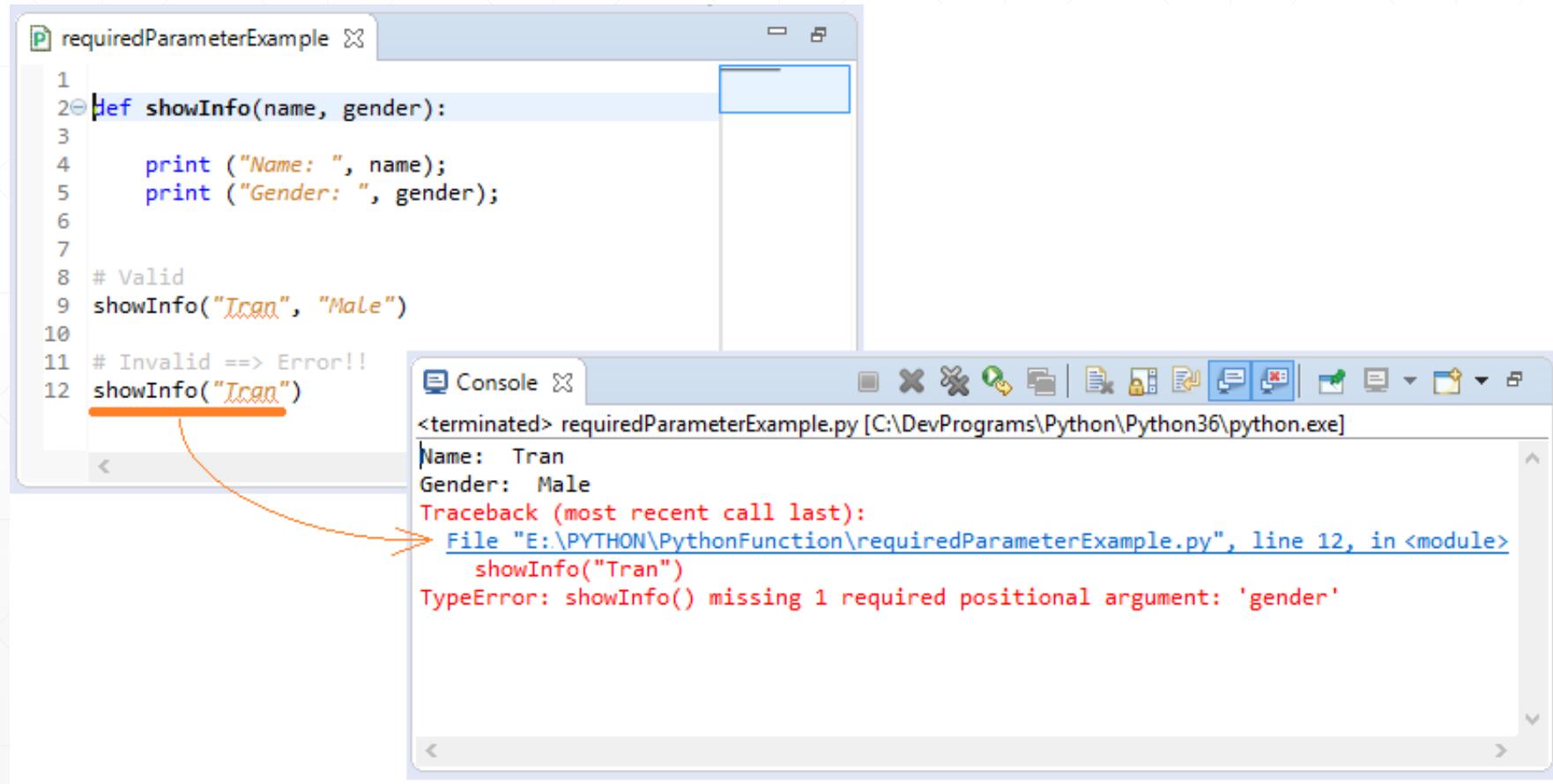
Ví dụ: Định nghĩa hàm showInfo, có 2 tham số, cả hai tham số này đều là bắt buộc.

- Khi gọi hàm này cần phải truyền 2 tham số vào cho hàm.
- Ngược lại chương trình sẽ xảy ra lỗi.

```
def showInfo(name, gender):  
    print ("Name: ", name);  
    print ("Gender: ", gender);
```

THAM SỐ CỦA HÀM

- Tham số bắt buộc



```
P requiredParameterExample ✘
1
2 def showInfo(name, gender):
3
4     print ("Name: ", name);
5     print ("Gender: ", gender);
6
7
8 # Valid
9 showInfo("Tran", "Male")
10
11 # Invalid ==> Error!!
12 showInfo("Tran")
```

```
Console ✘
<terminated> requiredParameterExample.py [C:\DevPrograms\Python\Python36\python.exe]
Name: Tran
Gender: Male
Traceback (most recent call last):
  File "E:\PYTHON\PythonFunction\requiredParameterExample.py", line 12, in <module>
    showInfo("Tran")
TypeError: showInfo() missing 1 required positional argument: 'gender'
```



THAM SỐ CỦA HÀM

- **Hàm với tham số mặc định**

- Hàm có thể có nhiều tham số, bao gồm các tham số bắt buộc và các tham số có giá trị mặc định.

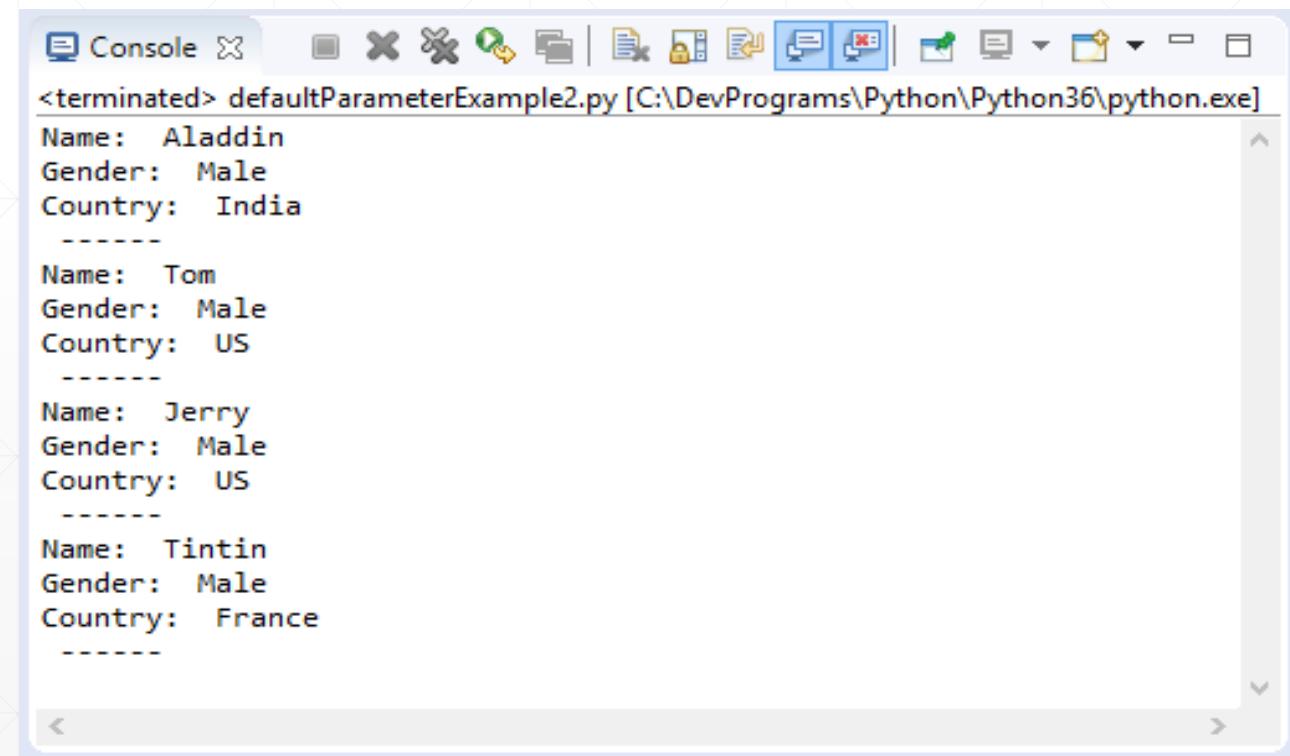
Ví dụ: Hàm showInfo có 3 tham số (name, gender = "Male", country = "US"):

- name là tham số bắt buộc.
- gender là tham số có giá trị mặc định "Male".
- country là tham số có giá trị mặc định "US".

THAM SỐ CỦA HÀM

- **Hàm với tham số mặc định**

```
def showInfo(name, gender = "Male", country ="US"):  
  
    print ("Name: ", name)  
  
    print ("Gender: ", gender)  
  
    print ("Country: ", country)
```



```
Console <terminated> defaultParameterExample2.py [C:\DevPrograms\Python\Python36\python.exe]  
Name: Aladdin  
Gender: Male  
Country: India  
-----  
Name: Tom  
Gender: Male  
Country: US  
-----  
Name: Jerry  
Gender: Male  
Country: US  
-----  
Name: Tintin  
Gender: Male  
Country: France  
-----
```



THAM SỐ CỦA HÀM

- **Hàm có tham số với độ dài thay đổi**

Tham số với độ dài thay đổi (Variable-length Parameter) là một tham số đặc biệt. Khi gọi hàm, bạn có thể truyền (pass) 0, 1 hoặc nhiều giá trị ứng với tham số đó.

Chú ý: "Variable-length Parameter" luôn phải là tham số cuối cùng của hàm.

Ví dụ: Hàm sumValues có 3 tham số:

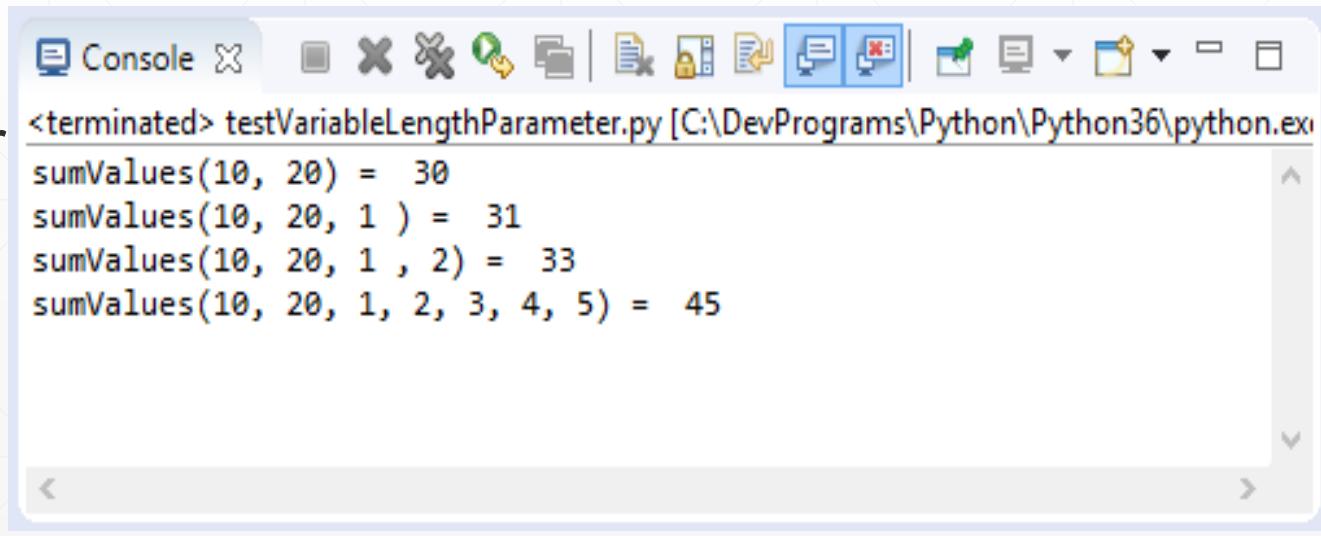
Tham số a, b là bắt buộc.

Tham số *others là "Variable-Length Parameter".

THAM SỐ CỦA HÀM

- **Hàm với tham số mặc định**

```
def sumValues(a, b, *others):  
  
    retValue = a + b  
  
    # Tham số 'others' giống như một mảng.  
  
    for other in others :  
  
        retValue = retValue + other  
  
    return retValue
```



```
Console <terminated> testVariableLengthParameter.py [C:\DevPrograms\Python\Python36\python.exe]  
sumValues(10, 20) = 30  
sumValues(10, 20, 1 ) = 31  
sumValues(10, 20, 1 , 2) = 33  
sumValues(10, 20, 1, 2, 3, 4, 5) = 45
```



THẢO LUẬN NHÓM

NỘI DUNG:

Các phương thức truyền tham số giữa các hàm trong ngôn ngữ lập trình Python.



BÀI TẬP

CHUẨN BỊ CHO BUỔI HỌC TIẾP THEO:

1. Đọc các tài liệu về nội dung mục 3.4; 3.5.
2. Tìm hiểu về các hàm trong lập trình Python.

Chương 3: LẬP TRÌNH HÀM TRONG PYTHON



NỘI DUNG GIẢNG DẠY:

- 3.1. Định nghĩa hàm trong Python
- 3.2. Các loại hàm trong Python
- 3.3. Tham số của hàm
- 3.4. Hàm vô danh**
- 3.5. Các loại biến trong Python



HÀM VÔ DANH

Các hàm được gọi là vô danh (**anonymous**) nếu chúng không được định nghĩa theo cách thông thường bởi từ khóa **def**, mà sử dụng từ khóa **lambda**.

- Hàm vô danh có thể có 0 hoặc nhiều tham số, nhưng trong thân hàm chỉ có duy nhất một biểu thức (expression). Giá trị của biểu thức chính là giá trị trả về của hàm. Nhưng không được sử dụng từ khóa 'return' ngay trước biểu thức.

- Danh sách các tham số cách nhau bởi dấu phẩy, và không được đặt trong cặp dấu ngoặc tròn ().



HÀM VÔ DANH

- Trong thân của hàm vô danh, chúng ta không thể truy cập các biến bên ngoài, chỉ có thể truy cập các tham số của nó.
- Hàm vô danh không thể gọi trực tiếp hàm print, bởi vì lambda đòi hỏi một biểu thức.

lambda tham_so: bieu_thuc

- Thường thì hàm Lambda được sử dụng khi cần một hàm vô danh trong thời gian ngắn, ví dụ như dùng làm đối số cho một hàm bậc cao hơn. Hàm Lambda thường được sử dụng cùng với các hàm Python tích hợp sẵn như filter() hay map(),...



HÀM VÔ DANH

Khai báo một biến: hello = một hàm nặc danh và không có tham số.

```
hello = lambda : "Hello"
```

Khai báo một biến: mySum = một hàm nặc danh có 2 tham số.

```
mySum = lambda a, b : a + b
```

```
a= hello()
```

```
print (a)
```

```
a = mySum(10, 20)
```

```
print (a)
```

A screenshot of a Python console window titled "Console". The window shows the output of a script named "lambdaFunctionExample.py" located at "C:\DevPrograms\Python\". The console displays two lines of output: "Hello" and "30", which are the results of executing the lambda functions defined in the code above.

```
<terminated> lambdaFunctionExample.py [C:\DevPrograms\Python]
Hello
30
```

Chương 3: LẬP TRÌNH HÀM TRONG PYTHON



NỘI DUNG GIẢNG DẠY:

- 3.1. Định nghĩa hàm trong Python
- 3.2. Các loại hàm trong Python
- 3.3. Tham số của hàm
- 3.4. Hàm vô danh
- 3.5. Các loại biến trong Python**



CÁC LOẠI BIẾN TRONG PYTHON

Trong Python tồn tại các loại biến: biến toàn cục (global), biến cục bộ (local), biến nonlocal.

- **Biến toàn cục (global)**

- Trong ngôn ngữ lập trình Python, một biến được khai báo bên ngoài hàm hoặc trong phạm vi toàn cục được gọi là biến toàn cục hay biến global.
- Biến toàn cục có thể được truy cập từ bên trong hoặc bên ngoài hàm.



CÁC LOẠI BIẾN TRONG PYTHON

- **Biến toàn cục (global)**

```
x = "Biến toàn cục"      # Khai báo biến x
```

```
# Gọi x từ trong hàm vidu()
```

```
def vidu():
```

```
    print("x trong hàm vidu() :", x)
```

```
vidu()
```

```
# Gọi x ngoài hàm vidu()
```

```
print("x ngoài hàm vidu():", x)
```



CÁC LOẠI BIẾN TRONG PYTHON

- **Biến toàn cục (global)**

Chuyện gì sẽ xảy ra nếu chúng ta thay đổi giá trị của biến toàn cục trong hàm?

Ví dụ:

```
x = 2
```

```
def vidu():
```

```
    x=x*2
```

```
    print(x)
```

```
vidu()
```



CÁC LOẠI BIẾN TRONG PYTHON

- **Biến toàn cục (global)**

Nếu chạy code này sẽ nhận được thông báo lỗi:

UnboundLocalError: local variable 'x' referenced before assignment

- Lỗi này xuất hiện là do Python xử lý x như một biến cục bộ và x không được định nghĩa trong vidu().
- Để thay đổi biến toàn cục trong một hàm bạn sẽ phải sử dụng từ khóa global.



CÁC LOẠI BIẾN TRONG PYTHON

- **Biến cục bộ (local)**

Biến được khai báo bên trong một hàm hoặc trong phạm vi cục bộ được gọi là biến cục bộ hay biến local.

Ví dụ:

```
def vidu():

    y = "Biến cục bộ"

vidu()

print(y)
```



CÁC LOẠI BIẾN TRONG PYTHON

- **Biến cục bộ (local)**

Khi chạy code trên chúng ta sẽ nhận được thông báo lỗi:

NameError: name 'y' is not defined

→ Lỗi này xuất hiện là do chúng ta đã cố truy cập vào biến cục bộ y trong phạm vi toàn cục, nhưng y chỉ làm việc trong hàm vidu() hoặc phạm vi cục bộ.



CÁC LOẠI BIẾN TRONG PYTHON

- **Biến cục bộ (local)**

Thông thường, để tạo một biến cục bộ, chúng ta sẽ khai báo nó trong một hàm như ví dụ dưới đây:

Ví dụ:

```
def vidu():

    y = "Biến cục bộ"

    print(y)

vidu()
```



CÁC LOẠI BIẾN TRONG PYTHON

• Biến nonlocal

Trong Python, biến nonlocal được sử dụng trong hàm lồng nhau nơi mà phạm vi cục bộ không được định nghĩa.

→ Nói dễ hiểu thì biến nonlocal không phải biến local, không phải biến global, bạn khai báo một biến là nonlocal khi muốn sử dụng nó ở phạm vi rộng hơn local, nhưng chưa đến mức global.

Để khai báo biến nonlocal ta cần dùng đến từ khóa nonlocal.



CÁC LOẠI BIẾN TRONG PYTHON

• Biến nonlocal

Trong code trên có một hàm lồng là ham_trong(), ta dùng từ khóa nonlocal để tạo biến nonlocal.

Hàm ham_trong() được định nghĩa trong phạm vi của ham Ngoai().

Lưu ý: Nếu chúng ta thay đổi giá trị của biến nonlocal, sự thay đổi sẽ xuất hiện trong biến cục bộ.

```
def ham Ngoai():
    x = "Biến cục bộ"
    def ham Trong():
        nonlocal x
        x = "Biến nonlocal"
        print("Bên trong:", x)
    ham Trong()
    print("Bên ngoài:", x)
ham Ngoai()
```



BÀI TẬP

NỘI DUNG:

1. Viết một hàm số tính giai thừa của một số cho trước. Kết quả được in thành chuỗi trên một dòng, phân tách bởi dấu phẩy.
2. Viết hàm số chuyển chuỗi ký tự được nhập vào từ bàn phím từ chữ thường sang chữ hoa.



BÀI TẬP

CHUẨN BỊ CHO BUỔI HỌC TIẾP THEO:

1. Đọc các tài liệu về nội dung mục 4.1; 4.2; và 4.3.
2. Tìm hiểu về khái niệm kiểu dữ liệu có cấu trúc trong lập trình Python.
3. Các phương thức và các hàm xây dựng sẵn để xử lý String, List trong Python.



