

# 웹프로그래밍의 기초

Week3

파이썬 설치, 개념, 변수, 연산자

# Python Installation

# Python on Ubuntu

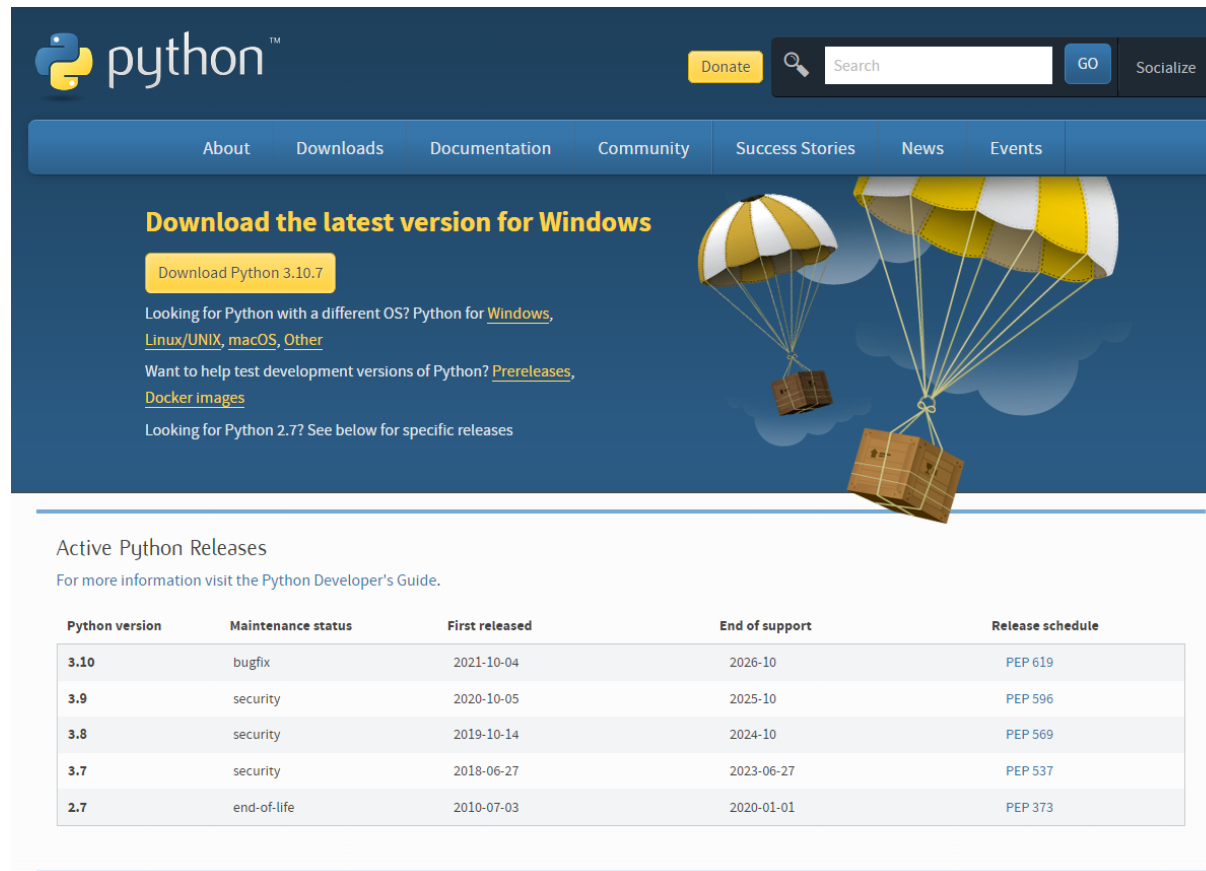
- From Ubuntu 18.04 LTS, Python 3 will be the only Python version installed by default.
- Default version of Python 3 can be upgraded by

```
$ sudo apt-get upgrade python3
```

```
testuser@ubuntu:~$ sudo apt-get upgrade python3
Reading package lists... Done
Building dependency tree
Reading state information... Done
python3 is already the newest version (3.8.2-0ubuntu2).
Calculating upgrade... Done
The following package was automatically installed and is no longer required:
  libfprint-2-tod1
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

# Python on other environments

- Can be downloaded via <https://www.python.org/downloads/>



The screenshot shows the Python.org website. At the top, there is a navigation bar with the Python logo, a search bar, and links for Donate, Socialize, and various sections: About, Downloads, Documentation, Community, Success Stories, News, and Events. The main content area features a large banner for downloading the latest version for Windows (Python 3.10.7). Below this, there are links for other operating systems (Linux/UNIX, macOS, Other), a link for prereleases, and a link for Docker images. A section titled 'Active Python Releases' provides a table of release information.

**Download the latest version for Windows**

Download Python 3.10.7

