

Sequential Programming Plan

Topic 03 Key concept

1. Sequential Programming

1.1 Sequential Structure



- Sequential refers to the process of executing commands created by coding in order.
- Although programs appear to perform tasks simultaneously due to their high speed, computers actually process commands in the order specified in the code.
- Naturally, even if the commands are the same, completely different results will occur if the order is changed.

1. Sequential Programming

1.1 Sequential Structure

- Sequential refers to the process of executing commands created by coding in order.



Therefore, when learning and coding Python in the future, it is important to determine the order based on the importance and purpose of the commands to create a program.

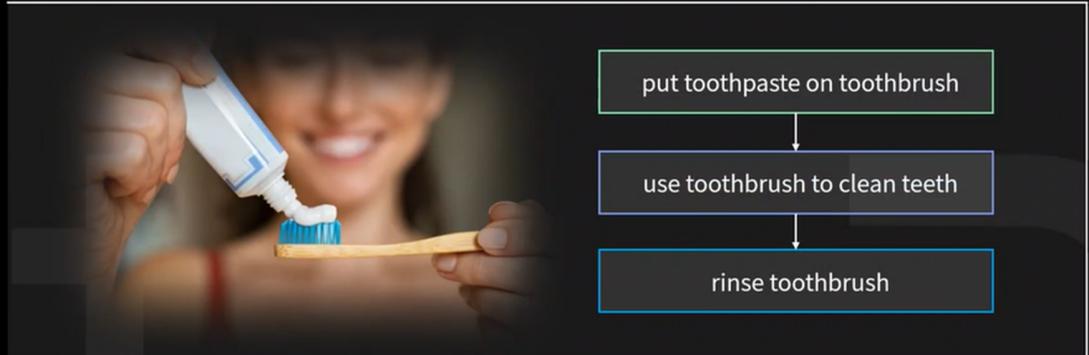
Prinme
AnB

-->2<--

1. Sequential Programming

1.1 Sequential Structure

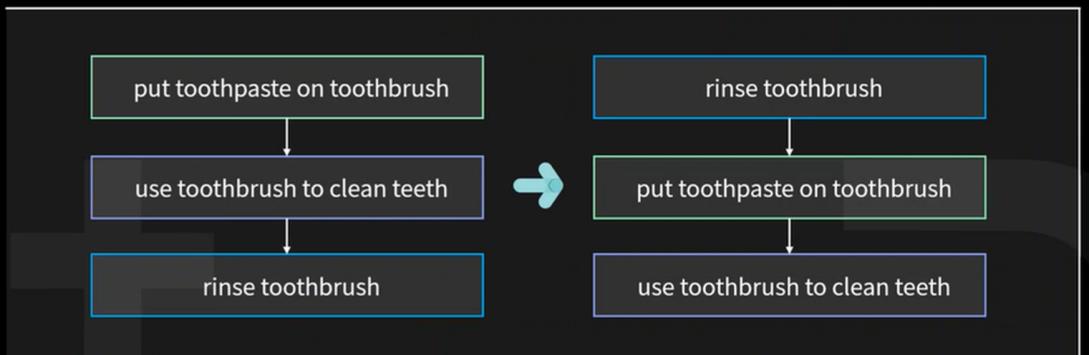
There are many cases in daily life where order is important.



1. Sequential Programming

1.1 Sequential Structure

What would happen if the order of the same command was changed?



1. Sequential Programming

1.1 Sequential Structure

What would happen if the order of the same command was changed?



Similarly, if the order of commands changes, the results will be different and it will be difficult to achieve the desired outcome.

Prinme
AnB

-->3<--

1. Sequential Programming

1.2 Pseudocode



- Pseudocode is the process of writing commands for the computer to execute in order.
- It is a good way to start coding with limited knowledge of programming because it uses everyday language.
- It can also serve as a bridge when completing flowcharts and transitioning to actual coding.

1. Sequential Programming

1.2 Pseudocode



- It can be used as a basic guide document among programmers during the implementation phase of the actual code.
- The main goal is to clearly describe and plan what needs to be done for each line of code.

1. Sequential Programming

1.2 Pseudocode

This program will allow the user to check the number whether it's even or odd

```
If "4"  
    print result  
    "This number is even"
```

```
If "3"  
    print result  
    "This number is odd"
```

- Start with a sentence that explains the purpose of the implementation.
- Write concisely.
- Use simple and distinguishable names, following rules to create them.
- Indentation and spacing are crucial.
- Express sequential structure, conditional structure, and repetitive structure (using if, for, while).