**TRƯỜNG ĐẠI HỌC VINH
VIỆN KỸ THUẬT VÀ CÔNG NGHỆ**

CHƯƠNG 4

CÁC KIỂU DỮ LIỆU CÓ CẤU TRÚC

Nghệ An, 2018

Chương 4:

CÁC KIỂU DỮ LIỆU CÓ CẤU TRÚC



NỘI DUNG GIẢNG DẠY:

- 4.1. Kiểu strings
- 4.2. Kiểu lists
- 4.3. Kiểu Tuples
- 4.4. Kiểu Dictionary

Chương 4:

CÁC KIỂU DỮ LIỆU CÓ CẤU TRÚC



NỘI DUNG GIẢNG DẠY:

4.1. Kiểu Strings

4.2. Kiểu Lists

4.3. Kiểu Tuples

4.4. Kiểu Dictionary



KIỂU STRINGS

- String là một trong các kiểu phổ biến nhất trong Python. String trong Python là immutable.
- Chúng ta có thể tạo các chuỗi bằng cách bao một text trong một trích dẫn đơn hoặc trích dẫn kép.
- Python coi các lệnh trích dẫn đơn và kép là như nhau.

Ví dụ:

```
var1 = 'Hello World!'
```

```
var2 = "Python Programming"
```



KIỂU STRINGS

- **Truy cập các giá trị trong String**

Python không hỗ trợ một kiểu chữ cái; chúng được coi như các chuỗi có độ dài là 1. Trong Python, String được lưu giữ dưới dạng các ký tự đơn trong vị trí ô nhớ liên tiếp nhau.

Lợi thế của sử dụng String là nó có thể được truy cập từ cả hai hướng (tiến về trước forward hoặc ngược về sau backward).

Việc lập chỉ mục của cả hai hướng đều được cung cấp bởi sử dụng String trong Python:

- Chỉ mục với hướng forward bắt đầu với 0,1,2,3,...
- Chỉ mục với hướng backward bắt đầu với -1,-2,-3,...



KIỂU STRINGS

- **Truy cập các giá trị trong String**

Để truy cập các giá trị trong String, bạn sử dụng các dấu ngoặc vuông có chỉ mục ở bên trong.

```
var1 = 'Hello World'  
var2 = "Python Programming"  
print "var1[0]: ", var1[0]  
print "var2[1:5]: ", var2[1:5]
```



KIỂU STRINGS

- **Cập nhật String trong Python**

Có thể cập nhật một chuỗi đang tồn tại bằng cách gán (hoặc tái gán) một biến cho string khác.

Giá trị mới có thể liên quan hoặc khác hoàn toàn giá trị trước đó.

Ví dụ:

```
var1 = 'Hello World'  
  
print "Chuoi hien tai la :- ", var1[:6] + 'Python'
```

Khi code trên được thực thi sẽ cho kết quả:

Chuoi hien tai la :- Hello Python



KIỂU STRINGS

- **Các toán tử để thao tác với String trong Python**

Có ba kiểu toán tử được hỗ trợ bởi String, đó là:

- Toán tử cơ bản;
- Toán tử membership;
- Toán tử quan hệ.



KIỂU STRINGS

- **Các toán tử cơ bản để thao tác với String**

- Có hai loại toán tử cơ bản có thể được sử dụng với String, đó là toán tử nối chuỗi + và toán tử lặp chuỗi *.
- Cả hai toán hạng được truyền cho phép nối chuỗi này phải cùng kiểu, nếu không sẽ tạo một lỗi.

Ví dụ:

```
>>> "hoang" + "nam"
```

Khi code trên được thực thi sẽ cho kết quả:

'hoangnam'



KIỂU STRINGS

- **Các toán tử membership để thao tác với String**

- Toán tử in: trả về true nếu một ký tự là có mặt trong chuỗi đã cho, nếu không nó trả về false.
- Toán tử not in: trả về true nếu một ký tự là không tồn tại trong chuỗi đã cho, nếu không nó trả về false.

- **Các toán tử quan hệ để thao tác với String**

- Tất cả các toán tử quan hệ (như <,>, <=, >=, ==, !=, <>) cũng có thể áp dụng cho các String.
- Các chuỗi được so sánh dựa trên giá trị ASCII hoặc Unicode.



THẢO LUẬN NHÓM

NỘI DUNG:

1. Các toán tử định dạng chuỗi trong Python.
2. Các phương thức và hàm đã xây dựng sẵn để xử lý chuỗi trong Python.

Chương 4: CÁC KIỂU DỮ LIỆU CÓ CẤU TRÚC



NỘI DUNG GIẢNG DẠY:

4.1. Kiểu Strings

4.2. Kiểu Lists

4.3. Kiểu Tuples

4.4. Kiểu Dictionary



LIST TRONG PYTHON

- List trong Python là cấu trúc dữ liệu mà có khả năng lưu giữ các kiểu dữ liệu khác nhau.
- List trong Python là thay đổi (mutable), nghĩa là Python sẽ không tạo một List mới nếu bạn sửa đổi một phần tử trong List.
- List là một container mà giữ các đối tượng khác nhau trong một thứ tự đã cho. Các hoạt động khác nhau như chèn hoặc xóa có thể được thực hiện trên List.
- Một List có thể được tạo ra bởi lưu trữ một dãy các kiểu giá trị khác nhau được phân biệt bởi các dấu phẩy.



LIST TRONG PYTHON

- Cú pháp để tạo List:

```
<ten_list>=[giatri1, giatri2, ..., giatriN];
```

Một List trong Python được bao xung quanh bởi các dấu ngoặc vuông [].

Ví dụ:

```
list1 = ['vatly', 'hoahoc', 1997, 2000];
```

```
list2 = [1, 2, 3, 4, 5];
```

```
list3 = ["a", "b", "c", "d"];
```



LIST TRONG PYTHON

- Truy cập các giá trị trong List trong Python:

Tương tự như chỉ mục của chuỗi, chỉ mục của List bắt đầu từ 0.

Để truy cập các giá trị trong List, sử dụng cú pháp sau:

```
<ten_list>[index]
```

để lấy giá trị có sẵn tại chỉ mục đó.

Ví dụ:

```
list1 = ['vatly', 'hoahoc', 1997, 2000];  
print "list1[0]: ", list1[0]
```

Kết quả:

list1[0]: vatly



LIST TRONG PYTHON

- **Các hoạt động cơ bản trên List trong Python:**

- Có thể thực hiện các hoạt động nối với toán tử + hoặc hoạt động lặp với * như trong các chuỗi. Điểm khác biệt là ở đây nó tạo một List mới, không phải là một chuỗi.

Ví dụ:

```
list1=[10,20]
```

```
list2=[30,40]
```

```
list3=list1+list2
```

```
print list3
```

Kết quả:

```
>>>
```

```
[10, 20, 30, 40]
```

```
>>>
```



LIST TRONG PYTHON

- **Cập nhật List trong Python:**

- Có thể cập nhật một hoặc nhiều phần tử của List bởi gán giá trị cho chỉ mục cụ thể đó. Cú pháp:

```
<ten_list>[index]=<giatri>
```

```
list = ['vatly', 'hoahoc', 1997, 2000];
list[2] = 2001;
print "Gia tri moi tai chi muc thu 2: "
print list[2]
```

Kết quả:

Gia tri moi tai chi muc thu 2 :
2001



LIST TRONG PYTHON

• Xóa phần tử trong List:

- Để xóa một phần tử trong List, bạn có thể sử dụng lệnh del nếu bạn biết chính xác phần tử nào bạn muốn xóa hoặc sử dụng phương thức remove() nếu bạn không biết. Ví dụ:

```
list1 = ['vatly', 'hoahoc', 1997, 2000];
del list1[2];
print "Cac phan tu cua List sau khi xoa:"
print list1
```

Kết quả:

Cac phan tu cua List sau khi
xoa:
['vatly', 'hoahoc', 2000]



THẢO LUẬN NHÓM

NỘI DUNG:

Các hàm và phương thức đã xây dựng sẵn để xử lý List trong Python.



BÀI TẬP

NỘI DUNG:

1. Tạo một biến my_string và gán cho nó một chuỗi nội dung bất kỳ, sau đó print ra chiều dài chuỗi đó, cuối cùng là print ra chuỗi đó đã được chuyển thành viết hoa hoàn toàn.
2. Nhập m, n để tạo 2 List số nguyên ngẫu nhiên A[n] và B[m] có giá trị khoảng {-100,+100}. Ghép A và B thành List C[p=m+n]. In ra List A, B và C.
 - Sắp xếp List C tăng dần.
 - Nhập số nguyên x, kiểm tra x có xuất hiện trong ListC hay không, nếu có thì xuất hiện mấy lần, ở vị trí đầu tiên nào.
 - Xóa các phần tử trùng nhau của List C, in ra List C mới.



BÀI TẬP

CHUẨN BỊ CHO BUỔI HỌC TIẾP THEO:

1. Đọc các tài liệu về nội dung mục 4.3; 4.4.
2. Các hàm được xây dựng sẵn cho Tuple, Dictionary trong Python.

Chương 4: CÁC KIỂU DỮ LIỆU CÓ CẤU TRÚC



NỘI DUNG GIẢNG DẠY:

- 4.1. Kiểu Strings
- 4.2. Kiểu Lists
- 4.3. Kiểu Tuples**
- 4.4. Kiểu Dictionary



KIỂU TUPLES

- Một tuple là một dãy các đối tượng không thay đổi (immutable) trong Python, vì thế tuple không thể bị thay đổi.
- Các tuple cũng là các dãy giống như List.
- Không giống List mà sử dụng các dấu ngoặc vuông, thì tuple sử dụng các dấu ngoặc đơn.
- Các đối tượng trong tuple được phân biệt bởi dấu phẩy và được bao quanh bởi dấu ngoặc đơn ().
- Giống như chỉ mục của chuỗi, chỉ mục của tuple bắt đầu từ 0.



KIẾU TUPLES

Ví dụ:

```
fruitTuple = ("apple", "apricot", "banana", "coconut", "lemon")  
otherTuple = (100, "one", "two", 3)
```

```
print ("Fruit Tuple:")  
print (fruitTuple)  
print (" ----- ")  
print ("Other Tuple:")  
print (otherTuple)
```

A screenshot of a Python console window titled "Console". The window shows the command "<terminated> tuplesExample.py [C:\DevPrograms\Python\Python36\python.exe]" and the output of the program. The output consists of two printed tuples: "Fruit Tuple: ('apple', 'apricot', 'banana', 'coconut', 'lemon')" followed by a dashed line, and "Other Tuple: (100, 'one', 'two', 3)".

```
Console <terminated> tuplesExample.py [C:\DevPrograms\Python\Python36\python.exe]  
Fruit Tuple:  
('apple', 'apricot', 'banana', 'coconut', 'lemon')  
-----  
Other Tuple:  
(100, 'one', 'two', 3)
```



KIỂU TUPLES

• So sánh List và Tuple

List và Tuple đều là một dãy (sequence) các phần tử. Chúng có các khác biệt sau:

- Khi viết một List bạn sử dụng cặp dấu ngoặc vuông [], trong khi viết một Tuple bạn sử dụng dấu ngoặc tròn ().
- List là kiểu dữ liệu có thể biến đổi (mutable), bạn có thể sử dụng phương thức như append() để thêm phần tử vào List, hoặc sử dụng phương thức remove() để xóa các phần tử ra khỏi List mà không làm tạo ra thêm một thực thể 'List' khác trên bộ nhớ.



KIỂU TUPLES

- **Truy cập các phần tử của Tuples**

- Có thể truy cập vào các phần tử của Tuple thông qua chỉ số.
 - Các phần tử của Tuple được đánh chỉ chỉ từ trái sang phải, bắt đầu từ 0.
 - Có thể truy cập vào các phần tử của Tuple theo chỉ số âm (Negative index), các phần tử được đánh chỉ số từ phải sang trái với các giá trị -1, -2, ...



KIỂU TUPLES

- **Truy cập các phần tử của Tuples**

Truy cập các giá trị trong tuple tương tự như khi truy cập các phần tử trong List.

Ví dụ:

```
tup1 = ('vatly', 'hoahoc', 1997, 2000);  
tup2 = (1, 2, 3, 4, 5, 6, 7 );  
  
print "tup1[0]: ", tup1[0]  
print "tup2[1:5]: ", tup2[1:5]
```

Kết quả:

```
tup1[0]: vatly  
tup2[1:5]: [2, 3, 4, 5]
```



KIỂU TUPLES

- **Các hoạt động cơ bản trên tuple trong Python**

Giống như String và List, chúng ta cũng có thể sử dụng toán tử nối + và toán tử lặp * với tuple. Điểm khác biệt là nó tạo ra một tuple mới, không tạo ra một chuỗi hay list.

Ví dụ:

```
data1=(1,2,3,4)
```

```
data2=('x','y','z')
```

```
data3=data1+data2
```

```
print data3
```

Kết quả:

```
>>>
```

```
(1, 2, 3, 4, 'x', 'y', 'z')
```

```
>>>
```



KIẾU TUPLES

- **Xóa các phần tử của tuple trong Python**

Xóa các phần tử đơn của tuple là điều không thể. Chúng ta chỉ có thể xóa toàn bộ tuple với lệnh del.

Ví dụ: Chú ý rằng sẽ có một exception được tạo ra, đó là bởi vì sau khi xóa thì tuple này không tồn tại nữa.

```
data=(10,20,'hoang',40.6,'z')
```

```
print data
```

```
del data      # Se xoa du lieu cua tuple
```

```
print data    # Se hien thi mot error boi vi tuple da bi xoa
```



THẢO LUẬN NHÓM

NỘI DUNG:

Các hàm và phương thức đã xây dựng sẵn để xử lý Tuple trong Python.

Chương 4: CÁC KIỂU DỮ LIỆU CÓ CẤU TRÚC



NỘI DUNG GIẢNG DẠY:

- 4.1. Kiểu Strings
- 4.2. Kiểu Lists
- 4.3. Kiểu Tuples
- 4.4. Kiểu Dictionary**
- 4.5. Bài tập



KIỂU DICTIONARY

- Dictionary trong Python là một tập hợp các cặp key và value không có thứ tự.
- Là một container mà chứa dữ liệu, được bao quanh bởi các dấu ngoặc mỏc 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.



KIỂU DICTIONARY

- Key và value được phân biệt riêng rẽ bởi một dấu hai chấm (:).
- Các item phân biệt nhau bởi một dấu phẩy (,).
- Các item khác nhau được bao quanh bên trong một cặp dấu ngoặc mớc 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'}
```



KIẾU DICTIONARY

- **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



KIẾU DICTIONARY

- **Các thuộc tính của key trong Dictionary:**

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

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

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

Traceback (most recent call last):

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

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

TypeError: list objects are unhashable



KIẾU DICTIONARY

- **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:

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

```
print "Id cua nhan vien la: ",data['Id']
```

```
print "Ten cua nhan vien la:",data['Ten']
```



KIỂU DICTIONARY

- **Cập nhật Dictionary trong Python:**

- Item (cặp key-value) có thể được cập nhật 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 .

Ví dụ:

```
data={'Id':100, 'Ten':'Thanh', 'Nghenghiep':'Developer'}  
data['Nghenghiep']='Manager'  
data['Mucluong']=12000000  
print data1
```



KIẾU DICTIONARY

- **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, sử dụng cú pháp:

```
del ten_dictionary
```



THẢO LUẬN NHÓM

NỘI DUNG:

1. Các hàm và phương thức đã được xây dựng sẵn cho Dictionary trong Python.
2. Đặc điểm của các kiểu dữ liệu có cấu trúc.



BÀI TẬP

NỘI DUNG:

1. Viết chương trình chấp nhận một chuỗi số, phân tách bằng dấu phẩy từ giao diện điều khiển, tạo ra một danh sách và một tuple chứa mọi số.
2. Viết chương trình sắp xếp tuple (name, age, score) theo thứ tự tăng dần, name là string, age và height là number. Tuple được nhập vào bởi người dùng. Tiêu chí sắp xếp là:

Sắp xếp theo name sau đó sắp xếp theo age, sau đó sắp xếp theo score. Ưu tiên là tên > tuổi > điểm.



BÀI TẬP

NỘI DUNG:

3. Định nghĩa một hàm có thể in dictionary chứa các key là số từ 1 đến 20 (bao gồm cả 1 và 20) và các giá trị bình phương của chúng.

4. Với số nguyên n nhất định, hãy viết chương trình để tạo ra một dictionary chứa (i, i^2) như là số nguyên từ 1 đến n (bao gồm cả 1 và n) sau đó in ra dictionary này. Ví dụ: Giả sử số n là 8 thì đầu ra sẽ là: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64}.



BÀI TẬP

CHUẨN BỊ CHO BUỔI HỌC TIẾP THEO:

1. Đọc các tài liệu về nội dung mục 5.1; 5.2.
2. Tìm hiểu về đặc điểm của một số loại module trong lập trình Python.



**TRƯỜNG ĐẠI HỌC VINH
VIỆN KỸ THUẬT VÀ CÔNG NGHỆ**

CHƯƠNG 5

THIẾT KẾ MODULE

Nghệ An, 2018

Chương 5: THIẾT KẾ MODULE



NỘI DUNG GIẢNG DAY:

- 5.1. Các loại module/thư viện
- 5.2. Đường dẫn tìm để load module
- 5.3. Lấy danh sách thuộc tính và phương thức của một module
- 5.4. Cách khai báo và sử dụng module
- 5.5. Package module

Chương 5: THIẾT KẾ MODULE



NỘI DUNG GIẢNG DẠY:

5.1. Các loại module/thư viện

- 5.2. Đường dẫn tìm để load module
- 5.3. Lấy danh sách thuộc tính và phương thức của một module
- 5.4. Cách khai báo và sử dụng module
- 5.5. Package module



CÁC LOẠI MODULE/THƯ VIỆN

Module (mô-đun) đề cập đến một file chứa những câu lệnh Python và các định nghĩa.

Module thường được sử dụng khi muốn chia chương trình lớn thành những file nhỏ hơn để dễ quản lý và tổ chức.

Phổ biến nhất là những hàm Python hay phải sử dụng sẽ được định nghĩa trong một module và nhập vào Python thay vì sao chép định nghĩa trong những chương trình khác nhau. Nhờ thế, module cho phép tái sử dụng code.



CÁC LOẠI MODULE/THƯ VIỆN

Có 3 loại module thường thấy là:

- Viết bằng Python: có phần mở rộng là *.py
- Các thư viện liên kết động: có phần mở rộng là *.dll, *.pyd, *.so, *.sl, ...
- C-Module liên kết với trình phiên dịch.

Chương 5: THIẾT KẾ MODULE



NỘI DUNG GIẢNG DAY:

5.1. Các loại module/thư viện

5.2. Đường dẫn tìm để load module

5.3. Lấy danh sách thuộc tính và phương thức của một module

5.4. Cách khai báo và sử dụng module

5.5. Package module



ĐƯỜNG DẪN TÌM ĐỂ LOAD MODULE

Để tải một module vào script sử dụng cú pháp:

```
import modulename
```

Khi gặp câu lệnh trên thì trình biên dịch sẽ tiến hành tìm kiếm file module tương ứng theo thứ tự thư mục sau:

1. Thư mục hiện hành mà script đang gọi;
2. Các thư mục trong PYTHONPATH (nếu có set);
3. Các thư mục cài đặt chuẩn trên Linux, Unix...



ĐƯỜNG DẪN TÌM ĐỂ LOAD MODULE

Có thể biết được đường dẫn mà một module đã được load bằng đoạn code dưới đây:

```
import math  
math.__file__
```

(Ví dụ trả về '/usr/lib/python2.5/lib-dynload/math.so')

```
import random  
random.__file__
```

(Ví dụ trả về '/usr/lib/python2.5/random.pyc')



THẢO LUẬN NHÓM

NỘI DUNG:

1. Tìm hiểu một số module quan trọng trong Python.
2. Tìm đường dẫn mà một số module.



BÀI TẬP

CHUẨN BỊ CHO BUỔI HỌC TIẾP THEO:

1. Đọc các tài liệu về nội dung mục 5.3; 5.4.
2. Tìm hiểu về đặc điểm của một số loại module trong lập trình Python.

Chương 5: THIẾT KẾ MODULE



NỘI DUNG GIẢNG DAY:

5.1. Các loại module/thư viện

5.2. Đường dẫn tìm để load module

5.3. Lấy danh sách thuộc tính và phương thức của một module

5.4. Cách khai báo và sử dụng module

5.5. Package module

LẤY DANH SÁCH THUỘC TÍNH VÀ PHƯƠNG THỨC CỦA MỘT MODULE



Để lấy được danh sách các thuộc tính và phương thức mà module hỗ trợ, sử dụng hàm **dir(modulename)**.

Ví dụ:

```
['__doc__', '__file__', '__name__', '__package__','acos', 'acosh', 'asin',
'asinh', 'atan', 'atan2','atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees',
'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum',
'gamma', 'hypot', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p',
'modf', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'trunc']
```

LẤY DANH SÁCH THUỘC TÍNH VÀ PHƯƠNG THỨC CỦA MỘT MODULE



Có thể gọi hàm `dir()` không truyền tham số để lấy các thuộc tính và phương thức của scope hiện tại đang thực thi.

Chương 5: THIẾT KẾ MODULE



NỘI DUNG GIẢNG DAY:

- 5.1. Các loại module/thư viện
- 5.2. Đường dẫn tìm để load module
- 5.3. Lấy danh sách thuộc tính và phương thức của một module
- 5.4. Cách khai báo và sử dụng module**
- 5.5. Package module



CÁCH KHAI BÁO VÀ SỬ DỤNG MODULE

Giả sử chúng ta tạo một file python *mymath.py* có nội dung như sau:

```
def cong(a, b):
```

```
    return a + b
```

```
def tru(a, b):
```

```
    return a - b
```

```
def nhan(a, b)
```

```
    return a * b
```



CÁCH KHAI BÁO VÀ SỬ DỤNG MODULE

Sau đó tạo một file có tên myexample.py, trong cùng thư mục với file mymath.py vừa tạo ở trên, có nội dung như sau:

```
import mymath  
  
num1 = 1  
  
num2 = 2  
  
print 'Tong hai so la: ',mymath.cong(num1, num2)
```

Vào command line, thực hiện gọi file myexample, sau khi thực hiện sẽ hiển thị lên màn hình là:

```
Tong hai so la: 3
```

Chương 5: THIẾT KẾ MODULE



NỘI DUNG GIẢNG DAY:

- 5.1. Các loại module/thư viện
- 5.2. Đường dẫn tìm để load module
- 5.3. Lấy danh sách thuộc tính và phương thức của một module
- 5.4. Cách khai báo và sử dụng module
- 5.5. Package module**



PACKAGE MODULE

Có thể gom nhiều module .py vào một thư mục và tên thư mục là tên của package và tạo một file `__init__.py` trong thư mục này.

Như vậy, cấu trúc thư của một package sẽ như sau:

```
| -- mypack
|   | -- __init__.py
|   | -- mymodule1.py
|   | -- mymodule2.py
|
|
```



PACKAGE MODULE

Có thể sử dụng mymodule1 theo cú pháp import sau:

```
import mypack.mymodule1
```

hoặc

```
import mypack.mymodule1 as mymodule1
```

hoặc

```
import mypack.mymodule1 as mod
```



PACKAGE MODULE

Khi sử dụng một module thuộc một package thì các lệnh trong file `__init__.py` sẽ được thực hiện trước.

Thông thường thì file `__init__.py` sẽ rỗng.

Có thể tạo các subpackage bên trong một package theo đúng cấu trúc thư mục, có file `__init__.py`.

Ví dụ:

```
import mypack.mysubpack.mysubpack.module
```



BÀI TẬP

NỘI DUNG:

1. Tạo một số thập phân ngẫu nhiên, có giá trị nằm trong khoảng từ 10 đến 100 bằng cách sử dụng module math của Python.
2. viết chương trình để xuất một số ngẫu nhiên, chia hết cho 5 và 7, từ 0 đến 200 (gồm cả 0 và 200), sử dụng module random và list comprehension..



BÀI TẬP

CHUẨN BỊ CHO BUỔI HỌC TIẾP THEO:

1. Đọc các tài liệu về nội dung mục 6.1; 6.2; 6.3 và 6.4.
2. Tìm hiểu và so sánh lập trình cấu trúc với hướng đối tượng.



TRƯỜNG ĐẠI HỌC VINH
VIỆN KỸ THUẬT VÀ CÔNG NGHỆ

CHƯƠNG 6

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Nghệ An, 2018

Chương 6: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG



NỘI DUNG GIẢNG DẠY:

- 6.1. Lập trình hướng đối tượng
- 6.2. Tạo class trong Python
- 6.3. Tham số có mặc định trong Constructor
- 6.4. So sánh các đối tượng
- 6.5. Thuộc tính
- 6.6. Các hàm truy cập vào thuộc tính
- 6.7. Biến của lớp
- 6.8. Kế thừa và đa kế thừa

Chương 6: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG



NỘI DUNG GIẢNG DẠY:

6.1. Lập trình hướng đối tượng

- 6.2. Tạo class trong Python
- 6.3. Tham số có mặc định trong Constructor
- 6.4. So sánh các đối tượng
- 6.5. Thuộc tính
- 6.6. Các hàm truy cập vào thuộc tính
- 6.7. Biến của lớp
- 6.8. Kế thừa và đa kế thừa



LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

Lập trình hướng đối tượng (Object-oriented programming - OOP) là một kỹ thuật hỗ trợ, cho phép lập trình viên trực tiếp làm việc với các đối tượng mà họ định nghĩa lên.

Kĩ thuật này giúp tăng năng suất, đơn giản hóa độ phức tạp khi bảo trì cũng như mở rộng phần mềm.

Có khá nhiều ngôn ngữ lập trình theo hướng đối tượng như C++, Java, PHP,... và còn cả Python.

Khái niệm về OOP trong Python tập trung vào việc tạo code sử dụng lại. Khái niệm này còn được gọi là DRY (Don't Repeat Yourself).



LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

- **Các nguyên lý cơ bản:**

- Tính kế thừa (Inheritance): cho phép một lớp (class) có thể kế thừa các thuộc tính và phương thức từ các lớp khác đã được định nghĩa.

- Tính đóng gói (Encapsulation): là quy tắc yêu cầu trạng thái bên trong của một đối tượng được bảo vệ và tránh truy cập được từ code bên ngoài.

- Tính đa hình (Polymorphism): là khái niệm mà hai hoặc nhiều lớp có những phương thức giống nhau nhưng có thể thực thi theo những cách thức khác nhau.



LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

• Lớp (Class) và Đối tượng (Object):

- Đối tượng (Object) là những thực thể tồn tại có hành vi. Ví dụ đối tượng là một xe ô tô có tên hãng, màu sắc, loại nguyên liệu, hành vi đi, dừng, đỗ, nổ máy...

- Lớp (Class) là một kiểu dữ liệu đặc biệt do người dùng định nghĩa, tập hợp nhiều thuộc tính đặc trưng cho mọi đối tượng được tạo ra từ lớp đó.

Thuộc tính là các giá trị của lớp. Sau này khi các đối tượng được tạo ra từ lớp, thì thuộc tính của lớp lúc này sẽ trở thành các đặc điểm của đối tượng đó.



LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

- **Phân biệt giữa Đối tượng (Object) và Lớp (Class):**

- Đối tượng (Object): có trạng thái và hành vi.
- Lớp (Class): có thể được định nghĩa như là một template mô tả trạng thái và hành vi mà loại đối tượng của lớp hỗ trợ.

Một đối tượng là một thực thể (instance) của một lớp.

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

- Phân biệt giữa Đối tượng (Object) và Lớp (Class):





LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

- **Phương thức:**

Phương thức (Method) là các hàm được định nghĩa bên trong phần thân của một lớp. Chúng được sử dụng để xác định các hành vi của một đối tượng.

- **Kế thừa (Inheritance):**

Tính kế thừa cho phép một lớp (class) có thể kế thừa các thuộc tính và phương thức từ các lớp khác đã được định nghĩa. Lớp đã có gọi là lớp cha, lớp mới phát sinh gọi là lớp con. Lớp con kế thừa tất cả thành phần của lớp cha, có thể mở rộng các thành phần kế thừa và bổ sung thêm các thành phần mới.



LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

- **Đóng gói (Encapsulation):**

Sử dụng OOP trong Python, chúng ta có thể hạn chế quyền truy cập vào trạng thái bên trong của đối tượng. Điều này ngăn chặn dữ liệu bị sửa đổi trực tiếp, được gọi là đóng gói.

Trong Python, chúng ta biểu thị thuộc tính private này bằng cách sử dụng dấu gạch dưới làm tiền tố: "_" hoặc "__".



LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

- **Đa hình (Polymorphism):**

Tính đa hình là khái niệm mà hai hoặc nhiều lớp có những phương thức giống nhau nhưng có thể thực thi theo những cách thức khác nhau.

Giả sử, chúng ta cần tô màu một hình khối, có rất nhiều lựa chọn cho hình của bạn như hình chữ nhật, hình vuông, hình tròn. Tuy nhiên, bạn có thể sử dụng cùng một phương pháp để tô màu bất kỳ hình dạng nào.



THẢO LUẬN NHÓM

NỘI DUNG:

1. So sánh lập trình cấu trúc với hướng đối tượng.
2. Liệt kê các ngôn ngữ lập trình cấu trúc và các ngôn ngữ lập trình hướng đối tượng.

Chương 6: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG



NỘI DUNG GIẢNG DẠY:

- 6.1. Lập trình hướng đối tượng
- 6.2. Tạo class trong Python**
- 6.3. Tham số có mặc định trong Constructor
- 6.4. So sánh các đối tượng
- 6.5. Thuộc tính
- 6.6. Các hàm truy cập vào thuộc tính
- 6.7. Biến của lớp
- 6.8. Kế thừa và đa kế thừa



TẠO CLASS TRONG PYTHON

Đối tượng (Object) chỉ đơn giản là một tập hợp các dữ liệu (các biến) và các phương thức (các hàm) hoạt động trên các dữ liệu đó.
Và, lớp (class) là một kế hoạch chi tiết cho đối tượng.

Ví dụ:

Class như một bản phác thảo của một ngôi nhà chứa tất cả các chi tiết về sàn nhà, cửa ra vào, cửa sổ,...

Dựa trên những mô tả này, chúng ta sẽ xây dựng những ngôi nhà.
Vậy nhà ở đây chính là đối tượng.

TẠO CLASS TRONG PYTHON



Vì nhiều ngôi nhà có thể được làm từ một mô tả nên chúng ta có thể tạo ra nhiều vật thể từ một lớp.

Một đối tượng cũng được gọi là một thể hiện (instance) của một lớp và quá trình tạo đối tượng này được gọi là instantiation.



TẠO CLASS TRONG PYTHON

- **Khai báo Class:**

```
class ClassName:
```

'Mô tả ngắn về class (Không bắt buộc)'

```
# Code ...
```

Để định nghĩa một lớp chúng ta sử dụng từ khóa class, tiếp sau đó là tên của lớp và dấu hai chấm (:).

Dòng đầu tiên trong thân của lớp là chuỗi (string) mô tả ngắn gọn về lớp này (Không bắt buộc), bạn có thể truy cập vào chuỗi này thông qua ClassName.__doc__ .



TẠO CLASS TRONG PYTHON

Trong thân của lớp bạn có thể khai báo các thuộc tính, phương thức (method) và các phương thức khởi tạo (Constructor).

- **Thuộc tính (Attribute):**

Thuộc tính là một thành viên thành viên của lớp.

Ví dụ: Hình chữ nhật có hai thuộc tính width và height (Chiều rộng và chiều cao).





TẠO CLASS TRONG PYTHON

- **Phương thức khởi tạo (Constructor):**

- Phương thức khởi tạo (constructor) là một phương thức đặc biệt của lớp (class), nó luôn có tên là `__init__`
- Tham số đầu tiên của constructor luôn là `self` (một từ khóa ám chỉ chính class đó).
- Constructor được sử dụng để tạo ra một đối tượng.
- Constructor gán các giá trị từ tham số vào các thuộc tính của đối tượng sẽ được tạo ra.



TẠO CLASS TRONG PYTHON

- **Phương thức khởi tạo (Constructor):**

- Chỉ có thể định nghĩa nhiều nhất một phương thức khởi tạo (constructor) trong class.
- Nếu class không được định nghĩa constructor, Python mặc định coi rằng nó thừa kế từ constructor của lớp cha.



TẠO CLASS TRONG PYTHON

- Ví dụ:

```
class Rectangle :  
    'This is Rectangle class'  
    # Một phương thức được sử dụng để tạo đối tượng.  
    def __init__(self, width, height):  
        self.width = width  
        self.height = height  
    def getWidth(self):  
        return self.width
```



TẠO CLASS TRONG PYTHON

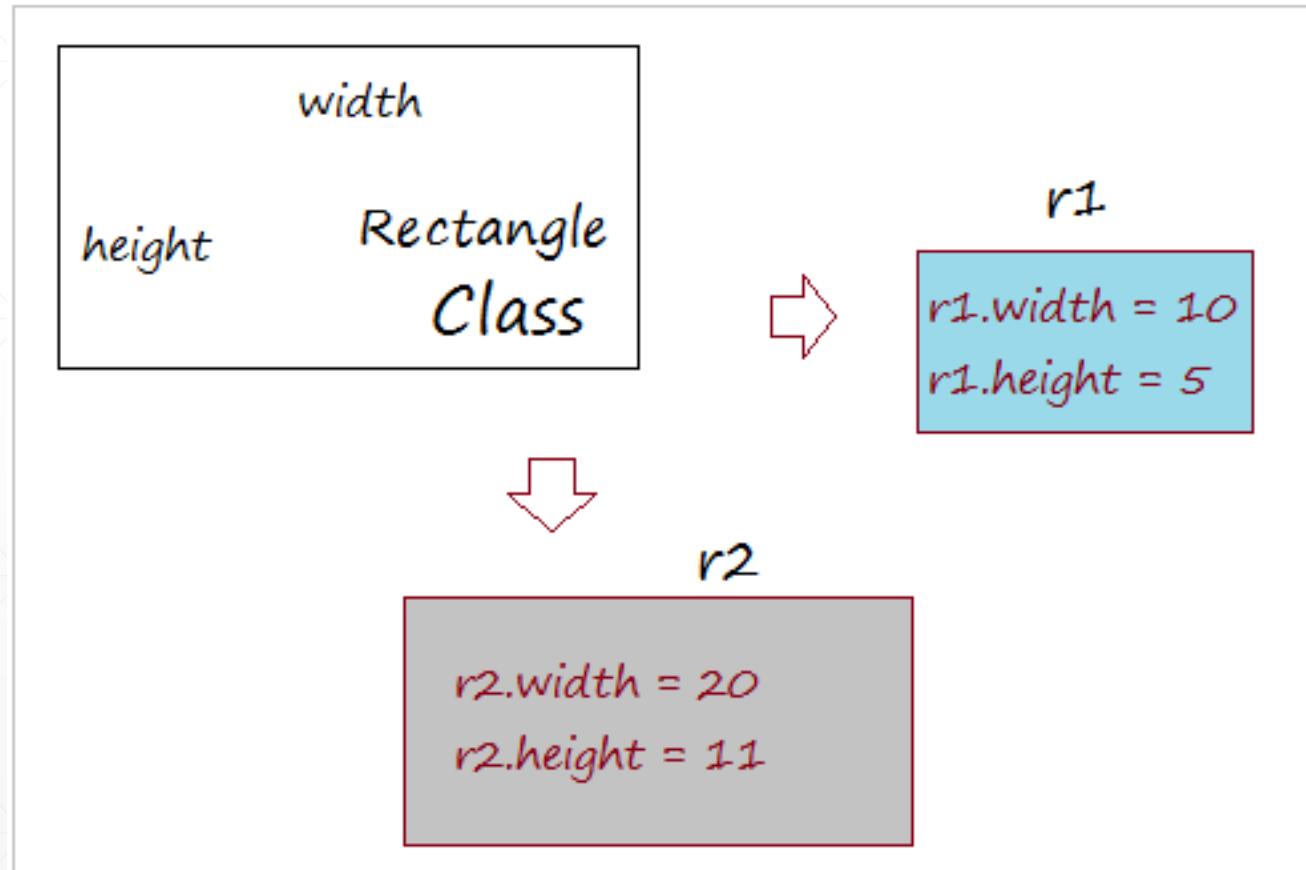
- Ví dụ:

```
def getHeight(self):  
    return self.height  
  
    # Phương thức tính diện tích.  
  
def getArea(self):  
    return self.width * self.height
```



TẠO CLASS TRONG PYTHON

- Tạo đối tượng từ lớp Rectangle:





TẠO CLASS TRONG PYTHON

- **Tạo đối tượng từ lớp Rectangle:**

```
from rectangle import Rectangle  
  
# Tạo 2 đối tượng: r1 & r2  
  
r1 = Rectangle(10,5)  
  
r2 = Rectangle(20,11)  
  
print ("r1.width = ", r1.width)  
  
print ("r1.height = ", r1.height)  
  
print ("r1.getWidth() = ", r1.getWidth())  
  
print ("r1.getArea() = ", r1.getArea())
```



TẠO CLASS TRONG PYTHON

- Tạo đối tượng từ lớp Rectangle:

```
print ("-----")
print ("r2.width = ", r2.width)
print ("r2.height = ", r2.height)
print ("r2.getWidth() = ", r2.getWidth())
print ("r2.getArea() = ", r2.getArea())
```

```
-----> testRectangle.py [C:\DevPrograms\Python\Python36\python.exe]
r1.width = 10
r1.height = 5
r1.getWidth() = 10
r1.getArea() = 50
-----
r2.width = 20
r2.height = 11
r2.getWidth() = 20
r2.getArea() = 220
```

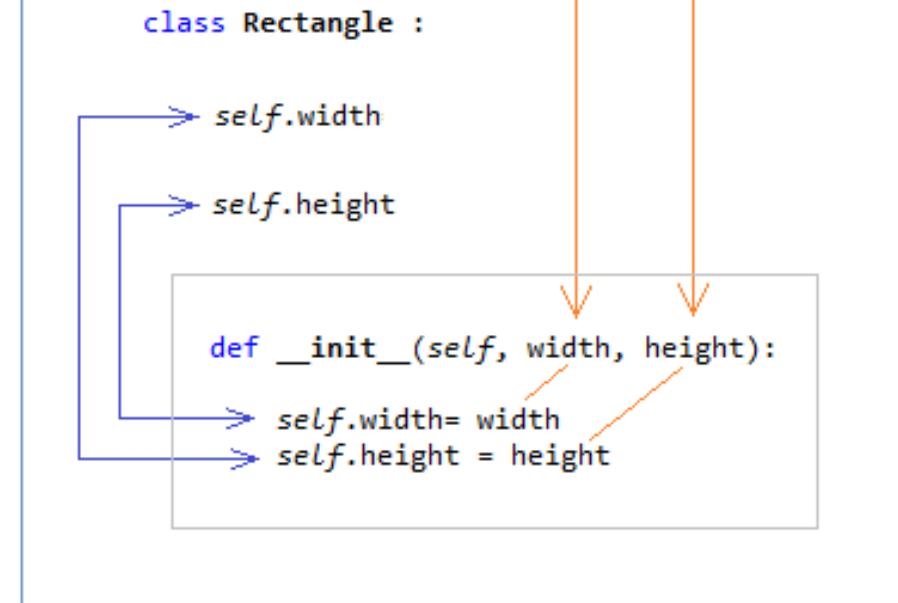


TẠO CLASS TRONG PYTHON

- Điều gì xảy ra khi tạo đối tượng từ một class?

Khi tạo một đối tượng của lớp Rectangle, phương thức khởi tạo (constructor) của class đó sẽ được gọi để tạo một đối tượng, và các thuộc tính của đối tượng sẽ được gán giá trị từ tham số.