Prisme

AmB

— → / < —

1. Sequential Programming

1.3 Flowchart



- A flowchart is a representation of commands using standardized shapes and arrows to depict the flow of operations.
- The shapes and symbols used adhere to the standards set by the International Organization for Standardization (ISO).
- The flowchart should be written without overlapping, with the direction from top to bottom and left to right.

1. Sequential Programming

1.3 Flowchart

Start , End	Preparation	Action or Process	Decision
Indicates the start and end of a flowchart.	Initializing data.	A single calculation or operation.	Comparing and judging conditions, branching the flow based on the results.
Input and output of data.	Indicates what will be printed as a document.	Indicates the connection to another flowchart with this symbol.	Indicates the flow of processing.

1. Sequential Programming

1.4 Three Structures Represented by Flowcharts for Program Execution

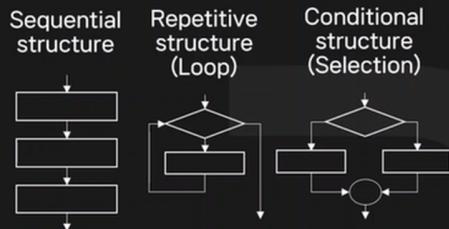
- Flowcharts represent **sequential**, **conditional**, and **repetitive** structures. These three structures can be applied not only in coding but also in real life.

In daily life—

- ➔ Go straight west at the intersection.
- ➔ If you see a post office, turn right at the post office.
- ➔ Continue straight.
- ➔ If you see an Italian restaurant, the third building is the accommodation you are looking for.

Application
→

Three structures of a program—



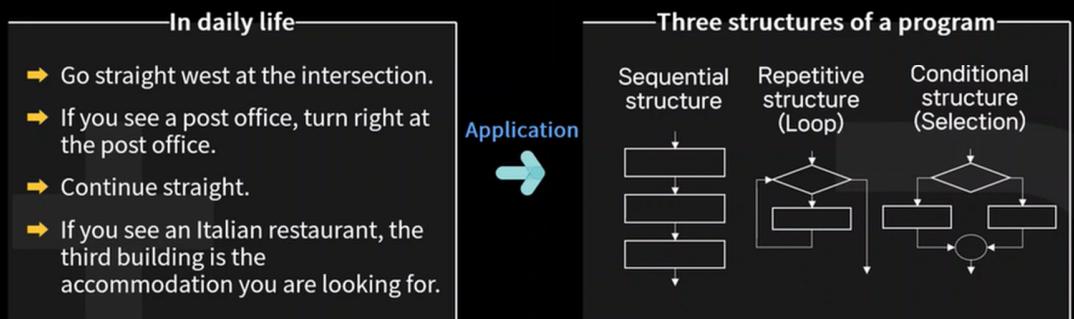
Prinme
AmB

-->5<--

1. Sequential Programming

1.4 Three Structures Represented by Flowcharts for Program Execution

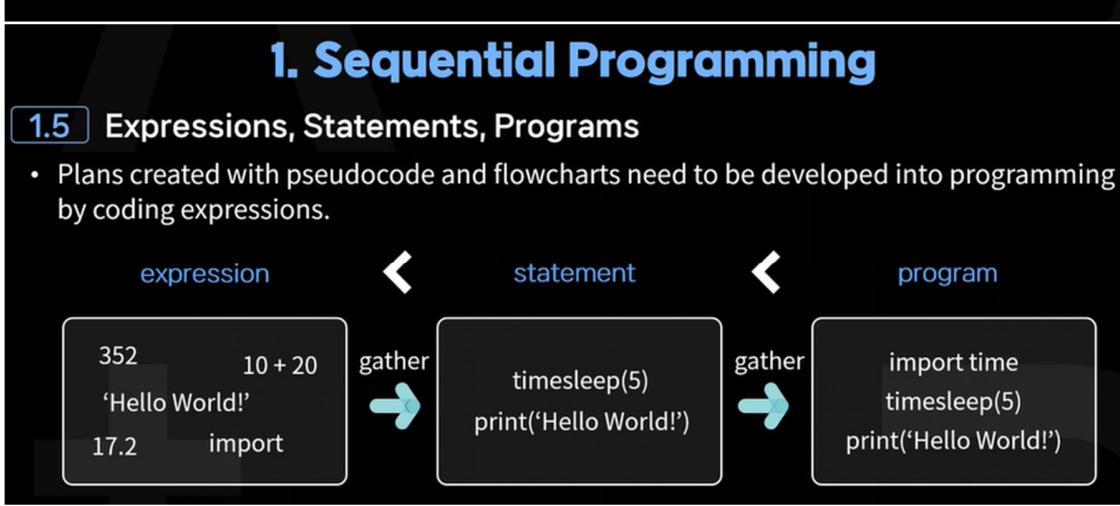
- First, they need to be written in sequential structure. Then, some parts of the order can be developed into repetitive or conditional structures.