```
r1 = Rectangle(10, 5)
```



Chương 6: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG



NỘI DUNG GIẢNG DẠY:

- 6.1. Lập trình hướng đối tượng
- 6.2. Tạo class trong Python
- 6.3. Tham số có mặc định trong Constructor**
- 6.4. So sánh các đối tượng
- 6.5. Thuộc tính
- 6.6. Các hàm truy cập vào thuộc tính
- 6.7. Biến của lớp
- 6.8. Kế thừa và đa kế thừa



THAM SỐ CÓ MẶC ĐỊNH TRONG CONSTRUCTOR

Khác với các ngôn ngữ khác, lớp trong Python chỉ có nhiều nhất một phương thức khởi tạo (constructor), tuy nhiên Python cho phép tham số có giá trị mặc định.

Chú ý: Tất cả các tham số bắt buộc (required parameters) phải đặt trước tất cả các tham số có giá trị mặc định.



THAM SỐ CÓ MẶC ĐỊNH TRONG CONSTRUCTOR

- **Tạo đối tượng Tham số age và gender có giá trị mặc định:**

```
class Person:  
    def __init__(self, name, age = 1, gender = "Male" ):  
        self.name = name  
        self.age = age  
        self.gender= gender  
    def showInfo(self):  
        print ("Name: ", self.name)  
        print ("Age: ", self.age)  
        print ("Gender: ", self.gender)
```



THAM SỐ CÓ MẶC ĐỊNH TRONG CONSTRUCTOR

- Ví dụ sử dụng:

```
from person import Person  
aimee = Person("Aimee", 21, "Female")  
aimee.showInfo()  
print (" ----- ")  
alice = Person( "Alice" )  
alice.showInfo()  
print (" ----- ")  
tran = Person("Tran", 37)  
tran.showInfo()
```

The screenshot shows a Python IDE's console window titled "Console". It displays the output of a script named "testPerson.py" which uses the "Person" class from a module named "person". The output shows three instances of the Person class being created and their details being printed. The first instance is for "Aimee" (Age 21, Female). The second instance is for "Alice" (Age 1, Male). The third instance is for "Tran" (Age 37, Male). Each instance is preceded by a dashed line separator.

```
<terminated> testPerson.py [C:\DevPrograms\Python\Python36\python.exe]  
Name: Aimee  
Age: 21  
Gender: Female  
-----  
Name: Alice  
Age: 1  
Gender: Male  
-----  
Name: Tran  
Age: 37  
Gender: Male
```

Chương 6: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG



NỘI DUNG GIẢNG DẠY:

- 6.1. Lập trình hướng đối tượng
- 6.2. Tạo class trong Python
- 6.3. Tham số có mặc định trong Constructor

6.4. So sánh các đối tượng

- 6.5. Thuộc tính
- 6.6. Các hàm truy cập vào thuộc tính
- 6.7. Biến của lớp
- 6.8. Kế thừa và đa kế thừa

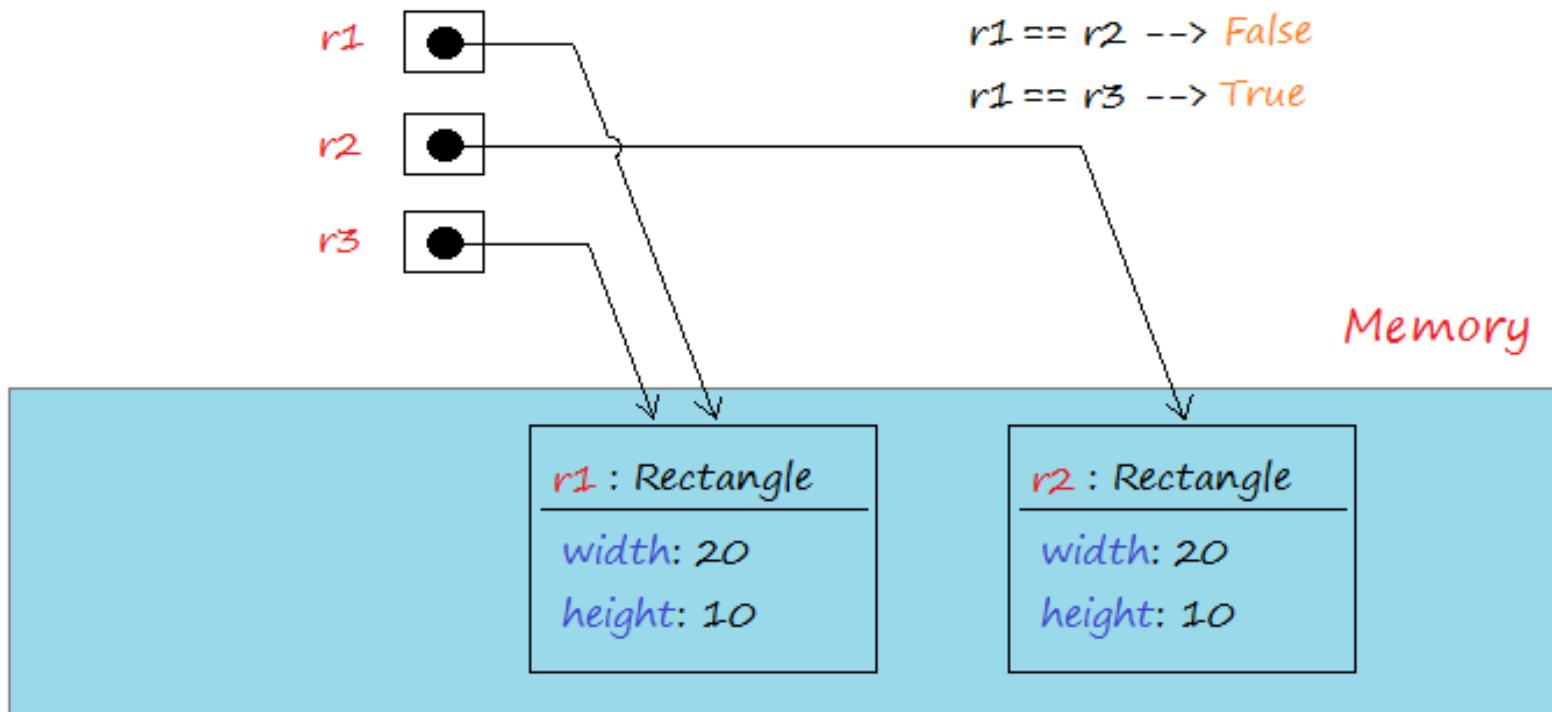


SO SÁNH CÁC ĐỐI TƯỢNG

- Trong Python, khi tạo một đối tượng thông qua phương thức khởi tạo (constructor), sẽ có một thực thể thực sự được tạo ra nằm trên bộ nhớ, nó có một địa chỉ xác định.
- Một phép toán gán đối tượng AA bởi một đối tượng BB không tạo ra thêm thực thể trên bộ nhớ, nó chỉ là trả địa chỉ của AA tới địa chỉ của BB.
- Toán tử `==` dùng để so sánh địa chỉ 2 đối tượng trả đến, nó trả về True nếu cả 2 đối tượng cùng trả tới cùng một địa chỉ trên bộ nhớ.
- Toán tử `!=` cũng sử dụng để so sánh 2 địa chỉ của 2 đối tượng trả đến, nó trả về True nếu 2 đối tượng trả tới 2 địa chỉ khác nhau.

SO SÁNH CÁC ĐỐI TƯỢNG

```
r1 = Rectangle(width=20, height=10)
r2 = Rectangle(width=20, height=10)
r3 = r1
```





BÀI TẬP

CHUẨN BỊ CHO BUỔI HỌC TIẾP THEO:

Đọc các tài liệu về nội dung mục 6.5; 6.6; 6.7 và 6.8.

Chương 6: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG



NỘI DUNG GIẢNG DẠY:

- 6.1. Lập trình hướng đối tượng
- 6.2. Tạo class trong Python
- 6.3. Tham số có mặc định trong Constructor
- 6.4. So sánh các đối tượng

6.5. Thuộc tính

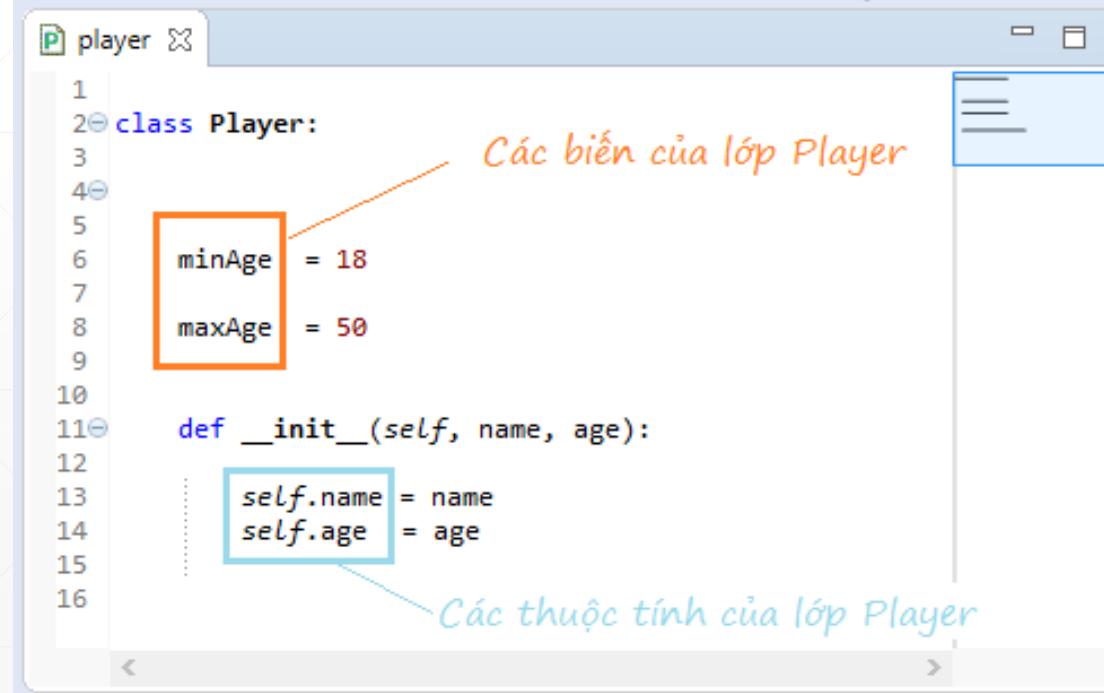
- 6.6. Các hàm truy cập vào thuộc tính
- 6.7. Biến của lớp
- 6.8. Kế thừa và đa kế thừa

THUỘC TÍNH (ATTRIBUTE)

Trong Python có 2 khái niệm khá giống nhau, bạn cần phải phân biệt nó:

- Thuộc tính (Attribute);
- Biến của lớp.

Ví dụ:



```
P player ✘
1
2 class Player:
3
4
5     minAge = 18
6
7     maxAge = 50
8
9
10    def __init__(self, name, age):
11        self.name = name
12        self.age = age
13
14
15
16
```

Các biến của lớp Player

Các thuộc tính của lớp Player

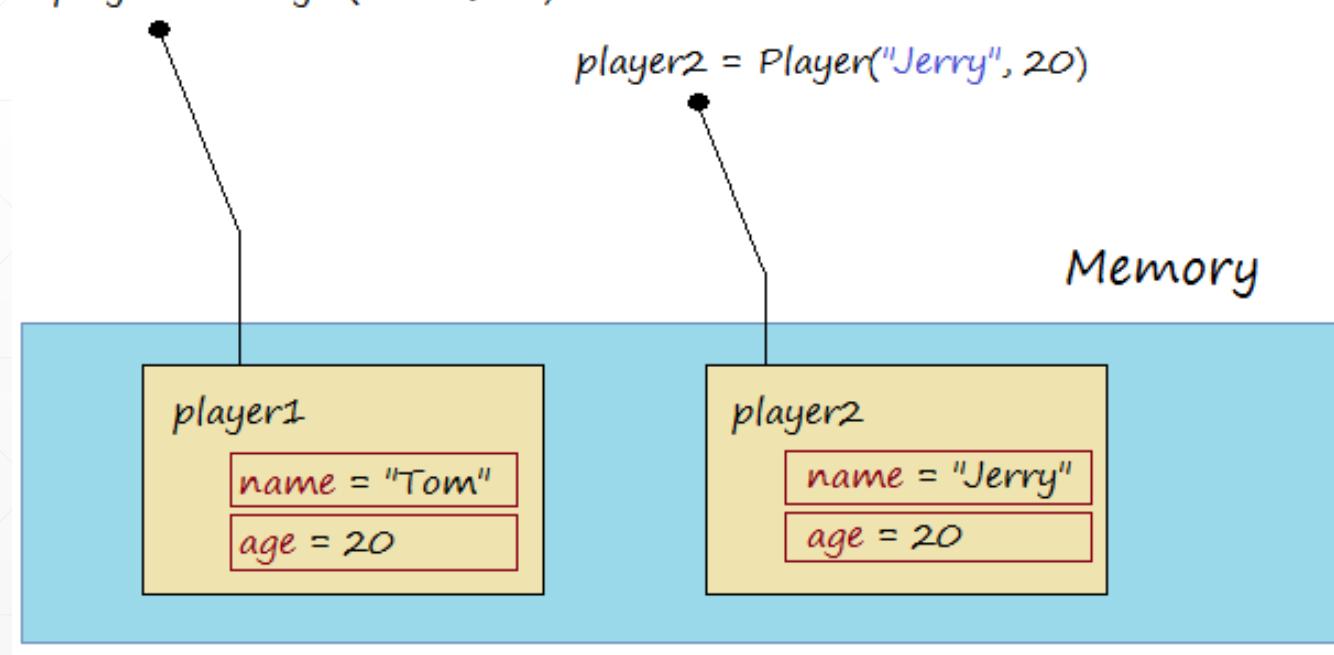
THUỘC TÍNH (ATTRIBUTE)

Các đối tượng được tạo ra từ một lớp, chúng sẽ nằm tại các địa chỉ khác nhau trên bộ nhớ (memory), và các thuộc tính "cùng tên" của chúng cũng có các địa chỉ khác nhau trên bộ nhớ.

Ví dụ:

```
player1 = Player("Tom", 20)
```

```
player2 = Player("Jerry", 20)
```





THUỘC TÍNH (ATTRIBUTE)

Ví dụ:

```
from player import Player  
  
player1 = Player("Tom", 20)  
  
player2 = Player("Jerry", 20)  
  
print ("player1.name = ", player1.name)  
print ("player1.age = ", player1.age)  
  
print ("player2.name = ", player2.name)  
print ("player2.age = ", player2.age)  
  
print ("-----")
```



THUỘC TÍNH (ATTRIBUTE)

Ví dụ:

```
print (" ----- ")
print ("Assign new value to player1.age = 21 ")
# Gán giá trị mới cho thuộc tính age của player1.

player1.age = 21
print ("player1.name = ", player1.name)
print ("player1.age = ", player1.age)
print ("player2.name = ", player2.name)
print ("player2.age = ", player2.age)
```

```
Console X
<terminated> testAttributePlayer.py [C:\DevPrograms\Python\Python36\python.exe]
player1.name = Tom
player1.age = 20
player2.name = Jerry
player2.age = 20
-----
Assign new value to player1.age = 21
player1.name = Tom
player1.age = 21
player2.name = Jerry
player2.age = 20
```

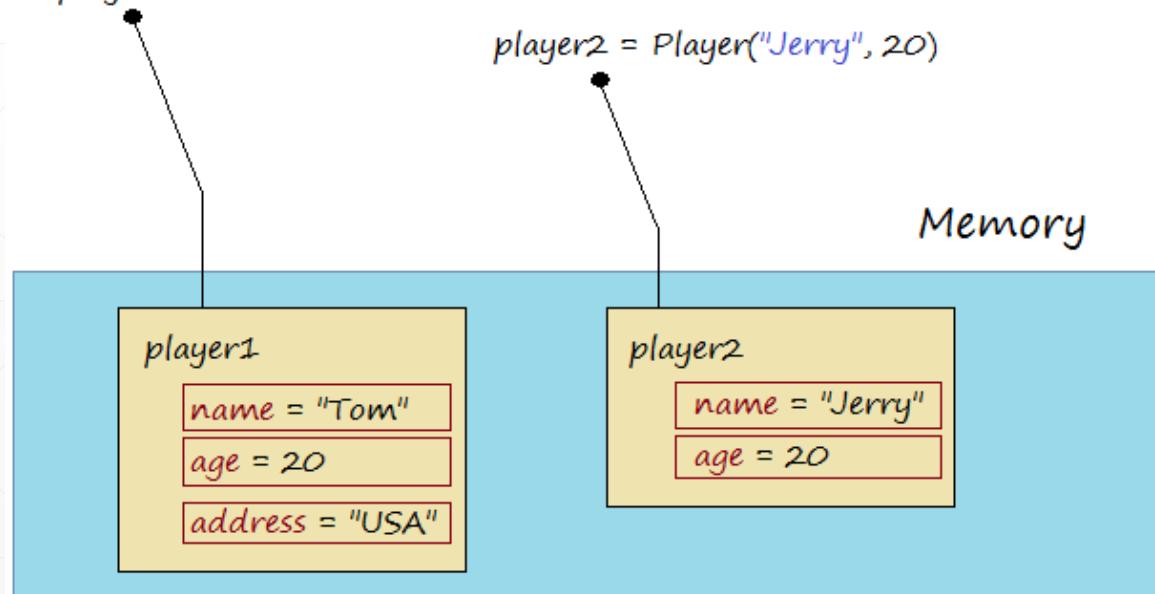
THUỘC TÍNH (ATTRIBUTE)

Python cho phép tạo ra một thuộc tính mới cho một đối tượng có trước.

Ví dụ: Đối tượng player1 và thuộc tính mới có tên address.

```
player1 = Player("Tom", 20)  
player1.address = "USA"
```

```
player2 = Player("Jerry", 20)
```





THUỘC TÍNH (ATTRIBUTE)

Ví dụ:

```
from player import Player  
  
player1 = Player("Tom", 20)  
  
player2 = Player("Jerry", 20)  
  
# Tạo một thuộc tính có tên 'address' cho player1.  
  
player1.address = "USA"  
  
print ("player1.name = ", player1.name)  
  
print ("player1.age = ", player1.age)  
  
print ("player1.address = ", player1.address)
```



THUỘC TÍNH (ATTRIBUTE)

Ví dụ:

```
print ("-----")
print ("player2.name = ", player2.name)
print ("player2.age = ", player2.age)
# player2 không có thuộc tính 'address' (Lỗi xảy ra tại đây).
print ("player2.address = ", player2.address)
```

The screenshot shows a Python console window with the following output:

```
Console <terminated> testNewAttributePlayer.py [C:\DevPrograms\Python\Python36\python.exe]
player1.name = Tom
player1.age = 20
player1.address = USA
-----
player2.name = Jerry
player2.age = 20
Traceback (most recent call last):
  File "E:\ECLIPSE TUTORIAL\PYTHON\PythonClassObject\testNewAttributePlayer.py", line
    print ("player2.address = ", player2.address)
AttributeError: 'Player' object has no attribute 'address'
```

The error message at the bottom, "AttributeError: 'Player' object has no attribute 'address'", is highlighted with a red rectangle.

Chương 6: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG



NỘI DUNG GIẢNG DẠY:

- 6.1. Lập trình hướng đối tượng
- 6.2. Tạo class trong Python
- 6.3. Tham số có mặc định trong Constructor
- 6.4. So sánh các đối tượng
- 6.5. Thuộc tính

6.6. Các hàm truy cập vào thuộc tính

- 6.7. Biến của lớp
- 6.8. Kế thừa và đa kế thừa



CÁC HÀM TRUY CẬP VÀO THUỘC TÍNH

- Thông thường bạn truy cập vào thuộc tính của một đối tượng thông qua toán tử "dấu chấm" (ví dụ player1.name). Tuy nhiên Python cho phép bạn truy cập chúng thông qua hàm (function).

Hàm	Mô tả
getattr(obj, name[, default])	Trả về giá trị của thuộc tính, hoặc trả về giá trị mặc định nếu đối tượng không có thuộc tính này.
hasattr(obj, name)	Kiểm tra xem đối tượng này có thuộc tính cho bởi tham số 'name' hay không.
setattr(obj, name, value)	Set giá trị vào thuộc tính. Nếu thuộc tính không tồn tại, thì nó sẽ được tạo ra.
delattr(obj, name)	Xóa bỏ thuộc tính.



CÁC HÀM TRUY CẬP VÀO THUỘC TÍNH

Ví dụ:

```
from player import Player  
  
player1 = Player("Tom", 20)  
  
# getattr(obj, name[, default])  
  
print ("getattr(player1,'name') = " , getattr(player1,"name") )  
  
print ("setattr(player1,'age', 21): ")  
  
# setattr(obj,name,value)  
  
setattr(player1,"age", 21)  
  
print ("player1.age = ", player1.age)
```



CÁC HÀM TRUY CẬP VÀO THUỘC TÍNH

Ví dụ:

```
# Kiểm tra player1 có thuộc tính (attribute) address hay không?  
hasAddress = hasattr(player1, "address")  
print ("hasattr(player1, 'address') ? ", hasAddress)  
  
# Tạo thuộc tính 'address' cho đối tượng 'player1'.  
print ("Create attribute 'address' for object 'player1'")  
setattr(player1, 'address', "USA")  
print ("player1.address = ", player1.address)
```



CÁC HÀM TRUY CẬP VÀO THUỘC TÍNH

Ví dụ:

```
# Xóa thuộc tính 'address'.
delattr(player1, "address")
```

A screenshot of a Python IDE showing a console window. The title bar of the window says "Console". The console output shows the following code and its execution results:

```
<terminated> testAttFunctions.py [C:\DevPrograms\Python\Python36\python.exe]
getattr(player1, 'name') = Tom
setattr(player1, 'age', 21):
player1.age = 21
hasattr(player1, 'address') ? False
Create attribute 'address' for object 'player1'
player1.address = USA
```

The code demonstrates the use of the `getattr`, `setattr`, and `hasattr` functions to interact with the `player1` object's attributes. The `delattr` function is mentioned in the accompanying text but is not visible in the current screenshot of the IDE.

Chương 6: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG



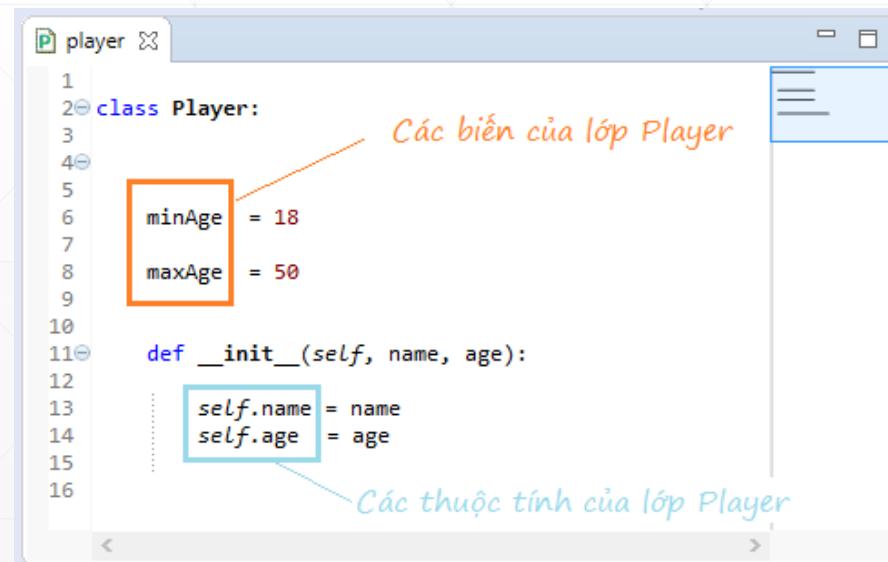
NỘI DUNG GIẢNG DẠY:

- 6.1. Lập trình hướng đối tượng
- 6.2. Tạo class trong Python
- 6.3. Tham số có mặc định trong Constructor
- 6.4. So sánh các đối tượng
- 6.5. Thuộc tính
- 6.6. Các hàm truy cập vào thuộc tính
- 6.7. Biến của lớp**
- 6.8. Kế thừa và đa kế thừa

BIẾN CỦA LỚP

Trong Python khái niệm "Biến của lớp (Class's Variable)" tương đương với khái niệm trường tĩnh (Static Field) của các ngôn ngữ khác như Java, CSharp.

Biến của lớp có thể được truy cập thông qua tên lớp hoặc thông qua đối tượng.



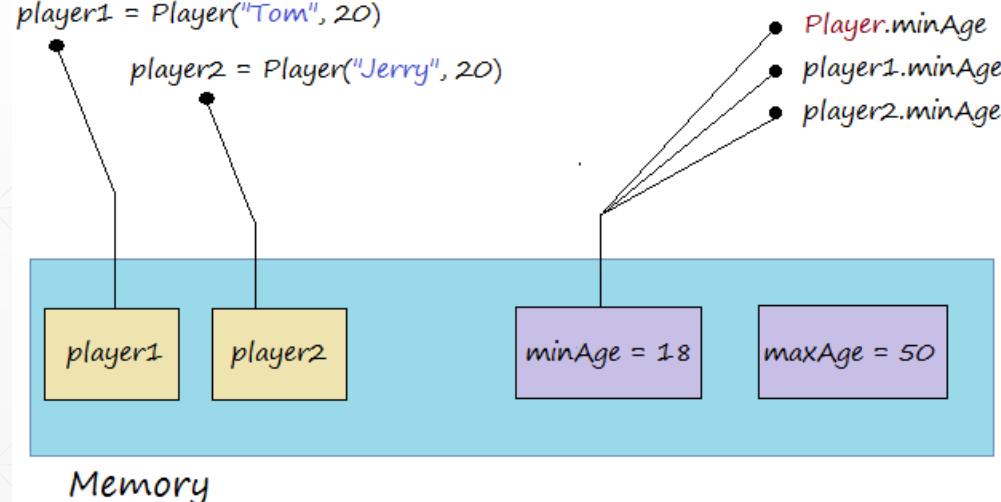
```
player ✘
1
2 class Player:
3
4
5     minAge = 18
6
7     maxAge = 50
8
9
10 def __init__(self, name, age):
11     self.name = name
12     self.age = age
```

BIẾN CỦA LỚP

Nên truy cập "biến của lớp" thông qua tên lớp thay vì truy cập thông qua đối tượng, điều này giúp tránh nhầm lẫn giữa "biến của lớp" và thuộc tính.

Mỗi biến của lớp, có một địa chỉ nằm trên bộ nhớ (memory) và chia sẻ cho mọi đối tượng của lớp.

Ví dụ:





BIẾN CỦA LỚP

Ví dụ:

```
from player import Player  
  
player1 = Player("Tom", 20)  
  
player2 = Player("Jerry", 20)  
  
# Truy cập thông qua tên lớp.  
  
print ("Player.minAge = ", Player.minAge)  
  
# Truy cập thông qua đối tượng.  
  
print ("player1.minAge = ", player1.minAge)  
print ("player2.minAge = ", player2.minAge)
```



BIẾN CỦA LỚP

Ví dụ:

```
print (" ----- ")  
print ("Assign new value to minAge via class name, and print..")  
# Gán một giá trị mới cho minAge thông qua tên lớp.  
Player.minAge = 19  
print ("Player.minAge = ", Player.minAge)  
print ("player1.minAge = ", player1.minAge)  
print ("player2.minAge = ", player2.minAge)
```



BIẾN CỦA LỚP

Ví dụ:

A screenshot of a Windows-style console window titled "Console". The window shows the output of a Python script named "testVariablePlayer.py". The output demonstrates that changing a class variable via its class name affects all instances of that class.

```
<terminated> testVariablePlayer.py [C:\DevPrograms\Python\Python36\python.exe]
Player.minAge = 18
player1.minAge = 18
player2.minAge = 18
-----
Assign new value to minAge via class name, and print..
Player.minAge = 19
player1.minAge = 19
player2.minAge = 19
```

Chương 6: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG



NỘI DUNG GIẢNG DẠY:

- 6.1. Lập trình hướng đối tượng
- 6.2. Tạo class trong Python
- 6.3. Tham số có mặc định trong Constructor
- 6.4. So sánh các đối tượng
- 6.5. Thuộc tính
- 6.6. Các hàm truy cập vào thuộc tính
- 6.7. Biến của lớp
- 6.8. Kế thừa và đa kế thừa**



KẾ THỪA (INHERITANCE)

Kế thừa (Inheritance) cho phép một lớp (class) có thể kế thừa các thuộc tính và phương thức từ các lớp khác đã được định nghĩa.

Lớp đã có gọi là lớp cha (base class hoặc parent class), lớp mới phát sinh gọi là lớp con (child class hoặc derived class).

Lớp con kế thừa tất cả thành phần của lớp cha, có thể mở rộng các thành phần kế thừa và bổ sung thêm các thành phần mới.

class BaseClass:

Body of base class

class DerivedClass(BaseClass):

Body of derived class



KẾ THỪA (INHERITANCE)

Ví dụ:

Đa giác là một hình khép kín có 3 cạnh trở lên. Chúng ta có một lớp gọi là DaGiac được định nghĩa như sau.

```
class DaGiac:  
    def __init__(self, socanh):  
        self.n = socanh  
        self.canh = [0 for i in range(socanh)]  
    def nhapcanh(self):  
        self.canh = [float(input("Bạn hãy nhập giá trị cạnh "+str(i+1)+" : ")) for i in  
range(self.n)]
```



KẾ THỪA (INHERITANCE)

```
def hienthicanh(self):  
    for i in range(self.n):  
        print("Giá trị cạnh",i+1,"là",self.canh[i])
```

Class DaGiac có thuộc tính n để định nghĩa số cạnh và canh để lưu giá trị mỗi cạnh. Hàm nhapcanh() lấy độ lớn các cạnh và hienthicanh() sẽ hiện thị danh sách các cạnh của đa giác.

Hình tam giác là đa giác có ba cạnh, nên ta sẽ tạo một **Lớp TamGiac kế thừa từ DaGiac**. Class mới này sẽ thừa kế tất cả các thuộc tính sẵn có trong lớp cha nên bạn sẽ không cần khai báo lại (khả năng sử dụng lại code).



KẾ THỪA (INHERITANCE)

```
class TamGiac(DaGiac):
    def __init__(self):
        DaGiac.__init__(self,3)

    def dientich(self):
        a, b, c = self.canh
        # Tính nửa chu vi
        s = (a + b + c) / 2
        area = (s*(s-a)*(s-b)*(s-c)) ** 0.5
        print('Diện tích của hình tam giác là %0.2f' %area)
```



KẾ THỪA (INHERITANCE)

Kết quả thực hiện:

```
>>> t = TamGiac()
```

```
>>> t.nhapcanh()
```

Bạn hãy nhập giá trị cạnh 1 : 3

Bạn hãy nhập giá trị cạnh 2 : 5

Bạn hãy nhập giá trị cạnh 3 : 4

```
>>> t.hienthicanh()
```

Giá trị cạnh 1 là 3.0

Giá trị cạnh 2 là 5.0

Giá trị cạnh 3 là 4.0

```
>>> t.dientich()
```

Diện tích của hình tam giác là 6.00



KẾ THỪA (INHERITANCE)

- **Overriding (Ghi đè) trong Python:**

- Python cho phép ghi đè lên các phương thức của lớp cha. Bạn có thể thực hiện việc ghi đè phương thức của lớp cha nếu muốn có tính năng khác biệt hoặc đặc biệt trong lớp con.

- **Ví dụ:** class TamGiac sử dụng lại hàm `__init__()` từ class DaGiac, nếu muốn override (ghi đè) lại định nghĩa của hàm `__init__()` trong class cha, ta dùng hàm `super()`.

`super().__init__(3)` tương đương với `DaGiac.__init__(self, 3)`



KẾ THỪA (INHERITANCE)

- **Overriding (Ghi đè) trong Python:**

Ngoài ra Python có hai hàm `isinstance()` và `issubclass()` được dùng để kiểm tra mối quan hệ của hai lớp và `instance`.

Hàm `issubclass(classA, classB)` trả về `True` nếu class A là lớp con của class B.

Ví dụ:

```
>>> issubclass(DaGiac,TamGiac)
```

```
False
```

```
>>> issubclass(TamGiac,DaGiac)
```

```
True
```

```
>>> issubclass(bool,int)
```

```
True
```



KẾ THỪA (INHERITANCE)

- **Overriding (Ghi đè) trong Python:**

Hàm `isinstance(obj, Class)` trả về `True` nếu `obj` là một instance của lớp `Class` hoặc là một instance của lớp con của `Class`.

Ví dụ:

```
>>> isinstance(t,TamGiac)
```

```
True
```

```
>>> isinstance(t,DaGiac)
```

```
True
```

```
>>> isinstance(t,int)
```

```
False
```

```
>>> isinstance(t,object)
```

```
True
```



ĐA KẾ THỪA (MULTIPLE INHERITANCE)

Trong Python một lớp có thể được định nghĩa từ nhiều lớp cha. Điều này được gọi là đa kế thừa.

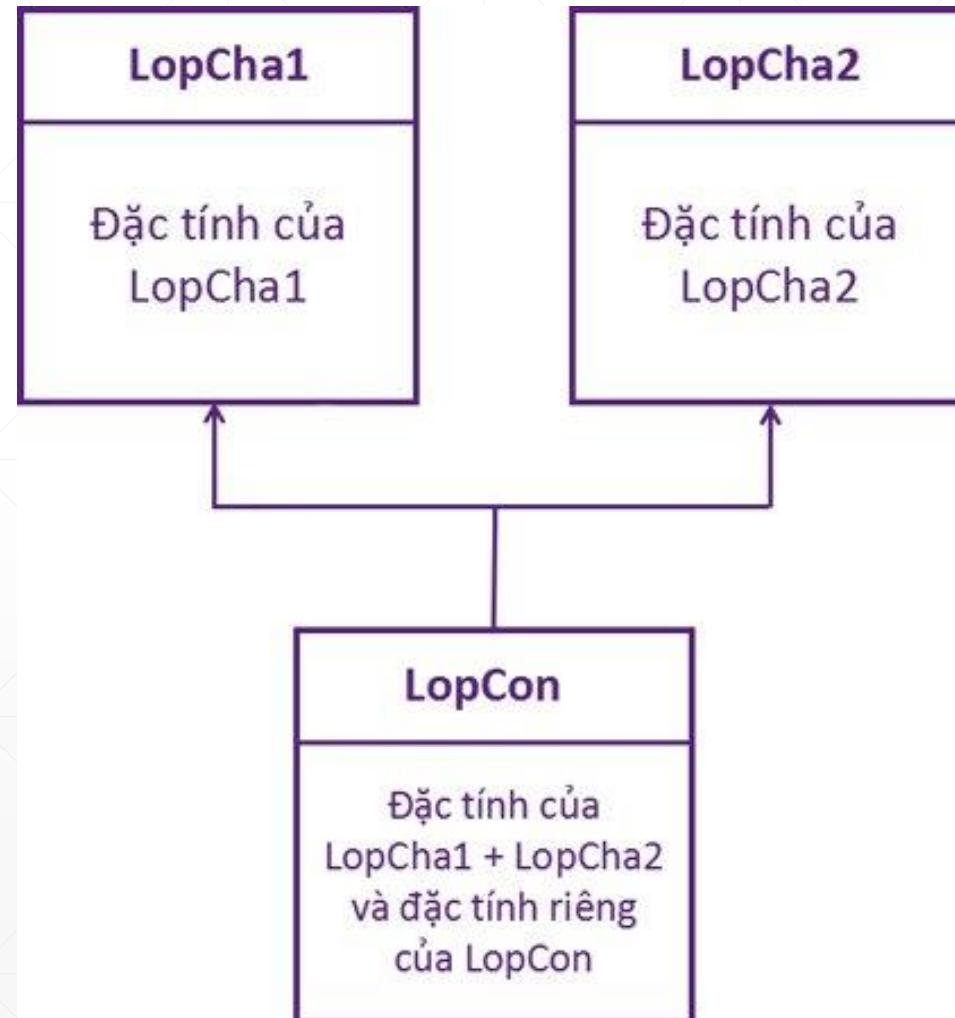
```
class LopCha1:  
    pass
```

```
class LopCha2:  
    pass
```

```
class LopCon(LopCha1, LopCha2):  
    pass
```



ĐA KẾ THỪA (MULTIPLE INHERITANCE)





ĐA KẾ THỪA (MULTIPLE INHERITANCE)

- Lớp con được định nghĩa từ nhiều lớp cha và kế thừa đặc tính của cả hai lớp.
- Các lớp cha có thể có các thuộc tính hoặc các phương thức giống nhau.
- Lớp con sẽ ưu tiên thừa kế thuộc tính, phương thức của lớp đứng đầu tiên trong danh sách thừa kế.



ĐA KẾ THỪA (MULTIPLE INHERITANCE)

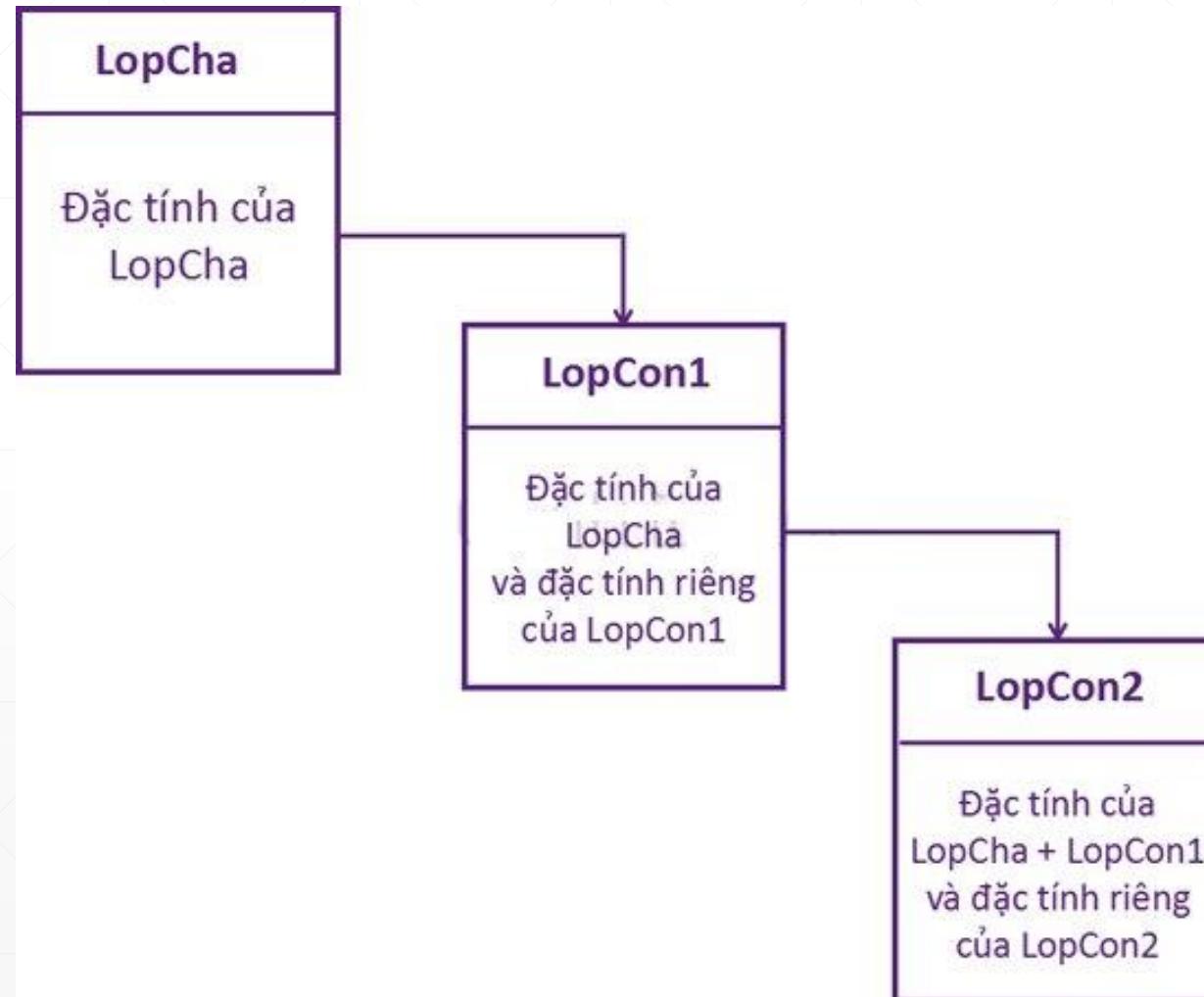
• Kế thừa đa cấp (Multilevel Inheritance)

Ngoài việc có thể kế thừa từ các lớp cha, có thể tạo lớp con mới kế thừa các lớp con trước đó, gọi là kế thừa đa cấp. Trường hợp này, các đặc tính của lớp cha và lớp con trước đó sẽ được lớp con mới kế thừa.