1. Sequential Programming

1.5 Expressions, Statements, Programs

- Plans created with pseudocode and flowcharts need to be developed into programming by coding expressions.



1. Sequential Programming

1.5 Expressions, Statements, Programs

- An expression is a simple code that produces a value, such as numbers, formulas, and strings. A collection of expressions is called a statement. Multiple statements form a program.



Prinme
AmB

-->6<--

2. Setting up the Practical Environment

2.1 Installing Anaconda

- We will set up the development environment using Anaconda for coding.

The screenshot shows the Anaconda Distribution page. At the top, there's a navigation bar with links for Enterprise, Pricing, Solutions, Resources, About, and Contact Sales. Below the navigation, the text "Anaconda Distribution" and "Free Download" is prominently displayed. A subtext says "Everything you need to get started in data science on your workstation." There's a checked checkbox for "Free distribution install" and a link "https://www.anaconda.com/products/individual".

2. Setting up the Practical Environment

2.1 Installing Anaconda

The first screenshot shows the "Choose Install Location" step, where the user can select the destination folder for the Anaconda installation. The second screenshot shows the "Advanced Installation Options" step, specifically highlighting the checkbox for "Add Anaconda3 to my PATH environment variable". A note in the background of this window states: "Not recommended. Instead, open Anaconda3 with the Windows Start menu and select 'Anaconda (64-bit)'. This 'Add to PATH' option makes Anaconda3 available from anywhere on your system, which may cause problems requiring you to uninstall and reinstall Anaconda." The third screenshot is a separate window for "Anaconda3 2021.05 (64-bit)" advertising PyCharm.

- If you have previously installed a separate Python program, you need to uncheck "Add Anaconda3 to my PATH environment variable" in Advanced Options because it will conflict with python.exe in Anaconda.

2. Setting up the Practical Environment

2.2 Creating Virtual Environments

- When programming with Python, you often use **external packages**.
- When managing multiple projects that use external packages or starting a new project, you may need to use the **latest version of Python** or **different types/versions of packages**.
- In such cases, it becomes very difficult to **manage modules and versions**, and problems can occur if they are not compatible between versions.

Prinme
AnB

-->7<--

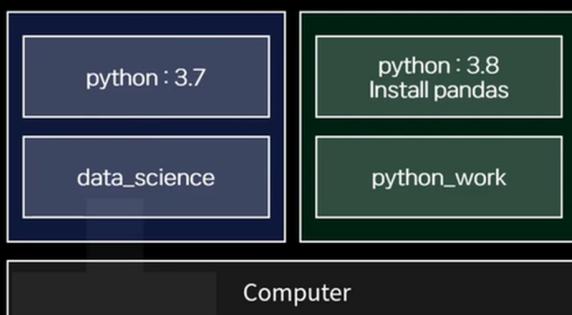
2. Setting up the Practical Environment

2.2 Creating Virtual Environments

- For this reason, we will learn how to create virtual environments.
- A Python **virtual environment** provides an independent space where you can manage modules and versions for each project. It allows you to build various virtual environments on a single server/computer.

2. Setting up the Practical Environment

2.2 Creating Virtual Environments



- For example, you can work with python 3.7 in the data_science virtual environment and work with python 3.8 and pandas in the python_work virtual environment, which gives you the effect of using two computers.

2. Setting up the Practical Environment

2.2 Creating Virtual Environments

- To create a Python virtual environment, run **Terminal on macOS or Anaconda Prompt on Windows**.
- In the Terminal or Anaconda Prompt window, enter the command "conda create -n python_work python=3.8" to create a virtual environment compatible with Python 3.8. 'python_work' is the name of the virtual workspace and can be defined with a different name.