```
class LopCha:  
    pass  
  
class LopCon1(LopCha):  
    pass  
  
class LopCon2(LopCon1):  
    pass
```



ĐA KẾ THỪA (MULTIPLE INHERITANCE)





ĐA KẾ THỪA (MULTIPLE INHERITANCE)

- **Thứ tự truy xuất phương thức (Method Resolution Order)**

Class được bắt nguồn từ object, trong kịch bản đa thừa kế, bất kỳ thuộc tính cần được truy xuất nào, đầu tiên sẽ được tìm kiếm trong lớp hiện tại. Nếu không tìm thấy, tìm kiếm tiếp tục vào lớp cha đầu tiên và từ trái qua phải.

Vậy thứ tự truy xuất sẽ là [LopCon, LopCha1, LopCha2, object].

Thứ tự này còn được gọi là tuyển tính hóa của LopCon và tập hợp các quy tắc được sử dụng để tìm thứ tự này được gọi là Thứ tự truy xuất phương thức (MRO).



ĐA KẾ THỪA (MULTIPLE INHERITANCE)

- **Thứ tự truy xuất phương thức (Method Resolution Order)**

- Nói một cách dễ hiểu, MRO dùng để hiển thị danh sách/tuple các class cha của một class nào đó.
- MRO được sử dụng theo hai cách:
 - __mro__: trả về một tuple
 - mro(): trả về một danh sách.



BÀI TẬP

NỘI DUNG:

1. Định nghĩa class Nguoi và 2 class con của nó: Nam, Nu. Tất cả các class có method "getGender" có thể in "Nam" cho class Nam và "Nữ" cho class Nu. .
2. Định nghĩa một class có tên là Shape và class con là Square. Square có hàm init để lấy đối số là chiều dài. Cả 2 class đều có hàm area để in diện tích của hình, diện tích mặc định của Shape là 0.



BÀI TẬP

CHUẨN BỊ CHO BUỔI HỌC TIẾP THEO:

1. Đọc các tài liệu về nội dung mục 7.1; 7.2.
2. Tìm hiểu khái niệm về tập tin và thư mục trong hệ điều hành.



**TRƯỜNG ĐẠI HỌC VINH
VIỆN KỸ THUẬT VÀ CÔNG NGHỆ**

CHƯƠNG 7

**THAO TÁC TRÊN TẬP TIN
VÀ THƯ MỤC TRONG PYTHON**

Nghệ An, 2018

Chương 7:

THAO TÁC TRÊN TẬP TIN VÀ THƯ MỤC TRONG PYTHON



NỘI DUNG GIẢNG DAY:

- 7.1. Tập tin (File)
- 7.2. Thư mục (Directory)

Chương 7: THAO TÁC TRÊN TẬP TIN VÀ THƯ MỤC TRONG PYTHON



NỘI DUNG GIẢNG DAY:

7.1. Tập tin (File)

7.2. Thư mục (Directory)



TẬP TIN (FILE)

• FILE LÀ GÌ?

- File hay còn gọi là tệp, tập tin. File là tập hợp của các thông tin được đặt tên và lưu trữ trên bộ nhớ máy tính như đĩa cứng, đĩa mềm, CD, DVD,....
- Khi muốn đọc hoặc ghi file, chúng ta cần phải mở file trước. Khi hoàn thành, file cần phải được đóng lại để các tài nguyên được gắn với file được giải phóng.
- Do đó, trong Python, một thao tác với file diễn ra theo thứ tự sau: Mở tệp tin → Đọc hoặc ghi → Đóng tệp



TẬP TIN (FILE)

- **MỞ FILE TRONG PYTHON:**

- Trước khi làm việc với bất cứ file nào, chúng ta phải mở file đó.
- Để mở một file, Python cung cấp hàm `open()`, hàm này trả về một đối tượng file mà được sử dụng với các hàm khác.
- Với file đã mở, bạn có thể thực hiện các hoạt động như đọc, ghi mới, ghi thêm ... trên file đó.



TẬP TIN (FILE)

- **MỞ FILE TRONG PYTHON:**

```
file object = open(file_name [, access_mode][, buffering])
```

Trong đó:

- ***filename***: là một giá trị chuỗi chứa tên của các file mà bạn muốn truy cập.
- ***access_mode***: xác định các chế độ của file được mở ra như read, write, append,... , đây là thông số tùy chọn và chế độ truy cập file mặc định là read (r).



TẬP TIN (FILE)

- ***buffering:***

- + Nếu buffer được thiết lập là 0, nghĩa là sẽ không có trình đệm nào diễn ra.
- + Nếu xác định là 1, thì trình đệm dòng được thực hiện trong khi truy cập một File.
- + Nếu là số nguyên lớn hơn 1, thì hoạt động đệm được thực hiện với kích cỡ bộ đệm đã cho.
- + Nếu là số âm, thì kích cỡ bộ đệm sẽ là mặc định.



THẢO LUẬN NHÓM

NỘI DUNG:

Danh sách các chế độ (mode) khác nhau khi mở một file. Ví dụ.



TẬP TIN (FILE)

• THUỘC TÍNH CỦA FILE:

Thuộc tính	Mô tả
file.closed	Trả về True nếu file đã đóng, ngược lại là False
file.mode	Trả về chế độ truy cập của file đang được mở
file.name	Trả về tên của file

```
file = open("plc.txt", "wb")
print "Tên của file là: ", file.name
print "File có đóng hoặc không? : ", file.closed
print "Chế độ mở file : ", file.mod
```



TẬP TIN (FILE)

• ĐÓNG FILE:

- Khi đã thực hiện xong các hoạt động trên file thì cuối cùng chúng ta cần đóng file đó.
- Python tự động đóng một file khi đối tượng tham chiếu của một file đã được tái gán cho một file khác. Tuy nhiên, sử dụng phương thức `close()` để đóng một file vẫn tốt hơn.

Cú pháp:

```
fileObject.close()
```

```
# Mở file
```

```
file = open("plc.txt", "r")
```

```
# Đóng file
```

```
file.close()
```



TẬP TIN (FILE)

- **ĐỌC FILE:**

- **Phương thức read**

Cú pháp:

```
fileObject.read([size])
```

- + Phương thức này trả về một chuỗi có kích thước bằng size.
- + Nếu không truyền size thì toàn bộ nội dung của file sẽ được đọc.



TẬP TIN (FILE)

- **ĐỌC FILE:**

- **Phương thức read**

Giả sử chúng ta có một file vidu.txt với nội dung như sau:

```
Hello all!
```

```
My name's Phuc.
```

Ví dụ:

```
f = open('vidu.txt', 'r')  
str= f.read()  
print ('Nội dung file là:\n', str)
```

Kết quả in ra màn hình:

```
Nội dung file là:  
Hello all!  
My name's Phuc.
```



TẬP TIN (FILE)

- **ĐỌC FILE:**

- **Phương thức readline**

Cú pháp:

```
fileObject.readline()
```

- + Phương thức này cho phép đọc một dòng trong file và trả về chuỗi.



TẬP TIN (FILE)

- **ĐỌC FILE:**

- **Phương thức readline**

Giả sử chúng ta có một file vidu.txt với nội dung như sau:

Ví dụ:

```
f = open('vidu.txt', 'r')  
line1 = f.readline()  
line2 = f.readline()  
print ('Dòng 1: ', line1)  
print ('Dòng 2: ', line2)
```

Kết quả in ra màn hình:

Dòng 1: Hello all!
Dòng 2: My name's Phuc.



TẬP TIN (FILE)

• GHI FILE:

Tương tự đọc file, để ghi một file ta cần mở file bằng cú pháp để ghi và sử dụng phương thức write để ghi vào.

Cú pháp:

```
fileObject.write(string)
```

- Phương thức này cho phép ghi một chuỗi có nội dung là string vào vị trí của con trỏ trong file.



TẬP TIN (FILE)

- **GHI FILE:**

Ví dụ:

```
# Mở file  
file = open("plc.txt", "wb")  
file.write( "Python là ngôn ngữ tốt nhất");  
# Đóng file  
file.close()
```

Nội dung bên trong file plc.txt sau khi thực hiện ghi file:

Python là ngôn ngữ tốt nhất



TẬP TIN (FILE)

• THAY TÊN FILE:

Phương thức `rename()` trong module `os` được sử dụng để thay tên file. Phương thức này nhận hai tham số là tên file cũ và tên file mới.

Cú pháp:

```
os.rename("<tên file hiện tại>", "<tên file mới>")
```

Ví dụ:

```
import os  
  
#Thay tên plc1.txt thành plc2.txt:  
os.rename( "plc1.txt", "plc2.txt" )
```



TẬP TIN (FILE)

• XÓA FILE:

Có thể sử dụng **phương thức remove()** của module os để xóa các file với tham số là tên file cần xóa.

Cú pháp:

```
os.remove("<tên file>")
```

Ví dụ:

```
import os  
# Xóa plc2.txt  
os.remove("plc2.txt")
```



TẬP TIN (FILE)

• VÍ TRÍ FILE:

- Phương thức `tell()` sẽ cho biết vị trí hiện tại bên trong file. Nói cách khác, việc đọc và ghi tiếp theo sẽ diễn ra trên các byte đó.
- Phương thức `seek(offset[, from])` thay đổi vị trí hiện tại bên trong file.
 - + Tham số `offset` là chỉ số byte để được di chuyển.
 - + Tham số `from` xác định vị trí tham chiếu mà từ đó byte được di chuyển.

Nếu `from` là 0 thì sử dụng phần đầu file như là vị trí tham chiếu.

Nếu `from` là 2 thì sử dụng phần cuối file như là vị trí tham chiếu.



TẬP TIN (FILE)

- VÍ TRÍ FILE:

Ví dụ:

```
# Mở file
file = open("plc.txt", "r+")
str = file.read(10);
print "Chuỗi đã đọc là: ", str
# Kiểm tra con trỏ hiện tại
vitri = file.tell();
print "Con trỏ hiện tại: ", vitri
```



TẬP TIN (FILE)

- VÍ TRÍ FILE:

```
# Đặt lại vị trí con trỏ tại vị trí đầu file  
vitri = file.seek(0, 0);  
  
str = file.read(10);  
  
print "Chuỗi đã đọc là : ", str  
  
# Đóng file  
file.close()
```



TẬP TIN (FILE)

- VÍ TRÍ FILE:

Kết quả hiện thị trên màn hình như sau:

Chuỗi đã đọc là : Python là

Con trỏ hiện tại : 10

Chuỗi đã đọc là : Python là

Chương 7: THAO TÁC TRÊN TẬP TIN VÀ THƯ MỤC TRONG PYTHON



NỘI DUNG GIẢNG DAY:

7.1. Tập tin (File)

7.2. Thư mục (Directory)



THƯ MỤC (DIRECTORY)

Python cũng cung cấp rất nhiều phương thức để xử lý các hoạt động đa dạng liên quan tới thư mục. **Module os** được xây dựng để cung cấp các phương thức giúp tạo, xóa, và thay đổi các thư mục.

- **HIỂN THỊ THƯ MỤC HIỆN TẠI:**

- Phương thức `getcwd()` hiển thị thư mục đang làm việc hiện tại, trả về kết quả dưới dạng một chuỗi.
- Có thể sử dụng phương thức `getcwdb()` để nhận về kết quả dưới dạng byte.



THƯ MỤC (DIRECTORY)

Python cũng cung cấp rất nhiều phương thức để xử lý các hoạt động đa dạng liên quan tới thư mục. **Module os** được xây dựng để cung cấp các phương thức giúp tạo, xóa, và thay đổi các thư mục.

- **HIỂN THỊ THƯ MỤC HIỆN TẠI:**

- Phương thức `getcwd()` hiển thị thư mục đang làm việc hiện tại, trả về kết quả dưới dạng một chuỗi.

- Có thể sử dụng **phương thức `getcwdb()`** để nhận về kết quả dưới dạng byte.

```
>>> import os
```

```
>>> os.getcwd()
```

```
'C:\\Program Files\\PyScripter'
```

```
>>> os.getcwdb()
```

```
b'C:\\Program Files\\PyScripter'
```



THƯ MỤC (DIRECTORY)

• THAY ĐỔI THƯ MỤC HIỆN TẠI

Thư mục làm việc hiện tại có thể được thay đổi bằng **phương thức chdir()**.

- **chdir()** nhận một tham số là tên thư mục muốn tới từ thư mục hiện tại.
- Có thể sử dụng cả dấu gạch chéo (/) hoặc dấu gạch chéo ngược (\) để tách các phần tử trong đường dẫn, nhưng tốt nhất vẫn nên sử dụng dấu gạch ngược (\).

```
>>> os.chdir('c:\\Python33')  
  
>>> print(os.getcwd())  
c:\\Python33
```



THƯ MỤC (DIRECTORY)

• DANH SÁCH THƯ MỤC VÀ FILE

Có thể liệt kê tất cả các tệp và thư mục con bên trong một thư mục bằng cách sử dụng **phương thức listdir()**.

- Phương thức này nhận một đường dẫn và trả về danh sách thư mục con và các file trong đường dẫn đó.

- Nếu không có đường dẫn nào được chỉ định, kết quả trả về sẽ truy xuất từ thư mục làm việc hiện tại.

```
>>> print(os.getcwd())
C:\Python33
>>> os.listdir()
['DLLs',
'Doc',
'include',
'Lib',
'libs',
'LICENSE.txt',
'NEWS.txt',
'python.exe',
'pythonw.exe',
'README.txt',
'Scripts',
'tcl',
'Tools']
```



THƯ MỤC (DIRECTORY)

• TẠO MỘT THƯ MỤC MỚI

Để tạo các thư mục mới, bạn sử dụng **phương thức mkdir()** của Module os.

- Có thể chọn nơi chứa thư mục mới bằng cách ghi đầy đủ đường dẫn tới nơi muốn tạo.
- Nếu đường dẫn đầy đủ không được chỉ định, thư mục mới sẽ được tạo trong thư mục làm việc hiện tại.

```
>>> os.mkdir('test')  
  
>>> os.listdir()  
['test']
```



THƯ MỤC (DIRECTORY)

• ĐỔI TÊN THƯ MỤC HOẶC TÊN FILE

Sử dụng **phương thức rename()** để đổi tên một thư mục hoặc một tập tin.

```
>>> os.listdir()  
['test']  
  
>>> os.rename('test','new_one')  
  
>>> os.listdir()  
['new_one']
```



THƯ MỤC (DIRECTORY)

- **XÓA BỎ THƯ MỤC HOẶC FILE**

- Để gỡ bỏ một file, sử dụng phương thức `remove()`
- Để xóa toàn bộ thư mục, sử dụng phương thức `rmdir()`.
- Phương thức `rmdir()` chỉ có thể xóa các thư mục rỗng.

```
>>> os.listdir()  
['new_one', 'old.txt']  
  
>>> os.remove('old.txt')  
  
>>> os.listdir()  
['new_one']  
  
>>> os.rmdir('new_one')  
  
>>> os.listdir()  
[]
```



THƯ MỤC (DIRECTORY)

• XÓA BỎ THƯ MỤC HOẶC FILE

- Để loại bỏ một thư mục không rỗng, chúng ta có thể sử dụng phương thức `rmtree()` bên trong module `shutil`.

```
>>> os.listdir()
['test']

>>> os.rmdir('test')
Traceback (most recent call last):
...
OSError: [WinError 145] The directory is not empty: 'test'

>>> import shutil

>>> shutil.rmtree('test')
>>> os.listdir()
[]
```



BÀI TẬP

NỘI DUNG:

Tạo một thư mục và các file. Sử dụng các hàm thực hiện đổi tên của file và thư mục.



BÀI TẬP

CHUẨN BỊ CHO BUỔI HỌC TIẾP THEO:

1. Đọc các tài liệu về nội dung mục 8.1; 8.2 và 8.3.
2. Tìm hiểu về cách tổ chức giao diện người dùng.



**TRƯỜNG ĐẠI HỌC VINH
VIỆN KỸ THUẬT VÀ CÔNG NGHỆ**

CHƯƠNG 8

LẬP TRÌNH GIAO DIỆN TRONG PYTHON

Nghệ An, 2018

Chương 8:

LẬP TRÌNH GIAO DIỆN TRONG PYTHON



NỘI DUNG GIẢNG DAY:

- 8.1. Giới thiệu về lập trình giao diện và thư viện Tkinter
- 8.2. Quản lý Layout
- 8.3. Widget
- 8.4. Menu
- 8.5. Hộp thoại
- 8.6. Đồ họa

Chương 8:

LẬP TRÌNH GIAO DIỆN TRONG PYTHON



NỘI DUNG GIẢNG DAY:

8.1. Giới thiệu về lập trình giao diện và thư viện Tkinter

8.2. Quản lý Layout

8.3. Widget

8.4. Menu

8.5. Hộp thoại

8.6. Đồ họa

GIỚI THIỆU VỀ LẬP TRÌNH GIAO DIỆN VÀ THƯ VIỆN TKINTER



- **Khái niệm về chương trình có giao diện đồ họa**

- Chương trình có giao diện đồ họa là một chương trình luôn chạy cho tới khi người dùng thoát chương trình (có *chạy 1 vòng lặp vô hạn để luôn hiển thị giao diện, gọi là main loop*).

Chương trình đồ họa hoạt động dựa trên những tương tác của người dùng và phản ứng với các tương tác đó (*bấm nút nào thì thực hiện công việc tương ứng*). Loại chương trình như vậy thuộc loại mô hình "*Event-driven programming*".

GIỚI THIỆU VỀ LẬP TRÌNH GIAO DIỆN VÀ THƯ VIỆN TKINTER



- **Khái niệm về chương trình có giao diện đồ họa**
 - Trong chương trình đồ họa, các thao tác của người dùng được gọi là các event, các hành động tương ứng của chương trình (các function) được gọi là các callback, được gắn vào các bộ phận giao diện (gắn callback vào nút bấm thì khi ta bấm nút, callback sẽ được gọi).
 - Các bộ phận giao diện như nút bấm, chữ, ô nhập ký tự ... được gọi là các widget.

GIỚI THIỆU VỀ LẬP TRÌNH GIAO DIỆN VÀ THƯ VIỆN TKINTER



- So sánh chương trình có giao diện kiểu dòng lệnh (**Command Line Interface - CLI**) và chương trình có giao diện đồ họa (**Graphical User Interface - GUI**)

CLI	GUI
<ul style="list-style-type: none">- Chiếm ít tài nguyên.- Người dùng có nhiều quyền xử lý hệ thống.- Chỉ việc gõ một vài dòng để thực hiện một việc.	<ul style="list-style-type: none">- Dễ dàng hơn cho người dùng khi tương tác với ứng dụng.- Có khả năng hoạt động đa nhiệm.

GIỚI THIỆU VỀ LẬP TRÌNH GIAO DIỆN VÀ THƯ VIỆN TKINTER



• Giới thiệu thư viện Tkinter

- Tkinter là một gói trong Python có chứa module Tk hỗ trợ cho việc lập trình GUI.
- Tk ban đầu được viết cho ngôn ngữ Tcl, sau đó Tkinter được viết ra để sử dụng Tk bằng trình thông dịch Tcl trên nền Python.
- Ngoài Tkinter ra còn có một số công cụ khác giúp tạo một ứng dụng GUI viết bằng Python như wxPython, PyQt, và PyGTK.

GIỚI THIỆU VỀ LẬP TRÌNH GIAO DIỆN VÀ THƯ VIỆN TKINTER



Để tạo ra một khung cửa sổ đơn giản trong Tkinter chỉ cần làm như sau:

```
from tkinter import Tk, Frame, BOTH

class Example(Frame):
    def __init__(self, parent):
        Frame.__init__(self, parent, background="white")
        self.parent = parent
        self.initUI()

    def initUI(self):
        pass
```

GIỚI THIỆU VỀ LẬP TRÌNH GIAO DIỆN VÀ THƯ VIỆN TKINTER



```
def initUI(self):  
    self.parent.title("Simple")  
    self.pack(fill=BOTH, expand=1)  
  
root = Tk()  
root.geometry("250x150+300+300")  
app = Example(root)  
root.mainloop()
```



Chương 8: LẬP TRÌNH GIAO DIỆN TRONG PYTHON



NỘI DUNG GIẢNG DAY:

8.1. Giới thiệu về lập trình giao diện và thư viện Tkinter

8.2. Quản lý Layout

8.3. Widget

8.4. Menu

8.5. Hộp thoại

8.6. Đồ họa



QUẢN LÝ LAYOUT

Trong Tkinter có hai loại widget:

- Các widget thường như nút bấm, textbox...
- Container, đây là những widget chứa các widget khác (còn được gọi là layout).

Trong Tkinter có 3 loại layout:

- **Place** là kiểu layout tự do, tức là bạn sẽ phải tự quy định vị trí cũng như kích thước của các widget.
- **Pack** sắp xếp các widget của bạn theo chiều ngang và chiều dọc.
- **Grid** sắp xếp widget theo dạng bảng.



QUẢN LÝ LAYOUT

- **Place:**

- Khi thiết kế giao diện thường dùng layout kiểu tự do:
 - Người lập trình phải tự quy định vị trí và kích thước cho các widget trên màn hình, nếu kích thước cửa sổ thay đổi thì kích thước và vị trí của các widget không thay đổi.
 - Thiết kế theo kiểu tự do cũng rất bất tiện khi muốn thay đổi, thêm hay bớt các widget thì hầu như sẽ phải sắp xếp lại toàn bộ widget.



QUẢN LÝ LAYOUT

Ví dụ:

```
import Tkinter as tk  
import random  
root = tk.Tk()  
  
# width x height + x_offset + y_offset:  
root.geometry("170x200+30+30")  
  
languages = ['Python','Perl','C++','Java','Tcl/Tk']  
labels = range(5)
```



QUẢN LÝ LAYOUT

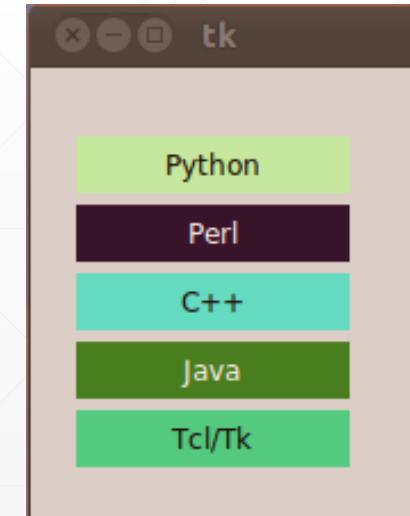
Ví dụ:

```
for i in range(5):
    ct = [random.randrange(256) for x in range(3)]
    brightness = int(round(0.299*ct[0] + 0.587*ct[1] +
0.114*ct[2]))
    ct_hex = "%02x%02x%02x" % tuple(ct)
    bg_colour = '#' + "".join(ct_hex)
    l = tk.Label(root,
                 text=languages[i],
```

QUẢN LÝ LAYOUT

Ví dụ:

```
fg='White' if brightness < 120 else 'Black',  
        bg=bg_colour)  
  
l.place(x = 20, y = 30 + i*30, width=120, height=25)  
  
root.mainloop()
```





QUẢN LÝ LAYOUT

- **Pack:**

Dễ sử dụng nhất trong ba trình quản lý hình học của Tk và Tkinter.

- Thay vì phải khai báo chính xác vị trí của một widget sẽ xuất hiện trên màn hình hiển thị, chúng ta có thể khai báo vị trí của các widget bằng lệnh pack tương đối với nhau. Lệnh pack sẽ chăm sóc các chi tiết.

- Trình quản lý layout này bị hạn chế về khả năng so với các trình quản lý **Grid** và **Place**. Đối với các ứng dụng đơn giản, nó chắc chắn là người quản lý của sự lựa chọn. Ví dụ, các ứng dụng đơn giản như đặt một số vật dụng cạnh nhau hoặc đặt lên nhau.



QUẢN LÝ LAYOUT

Ví dụ:

```
from Tkinter import *
root = Tk()
Label(root, text="Red Sun", bg="red", fg="white").pack()
Label(root, text="Green Grass", bg="green", fg="black").pack()
Label(root, text="Blue Sky", bg="blue", fg="white").pack()
mainloop()
```

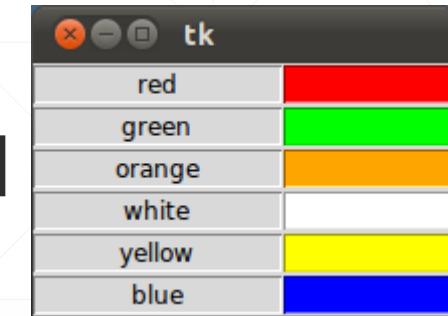




QUẢN LÝ LAYOUT

Ví dụ:

```
from Tkinter import *
colours = ['red','green','orange','white','yellow','blue']
r = 0
for c in colours:
    Label(text=c, relief=RIDGE,width=15).grid(row=r,column=0)
    Entry(bg=c, relief=SUNKEN,width=10).grid(row=r,column=1)
    r = r + 1
mainloop()
```





QUẢN LÝ LAYOUT

- **Grid:**

Trong nhiều trường hợp là sự lựa chọn tốt nhất.

Grid đặt các widget trong bảng 2 chiều, bao gồm một số hàng và cột.

- Vị trí của một widget được xác định bởi một hàng và một số cột.
- Các widget có cùng số cột và số hàng khác nhau sẽ ở trên hoặc dưới nhau. Tương ứng, các vật dụng có cùng số hàng nhưng số cột khác nhau sẽ nằm trên cùng một "dòng" và sẽ nằm cạnh nhau, tức là ở bên trái hoặc bên phải.

Chương 8:

LẬP TRÌNH GIAO DIỆN TRONG PYTHON



NỘI DUNG GIẢNG DAY:

- 8.1. Giới thiệu về lập trình giao diện và thư viện Tkinter
- 8.2. Quản lý Layout
- 8.3. Widget**
- 8.4. Menu
- 8.5. Hộp thoại
- 8.6. Đồ họa



WIDGET

Widget là các thành phần cấu tạo nên một ứng dụng GUI.

Widget rất đa dạng, có một số widget quan trọng cần phải có của bất kì nền tảng nào kể cả Tkinter ví dụ như button (nút bấm), check box hay scroll bar (thanh cuộn).

Ngoài những widget cơ bản lập trình viên còn có thể tùy chỉnh widget của riêng mình.



WIDGET

- **Checkbutton**: là widget hiển thị hộp đánh dấu.

Ví dụ:

```
from tkinter import Tk, Frame, Checkbutton  
from tkinter import BooleanVar, BOTH  
  
class Example(Frame):  
  
    def __init__(self, parent):  
        Frame.__init__(self, parent)  
        self.parent = parent  
  
        self.initUI()
```



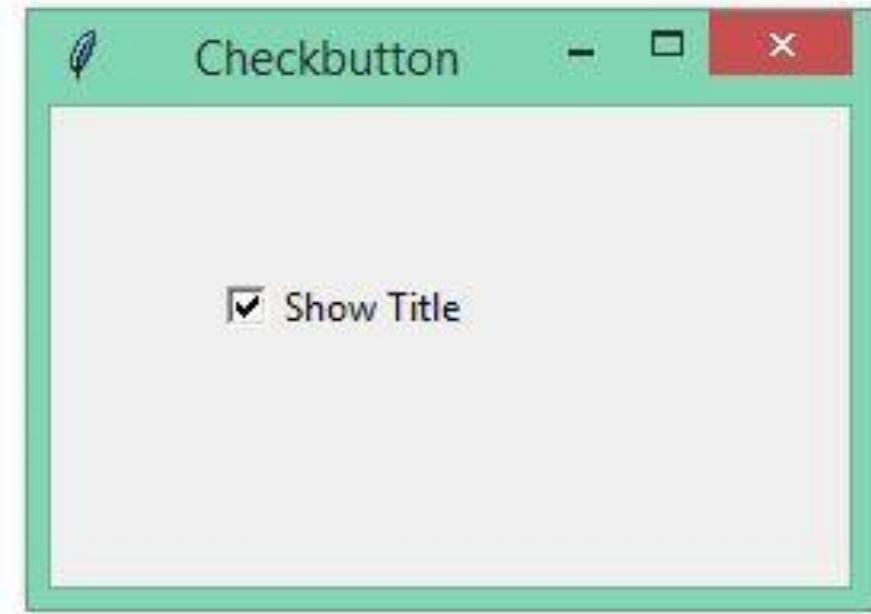
WIDGET

```
def initUI(self):  
    self.parent.title("Checkbutton")  
    self.pack(fill=BOTH, expand=True)  
    self.var = BooleanVar()  
    cb = Checkbutton(self, text="Show Title", variable=self.var,  
command=self.onClick)  
  
    cb.select()  
    cb.place(x=50, y=50)
```



WIDGET

```
def onClick(self):  
    if self.var.get() == True:  
        self.master.title("Checkbutton")  
    else:  
        self.master.title("")  
  
root = Tk()  
  
root.geometry("250x150+300+300")  
  
app = Example(root)  
  
root.mainloop()
```





WIDGET

- **Label:** dùng để hiển thị text hoặc hình ảnh.

Ví dụ: dùng Label để hiển thị ảnh lên màn hình.

```
from PIL import Image, ImageTk  
from tkinter import Tk, Frame, Label  
  
class Example(Frame):  
  
    def __init__(self, parent):  
        Frame.__init__(self, parent)  
        self.parent = parent  
        self.initUI()  
  
    def initUI(self):  
        img = ImageTk.PhotoImage(Image.open("img1.jpg"))  
        label = Label(self, image=img).pack()  
        self.parent.title("Image Example")  
        self.parent.geometry("300x200")  
        self.parent.mainloop()
```



WIDGET

```
def initUI(self):  
    self.parent.title("Label")  
    self.img = Image.open("C:\\tatras.jpg")  
    tatas = ImageTk.PhotoImage(self.img)  
    label = Label(self, image=tatas)  
    label.image = tatas  
  
    label.pack()  
    self.pack()
```

WIDGET

```
def setGeometry(self):  
    w, h = self.img.size  
    self.parent.geometry("%dx%d+300+300") % (w, h)  
  
root = Tk()  
ex = Example(root)  
ex.setGeometry()  
root.mainloop()
```





WIDGET

- **Scale:** hiển thị một thanh cuộn gắn với một khoảng giá trị nào đó.

Ví dụ:

```
from tkinter import Tk, BOTH, IntVar, LEFT  
from tkinter.ttk import Frame, Label, Scale, Style  
class Example(Frame):  
    def __init__(self, parent):  
        Frame.__init__(self, parent)  
        self.parent = parent  
        self.initUI()
```



WIDGET

```
def initUI(self):  
    self.parent.title("Scale")  
    self.style = Style()  
    self.style.theme_use("default")  
    self.pack(fill=BOTH, expand=1)  
    scale = Scale(self, from_=0, to=100, command=self.onScale)  
    scale.pack(side=LEFT, padx=15)  
    self.var = IntVar()  
    self.label = Label(self, text=0, textvariable=self.var)  
    self.label.pack(side=LEFT)
```



WIDGET

```
def onScale(self, val):  
    v = int(float(val))  
    self.var.set(v)  
  
root = Tk()  
ex = Example(root)  
root.geometry("250x100+300+300")  
root.mainloop()
```



WIDGET



- **Listbox:** cho phép hiển thị một danh sách các item. Người dùng có thể chọn một hoặc nhiều item.

```
from tkinter.ttk import Frame, Label  
from tkinter import Tk, BOTH, Listbox, StringVar, END  
  
class Example(Frame):  
    def __init__(self, parent):  
        Frame.__init__(self, parent)  
        self.parent = parent  
        self.initUI()  
  
    def initUI(self):
```



WIDGET

```
def initUI(self):  
    self.parent.title("Listbox")  
    self.pack(fill=BOTH, expand=1)  
  
    acts = ["Scarlet Johansson", "Rachel Weiss", "Natalie Portman",  
"Jessica Alba"]  
  
    lb = Listbox(self)
```



WIDGET

```
for i in acts:  
    lb.insert(END, i)  
  
lb.bind("<<ListboxSelect>>", self.onSelect)  
  
lb.pack(pady=15)  
  
self.var = StringVar()  
self.label = Label(self, text=0, textvariable=self.var)  
self.label.pack()
```

WIDGET

```
def onSelect(self, val):
    sender = val.widget
    idx = sender.curselection()
    value = sender.get(idx)
    self.var.set(value)

root = Tk()
ex = Example(root)
root.geometry("300x250+300+300")
root.mainloop()
```





THẢO LUẬN NHÓM

NỘI DUNG:

Cách tổ chức giao diện người dùng trong một số hệ điều hành thông dụng.



BÀI TẬP

NỘI DUNG:

Tạo một chương trình GUI giới thiệu về thông tin cá nhân.

CHUẨN BỊ CHO BUỔI HỌC TIẾP THEO:

Đọc các tài liệu về nội dung mục 8.4; 8.5 và 8.6.

Chương 8:

LẬP TRÌNH GIAO DIỆN TRONG PYTHON



NỘI DUNG GIẢNG DAY:

8.1. Giới thiệu về lập trình giao diện và thư viện Tkinter

8.2. Quản lý Layout

8.3. Widget

8.4. Menu

8.5. Hộp thoại

8.6. Đồ họa

8.7. Bài tập



MENU

Trình bày tất cả các lệnh và chức năng của ứng dụng, có sẵn cho người dùng thông qua giao diện người dùng.

Menu trong GUI được trình bày với sự kết hợp của văn bản và ký hiệu để thể hiện các lựa chọn bằng chuột (hoặc ngón tay trên màn hình cảm ứng) trên một trong các biểu tượng hoặc văn bản, một hành động sẽ được bắt đầu.

Một hành động hoặc hoạt động (lựa chọn) có thể, là mở hoặc lưu tệp, hoặc thoát hoặc thoát khỏi ứng dụng, ...



MENU

- **Ví dụ:** cách tạo menu trong Tkinter.

```
from tkinter import Frame, Tk, Menu
```

```
class Example(Frame):  
    def __init__(self, parent):  
        Frame.__init__(self, parent)  
  
        self.parent = parent  
        self.initUI()
```



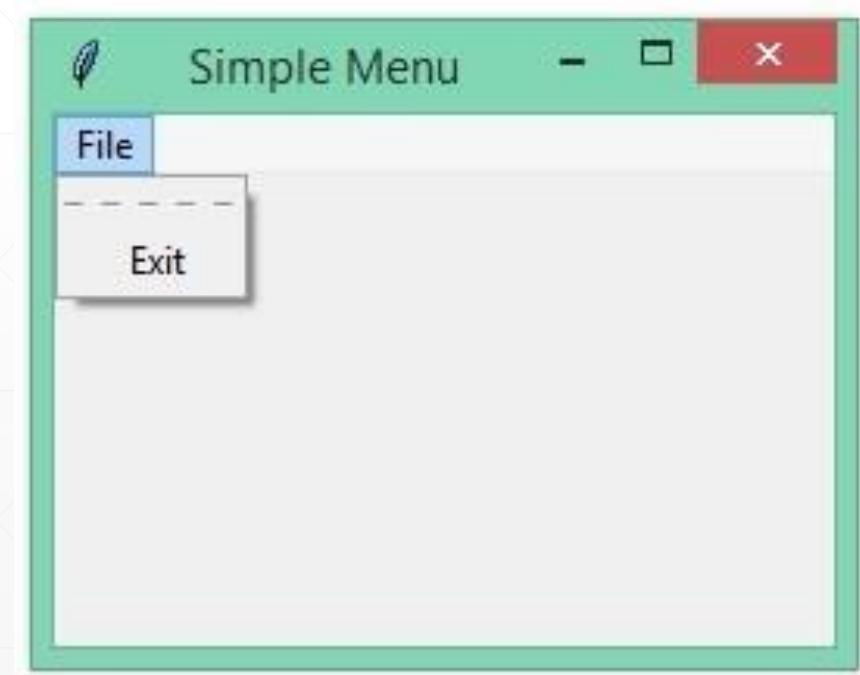
MENU

```
def initUI(self):  
    self.parent.title("Simple Menu")  
  
    menuBar = Menu(self.parent)  
    self.parent.config(menu=menuBar)  
  
    fileMenu = Menu(menuBar)  
    fileMenu.add_command(label="Exit", command=self.onExit)  
    menuBar.add_cascade(label="File", menu=fileMenu)
```



MENU

```
def onExit(self):  
    self.quit()  
  
root = Tk()  
root.geometry("250x150+300+300")  
app = Example(root)  
root.mainloop()
```





MENU

- **Tạo menu con:** tạo một menu con từ một menu cha.

```
from Tkinter import Tk, Frame, Menu
```

```
class Example(Frame):
```

```
    def __init__(self, parent):
```

```
        Frame.__init__(self, parent)
```

```
        self.parent = parent
```

```
        self.initUI()
```

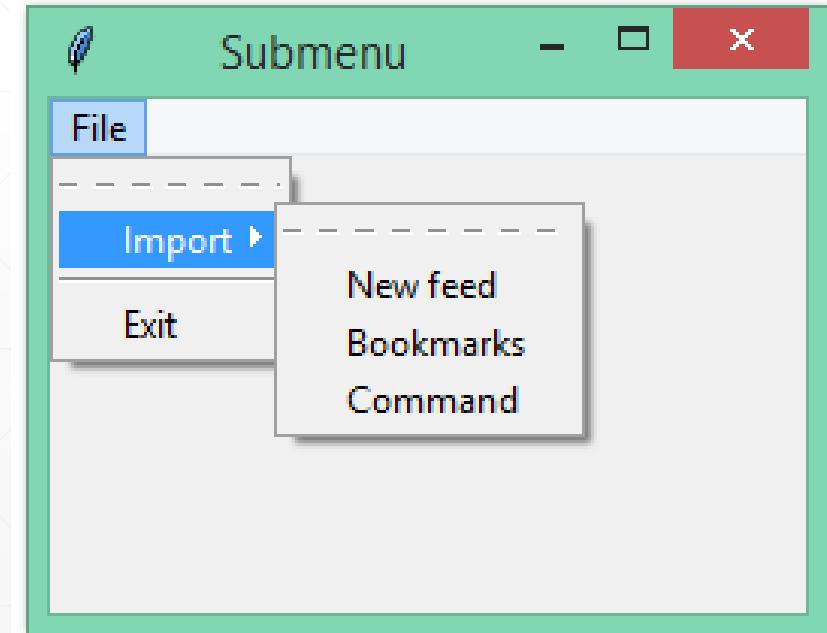


MENU

```
def initUI(self):  
    self.parent.title("Submenu")  
    menubar = Menu(self.parent)  
    self.parent.config(menu=menubar)  
    fileMenu = Menu(menubar)  
    submenu = Menu(fileMenu)  
    submenu.add_command(label="New feed")  
    submenu.add_command(label="Bookmarks")  
    submenu.add_command(label="Mail")  
    fileMenu.add_cascade(label='Import', menu=submenu)
```

MENU

```
fileMenu.add_separator()  
  
    fileMenu.add_command(label="Exit", command=self.onExit)  
  
menubar.add_cascade(label="File", menu=fileMenu)  
  
def onExit(self):  
    self.quit()  
  
root = Tk()  
root.geometry("250x150+300+300")  
app = Example(root)  
root.mainloop()
```





MENU

- **Popup menu:** còn được gọi là menu ngữ cảnh là menu được hiện ra khi click chuột lên cửa sổ.