A screenshot of an Anaconda Prompt window titled 'Anaconda Prompt'. The command line shows the user running the command 'conda create -n python_work python=3.8'. The command is highlighted with a red rectangle.

Prinme
AmB

-->8<--

2. Setting up the Practical Environment

2.2 Creating Virtual Environments

- When the prompt displays "Proceed ([y]/n)?", enter 'y' to continue the process.

Anaconda Prompt window showing a list of packages and a yellow box around the command "Proceed ([y]/n)? y".

```
ca-certificates      pkgs/main/win-64::ca-certificates-2021.7.5-haa95532_1
certifi               pkgs/main/win-64::certifi-2021.5.30-py38haa95532_0
openssl              pkgs/main/win-64::openssl-1.1.1k-h2bbff1b_0
pip                  pkgs/main/win-64::pip-21.0.1-py38haa95532_0
python               pkgs/main/win-64::python-3.8.11-h6244533_1
setuptools            pkgs/main/win-64::setuptools-52.0.0-py38haa95532_0
sqlite               pkgs/main/win-64::sqlite-3.36.0-h2bbff1b_0
vc                   pkgs/main/win-64::vc-14.2-h21ff451_1
vs2015_runtime        pkgs/main/win-64::vs2015_runtime-14.27.29016-h5e58377_2
wheel                pkgs/main/noarch::wheel-0.37.0-pyhd3eb1b0_0
wincertstore         pkgs/main/win-64::wincertstore-0.2-py38_0
```

Proceed ([y]/n)? y

2. Setting up the Practical Environment

2.2 Creating Virtual Environments

- Let's try entering the command "conda create -n data_science python=3.7" in the Terminal or Anaconda Prompt. This will create a new virtual workspace named 'data_science' compatible with Python 3.7.

Anaconda Prompt window showing the command "conda create -n data_science python=3.7" in the terminal.

```
(base) C:\Users\user>conda create -n data_science python=3.7
```

2. Setting up the Practical Environment

2.2 Creating Virtual Environments

- Again, when the prompt displays "Proceed ([y]/n)?", enter 'y' to continue the process.

Anaconda Prompt window showing a list of packages and a yellow box around the command "Proceed ([y]/n)? y".

```
ca-certificates      pkgs/main/win-64::ca-certificates-2021.7.5-haa95532_1
certifi               pkgs/main/win-64::certifi-2021.5.30-py38haa95532_0
openssl              pkgs/main/win-64::openssl-1.1.1k-h2bbff1b_0
pip                  pkgs/main/win-64::pip-21.0.1-py38haa95532_0
python               pkgs/main/win-64::python-3.8.11-h6244533_1
setuptools            pkgs/main/win-64::setuptools-52.0.0-py38haa95532_0
sqlite               pkgs/main/win-64::sqlite-3.36.0-h2bbff1b_0
vc                   pkgs/main/win-64::vc-14.2-h21ff451_1
vs2015_runtime        pkgs/main/win-64::vs2015_runtime-14.27.29016-h5e58377_2
wheel                pkgs/main/noarch::wheel-0.37.0-pyhd3eb1b0_0
wincertstore         pkgs/main/win-64::wincertstore-0.2-py38_0
```

Proceed ([y]/n)? y

Prinme
AmB

-->9<--

2. Setting up the Practical Environment

2.3 Anaconda Commands for Managing Virtual Environments

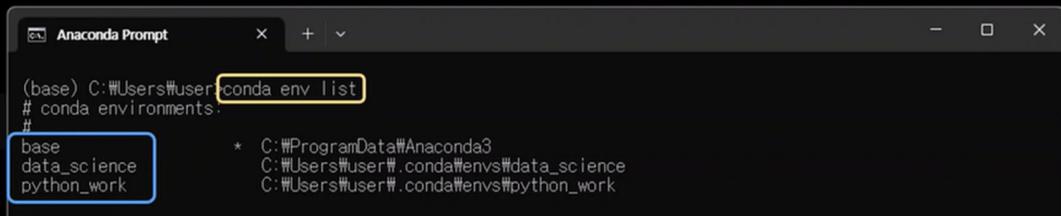
- You can perform various operations related to virtual environments by entering the following virtual environment commands in Terminal or Anaconda Prompt.