```
from Tkinter import Tk, Frame, Menu  
  
class Example(Frame):  
  
    def __init__(self, parent):  
        Frame.__init__(self, parent)  
  
        self.parent = parent  
        self.initUI()
```



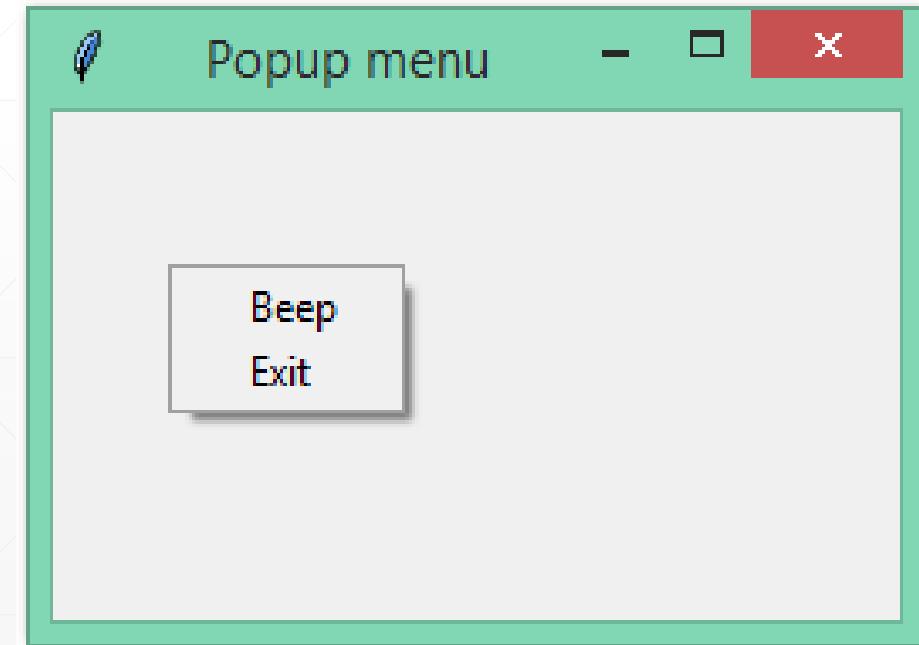
MENU

```
def initUI(self):  
    self.parent.title("Popup menu")  
    self.menu = Menu(self.parent, tearoff=0)  
    self.menu.add_command(label="Beep", command=self.bell())  
    self.menu.add_command(label="Exit", command=self.onExit)  
    self.parent.bind("<Button-3>", self.showMenu)  
    self.pack()  
  
def showMenu(self, e):  
    self.menu.post(e.x_root, e.y_root)
```



MENU

```
def onExit(self):  
    self.quit()  
  
root = Tk()  
root.geometry("250x150+300+300")  
app = Example(root)  
root.mainloop()
```



Chương 8:

LẬP TRÌNH GIAO DIỆN TRONG PYTHON



NỘI DUNG GIẢNG DAY:

- 8.1. Giới thiệu về lập trình giao diện và thư viện Tkinter
- 8.2. Quản lý Layout
- 8.3. Widget
- 8.4. Menu
- 8.5. Hộp thoại**
- 8.6. Đồ họa
- 8.7. Bài tập



HỘP THOẠI

- Có thể được sử dụng để hiển thị các hộp thông báo, hiển thị cảnh báo hoặc lỗi hoặc widget để chọn tệp và màu sắc.
- Ngoài ra còn có các hộp thoại đơn giản, yêu cầu người dùng nhập chuỗi, số nguyên hoặc số float.



HỘP THOẠI

- **Messagebox:** dùng để hiển thị thông báo cho người dùng và đôi khi còn dùng để đưa ra yêu cầu chọn lựa cho người dùng.

```
from tkinter.ttk import Frame, Button  
from tkinter import Tk, BOTH  
import tkinter.messagebox as mbox  
class Example(Frame):  
    def __init__(self, parent):  
        Frame.__init__(self, parent)  
        self.parent = parent  
        self.initUI()
```



HỘP THOẠI

```
def initUI(self):
    self.parent.title("Message Boxes")
    self.pack()
    error = Button(self, text="Error", command=self.onError)
    error.grid(padx=5, pady=5)
    warning = Button(self, text="Warning", command=self.onWarn)
    warning.grid(row=1, column=0)
    question = Button(self, text="Question", command=self.onQuest)
    question.grid(row=0, column=1)
    inform = Button(self, text="Information", command=self.onInfo)
    inform.grid(row=1, column=1)
```



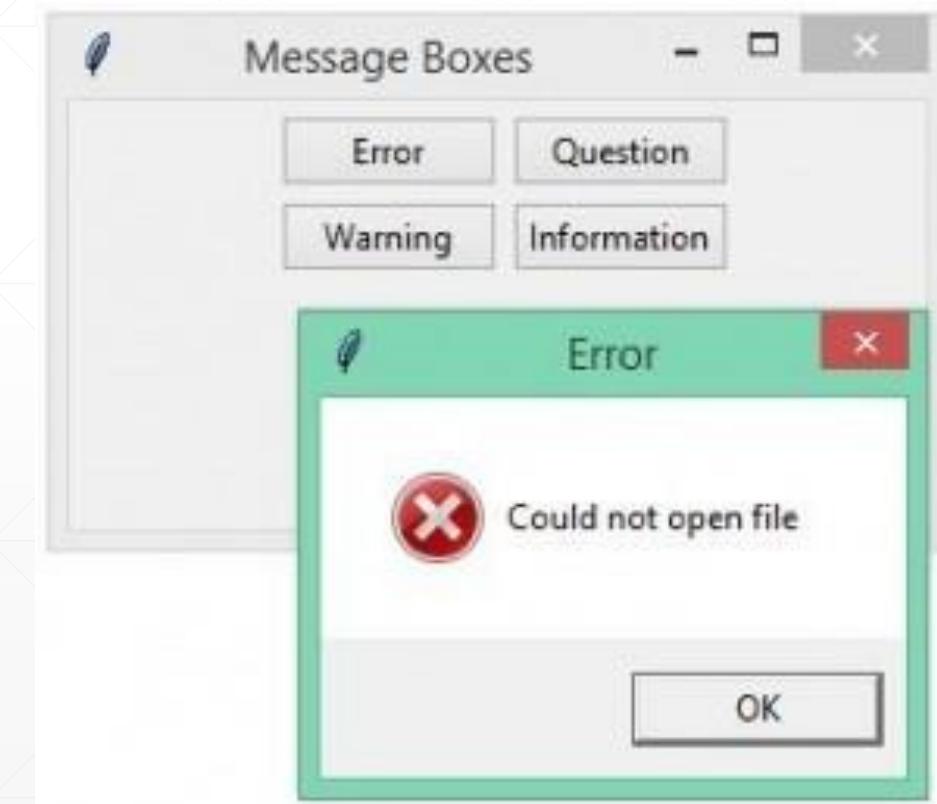
HỘP THOẠI

```
def onError(self):  
    mbox.showerror("Error", "Could not open file")  
def onWarn(self):  
    mbox.showwarning("Warning", "Deprecated function call")  
  
def onQuest(self):  
    mbox.askquestion("Question", "Are you sure to quit?")  
  
def onInfo(self):  
    mbox.showinfo("Information", "Download completed")
```



HỘP THOẠI

```
root = Tk()  
ex = Example(root)  
root.geometry("300x150+300+300")  
root.mainloop()
```





HỘP THOẠI

- **Hộp thoại chọn màu (Color chooser):** Các hệ điều hành hay có sẵn hộp thoại chọn màu cho chúng ta sử dụng.

```
from tkinter import Tk, Frame, Button, BOTH, SUNKEN  
from tkinter.colorchooser import askcolor
```

```
class Example(Frame):  
    def __init__(self, parent):  
        Frame.__init__(self, parent)  
        self.parent = parent  
        self.initUI()
```



HỘP THOẠI

```
def initUI(self):
    self.parent.title("Color chooser")
    self.pack(fill=BOTH, expand=1)
    self.btn      =      Button(self,           text="Choose      Color",
command=self.onChoose)
    self.btn.place(x=30, y=30)

    self.frame = Frame(self, border=1, relief=SUNKEN, width=100,
height=100)
    self.frame.place(x=160, y=30)
```

HỘP THOẠI



```
def onChoose(self):  
    (rgb, hx) = askcolor()  
    self.frame.config(bg=hx)
```

```
root = Tk()  
ex = Example(root)  
root.geometry("300x150+300+300")  
root.mainloop()
```





HỘP THOẠI

- **Hộp thoại chọn file (FileDialog):** Ví dụ: dùng hàm Open của module tkinter.filedialog để mở một File Dialog.

```
from tkinter import Frame, Tk, BOTH, Text, Menu, END  
from tkinter.filedialog import Open
```

```
class Example(Frame):  
    def __init__(self, parent):  
        Frame.__init__(self, parent)  
        self.parent = parent  
        self.initUI()
```



HỘP THOẠI

```
def initUI(self):
    self.parent.title("File dialog")
    self.pack(fill=BOTH, expand=1)
    menubar = Menu(self.parent)
    self.parent.config(menu=menubar)
    fileMenu = Menu(menubar)
    fileMenu.add_command(label="Open", command=self.onOpen)
    menubar.add_cascade(label="File", menu=fileMenu)
    self.txt = Text(self)
    self.txt.pack(fill=BOTH, expand=1)
```

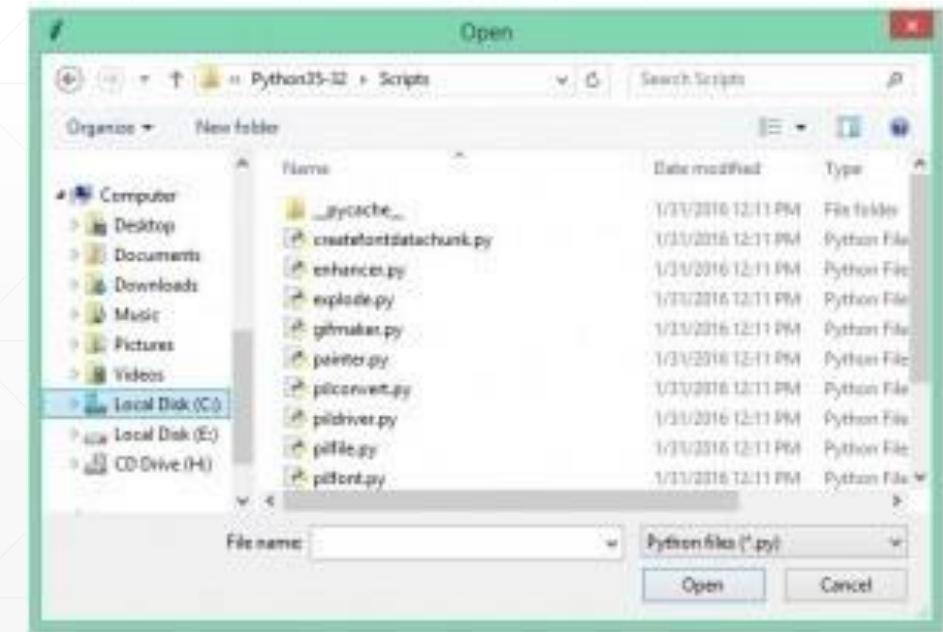


HỘP THOẠI

```
def onOpen(self):
    ftypes = [('Python files', '*.py'), ('All files', '*')]
    dlg = Open(self, filetypes = ftypes)
    fl = dlg.show()
    if fl != ":":
        text = self.readFile(fl)
        self.txt.insert(END, text)
def readFile(self, filename):
    f = open(filename, "r")
    text = f.read()
    return text
```

HỘP THOẠI

```
root = Tk()  
ex = Example(root)  
root.geometry("300x250+300+300")  
root.mainloop()
```



Chương 8:

LẬP TRÌNH GIAO DIỆN TRONG PYTHON



NỘI DUNG GIẢNG DAY:

- 8.1. Giới thiệu về lập trình giao diện và thư viện Tkinter
- 8.2. Quản lý Layout
- 8.3. Widget
- 8.4. Menu
- 8.5. Hộp thoại
- 8.6. Đồ họa**
- 8.7. Bài tập



ĐỒ HỌA

- **Vẽ đoạn thẳng:**

Để vẽ đoạn thẳng thì chúng ta dùng phương thức `create_line()` của lớp `Canvas`.

```
from tkinter import Tk, Canvas, Frame, BOTH
```

```
class Example(Frame):  
    def __init__(self, parent):  
        Frame.__init__(self, parent)  
        self.parent = parent  
        self.initUI()
```



ĐỒ HỌA

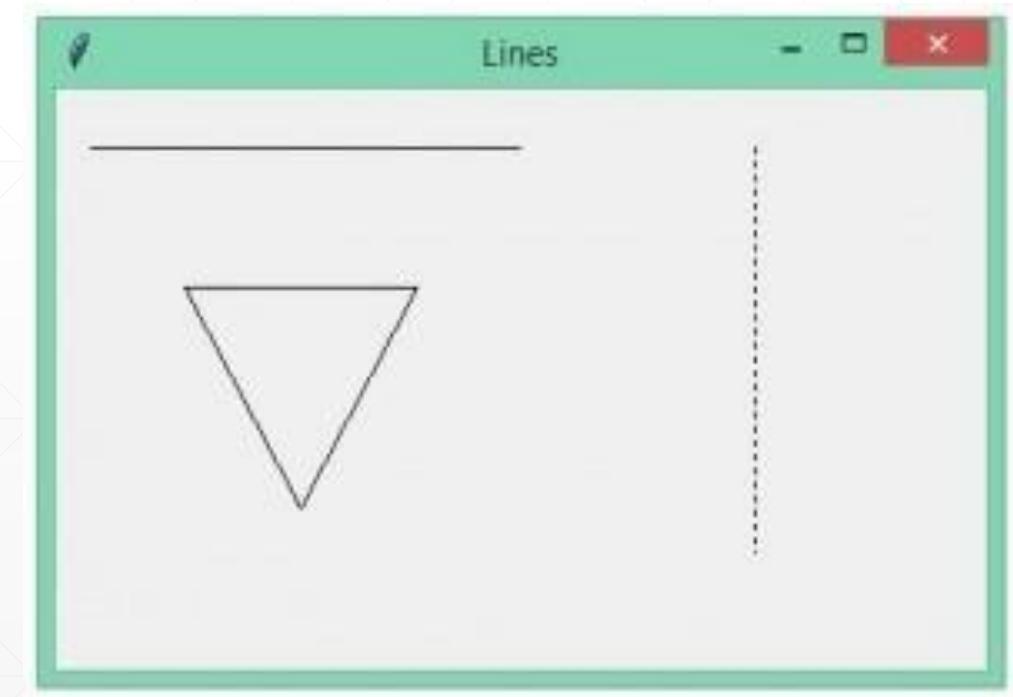
```
def initUI(self):
    self.parent.title("Lines")
    self.pack(fill=BOTH, expand=1)

    canvas = Canvas(self)
    canvas.create_line(15, 25, 200, 25)
    canvas.create_line(300, 25, 300, 200, dash=(4, 2))
    canvas.create_line(55, 85, 155, 85, 105, 180, 55, 85)

    canvas.pack(fill=BOTH, expand=1)
```

ĐỒ HỌA

```
root = Tk()  
ex = Example(root)  
root.geometry("400x250+300+300")  
root.mainloop()
```





ĐỒ HỌA

- **Vẽ màu:**

Màu trong máy tính là màu RGB, là tổ hợp của 3 giá trị đỏ (Red), xanh lá (Green) và xanh lam (Blue).

```
from tkinter import Tk, Canvas, Frame, BOTH
```

```
class Example(Frame):  
    def __init__(self, parent):  
        Frame.__init__(self, parent)  
        self.parent = parent  
        self.initUI()
```

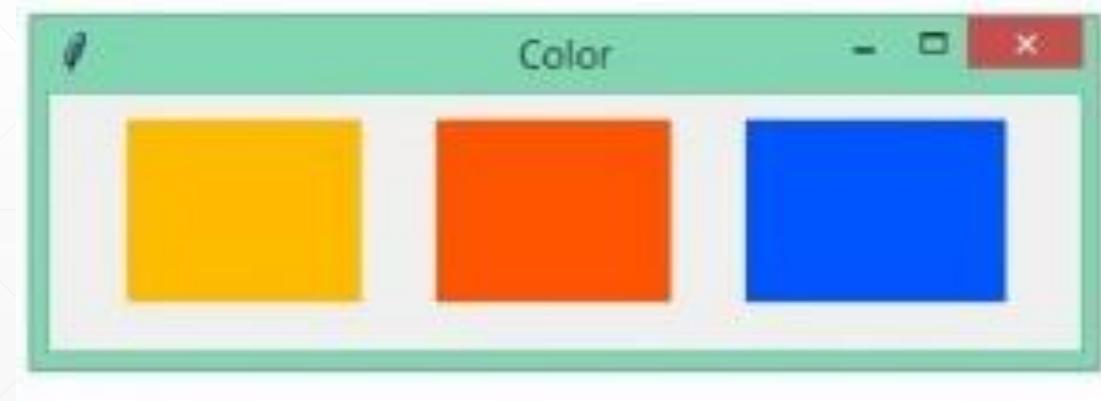
ĐỒ HỌA



```
def initUI(self):
    self.parent.title("Colors")
    self.pack(fill=BOTH, expand=1)
    canvas = Canvas(self)
    canvas.create_rectangle(30, 10, 120, 80, outline="#fb0",
fill="#fb0")
    canvas.create_rectangle(150, 10, 240, 80, outline="#f50",
fill="#f50")
    canvas.create_rectangle(270, 10, 370, 80, outline="#05f",
fill="#05f")
    canvas.pack(fill=BOTH, expand=1)
```

ĐỒ HỌA

```
root = Tk()  
ex = Example(root)  
root.geometry("400x100+300+300")  
root.mainloop()
```





ĐỒ HỌA

- **Vẽ một số đối tượng hình học khác:**

```
from tkinter import Tk, Canvas, Frame, BOTH
```

```
class Example(Frame):
```

```
    def __init__(self, parent):
```

```
        Frame.__init__(self, parent)
```

```
        self.parent = parent
```

```
        self.initUI()
```



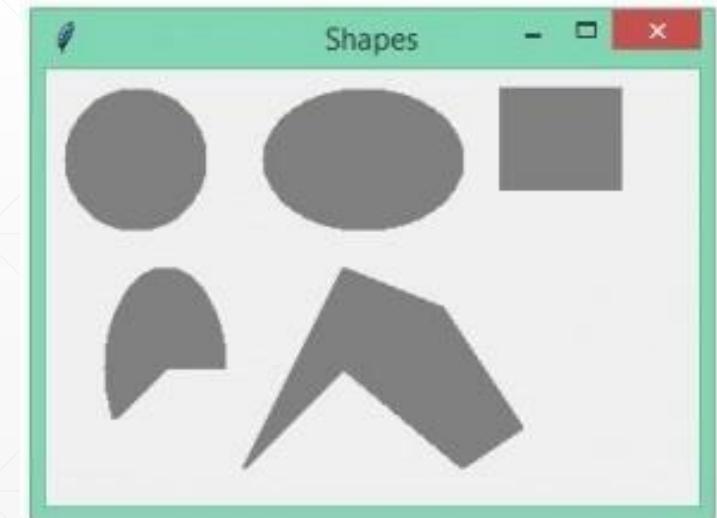
ĐỒ HỌA

```
def initUI(self):
    self.parent.title("Shapes")
    self.pack(fill=BOTH, expand=1)
    canvas = Canvas(self)
    canvas.create_oval(10, 10, 80, 80, outline="gray", fill="gray",
width=2)
    canvas.create_oval(110, 10, 210, 80, outline="gray",
fill="gray", width=2)
    canvas.create_rectangle(230, 10, 290, 60, outline="gray",
fill="gray", width=2)
```



ĐỒ HỌA

```
canvas.create_arc(30, 200, 90, 100, start=0, extent=210,  
outline="gray", fill="gray", width=2)  
    points = [150, 100, 200, 120, 240, 180, 210, 200, 150, 150,  
100, 200]  
        canvas.create_polygon(points, outline="gray", fill="gray",  
width=2)  
            canvas.pack(fill=BOTH, expand=1)  
root = Tk()  
ex = Example(root)  
root.geometry("330x220+300+300")  
root.mainloop()
```





BÀI TẬP

NỘI DUNG:

1. Tạo một chương trình GUI thực hiện chức năng của một máy tính (Calculator) đơn giản.
2. Tạo một chương trình GUI thực hiện giải các phương trình bậc nhất và bậc 2 với các hệ số do người dùng nhập vào từ bàn phím. Lưu kết quả giải phương trình vào file.



TỔNG KẾT HỌC PHẦN

Sau khi học xong học phần: **Kỹ thuật lập trình** các sinh viên cần nắm được các kiến thức và kỹ năng như sau:

- Kiến thức và kỹ năng căn bản về lập trình bao gồm hai phương pháp lập trình: lập trình có cấu trúc và lập trình hướng đối tượng.
- Xây dựng các thuật toán và thực hiện lập trình bằng ngôn ngữ Python để giải quyết vấn đề.
- Thể hiện phong cách lập trình chuyên nghiệp
- Kỹ năng về làm việc nhóm để viết chương trình phần mềm giải quyết các vấn đề trong thực tiễn.
- Sử dụng công cụ GITHUB để lưu trữ mã chương trình và làm việc nhóm.



TRƯỜNG ĐẠI HỌC VINH
VIỆN KỸ THUẬT VÀ CÔNG NGHỆ

CHÚC CÁC BẠN THÀNH CÔNG!