Action	Command
List virtual environments	conda env list
Create a virtual environment	conda create -n [environment name]
Clone a virtual environment	conda create --clone [source environment] --n [new environment]
Activate a virtual environment	conda activate [environment name]
Deactivate the current virtual environment	conda deactivate
Remove a virtual environment	conda remove --name [environment name] --all

2. Setting up the Practical Environment

2.3 Anaconda Commands for Managing Virtual Environments

- Run "conda env list" command in Terminal or Anaconda Prompt to see the list of virtual environments in Python.
- The default value is 'base', and you should select the virtual environment you want to install. Previous virtual environments such as 'data_science' and 'python_work' should be listed.



```
(base) C:\Users\user>conda env list
# conda environments:
#
base                         * C:\ProgramData\Anaconda3
C:\Users\user\.conda\envs\data_science
C:\Users\user\.conda\envs\python_work
```

2. Setting up the Practical Environment

2.3 Anaconda Commands for Managing Virtual Environments

- Use the command "conda activate data_science" to switch to a new working environment. The prompt will change from 'base' to 'data_science' to indicate the changed working environment.



```
(base) C:\Users\user>conda activate data_science
(data_science) C:\Users\user>
```

Prinme

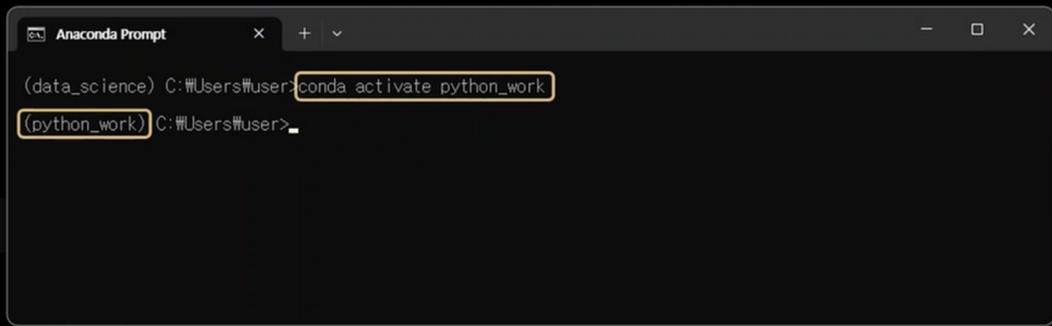
-->10<--

AnB

2. Setting up the Practical Environment

2.3 Anaconda Commands for Managing Virtual Environments

- This time, try switching to another working environment named 'python_work' by using the command "conda activate python_work".



The image shows a dark-themed Anaconda Prompt window. The title bar says 'Anaconda Prompt'. In the terminal area, the user is in the directory '(data_science) C:\Users\user'. They type the command 'conda activate python_work'. After pressing enter, the prompt changes to '(python_work) C:\Users\user>'.

2. Setting up the Practical Environment

2.4 Installing Jupyter Notebook



- Once you have completed the Anaconda installation, you can install Jupyter Notebook.

3. Writing Output Statements in Jupyter Notebook

3.1 Basic Usage of the print Statement

- Let's write a code to print "Hello World!" in Jupyter Notebook.
- The print() statement is used to display the value inside the parentheses.

```
print("Hello World!")
```

You can use either single or double quotation marks to define a string, but they should match at the beginning and the end.

The content between the opening and the closing quotation marks is called a **string**.

TIP

Unlike other languages, Python automatically adds a line break after printing a string with a single print statement.

3. Writing Output Statements in Jupyter Notebook

3.2 Printing Strings and Values

- To print a string using the print statement, enclose the string in quotation marks.

```
1 print('Hello World!')  
2 print("Hello World!")
```

```
Hello World!  
Hello World!
```

- To print a numeric value, simply enter the number without quotation marks.

```
1 print(10)  
2 print(7.5)
```

```
10  
7.5
```

- If you include quotation marks when printing a number, it will be treated as a string.

3. Writing Output Statements in Jupyter Notebook

3.2 Printing Strings and Values

- You can use a comma (,) within the print statement to print multiple values at once.

```
1 print('The radius of a circle', 4.0)
```

```
The radius of a circle 4.0
```

3. Writing Output Statements in Jupyter Notebook

3.3 Creating a Self-Introduction

- Let's create a self-introduction using the print statement.

```
1 print('Hello! I will introduce myself.')
2 print('Name: David Doe')
3 print('Age: 23')
4 print('Job: Data Scientist')
5 print('Address: Seoul, South Korea')
6 print('Place of Birth: Southern California, USA')
```

- You can decorate the self-introduction by repeating asterisks (*) or hyphens (-).

```
1 print('*****')
2 print('-----')
```

4. Basic Terminology for Learning

- When encountering new terms during the learning process, it's important to make a note of them and verify their exact definitions.
- It's also a good idea to refer to tutorials provided by the official Python documentation.

The screenshot shows the Python 3.11.4 Documentation page. At the top, there are links for "Download" and "Docs by version". The main content area features a large heading "Python 3.11.4 documentation". Below it, a message says "Welcome! This is the official documentation for Python 3.11.4." There are sections for "Parts of the documentation:", "What's new in Python 3.11?", "Tutorial", and "Library Reference". On the right, there are links for "Installing Python Modules" and "Distributing Python Modules". A footer at the bottom right includes a link to "https://docs.python.org/3/".

One Step Further

1. Introduction to Python Learning References

- The following learning references are useful for studying Python and can be referred to when you need help or want to learn more about Python.

The screenshot shows the Python.org homepage. The top navigation bar includes links for "Python", "PSF", "Docs", "PyPI", "Jobs", and "Community". The main content area features the Python logo and a search bar. Below the search bar, there is a navigation menu with links for "About", "Downloads", "Documentation", "Community", "Success Stories", "News", and "Events". A code snippet for generating a Fibonacci series is displayed on the left, and a section titled "Functions Defined" with explanatory text is on the right.

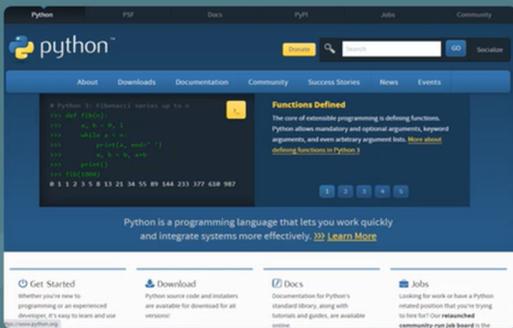
AmB → Prime

→ 13 < --

One Step Further

1. Introduction to Python Learning References

► Python Software Foundation



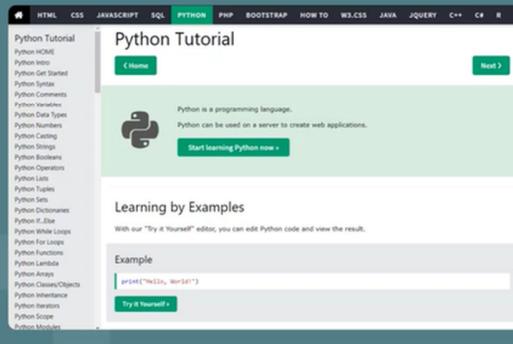
- A non-profit organization responsible for the development and distribution of Python.
- Provides free downloads of the latest Python version.
- Offers tutorials and documentation for learning Python.

<https://docs.python.org/3/>

One Step Further

1. Introduction to Python Learning References

► W3Schools



- To expand and apply Python further, where can you find the necessary resources?
- W3Schools offers tutorials on various programming languages, including Python.
- It provides the advantage of being able to run Python code examples directly in a web environment.

<https://www.w3schools.com>

AmB → Prime

→ 14 < --

One Step Further

1. Introduction to Python Learning References

- ▶ Stack Overflow

The screenshot shows a Stack Overflow question page. The title is "Reversing list element in python". The question was asked 1 month ago and has 55 views. It asks how to reverse list elements one by one in Python using a nested loop. A user named "2" provided an answer using a list comprehension. The code is as follows:

```
How to reverse list element one by one in python? I tried nested loop. But I couldn't display second loop as "Result".
-2
For example:
Input: [88,344,589]
Result: [285,443,798]
```

The answer has 2 upvotes and 1 comment. The Stack Overflow sidebar on the left shows various navigation links like Home, Questions, Tags, Users, COLLECTIVES, FIND A JOB, Jobs, Companies, Teams, and Stack Overflow for Teams. The right sidebar shows news from The Overflow Blog and Hot Meta Posts.

<https://stackoverflow.com/>

One Step Further

1. Introduction to Python Learning References

- ▶ Stack Overflow

This screenshot is identical to the one above, showing the same Stack Overflow question and interface. It features the same question title, timestamp, view count, and code snippet. The sidebar and footer are also the same.

<https://stackoverflow.com/>