Looking for Python with a different OS? Python for [Windows](#), [Linux/UNIX](#), [macOS](#), [Other](#)

Want to help test development versions of Python? [Prereleases](#), [Docker images](#)

Looking for Python 2.7? See below for specific releases

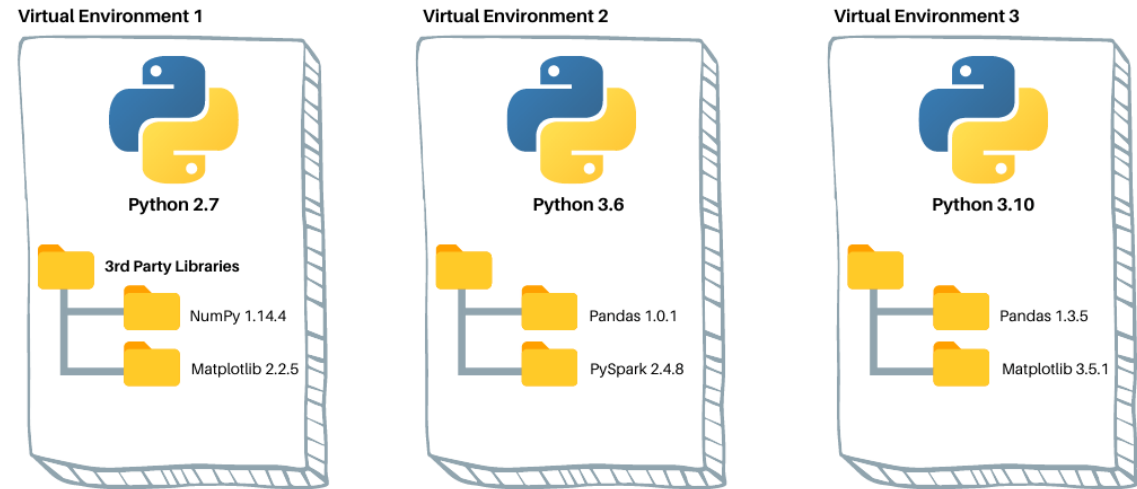
**Active Python Releases**

For more information visit the [Python Developer's Guide](#).

Python version	Maintenance status	First released	End of support	Release schedule
3.10	bugfix	2021-10-04	2026-10	PEP 619
3.9	security	2020-10-05	2025-10	PEP 596
3.8	security	2019-10-14	2024-10	PEP 569
3.7	security	2018-06-27	2023-06-27	PEP 537
2.7	end-of-life	2010-07-03	2020-01-01	PEP 373

# Python Virtual Environments

- `venv` / `pyenv`
  - If you are using a single version of Python say version 3.3+, and want to manage different virtual environments, then `venv` is all you need.
- `pyenv`
  - If you want to use multiple versions of Python at 3.3+, with or without virtual environments, then continue to read about `pyenv`.
- `pyenv-virtualenv`
  - If you also want to work with Python 2, then `pyenv-virtualenv` is a tool to consider.



# Anaconda

- allows to manage various combinations of Pythons and dependent packages

```
(base) 2022863_14080@Ubuntu-147:~/바탕화면$ python --version
Python 3.9.12
(base) 2022863_14080@Ubuntu-147:~/바탕화면$ conda create -n p37 python=3.7

(base) 2022863_14080@Ubuntu-147:~/바탕화면$ conda activate p37
(p37) 2022863_14080@Ubuntu-147:~/바탕화면$ python --version
Python 3.7.13
(p37) 2022863_14080@Ubuntu-147:~/바탕화면$
```

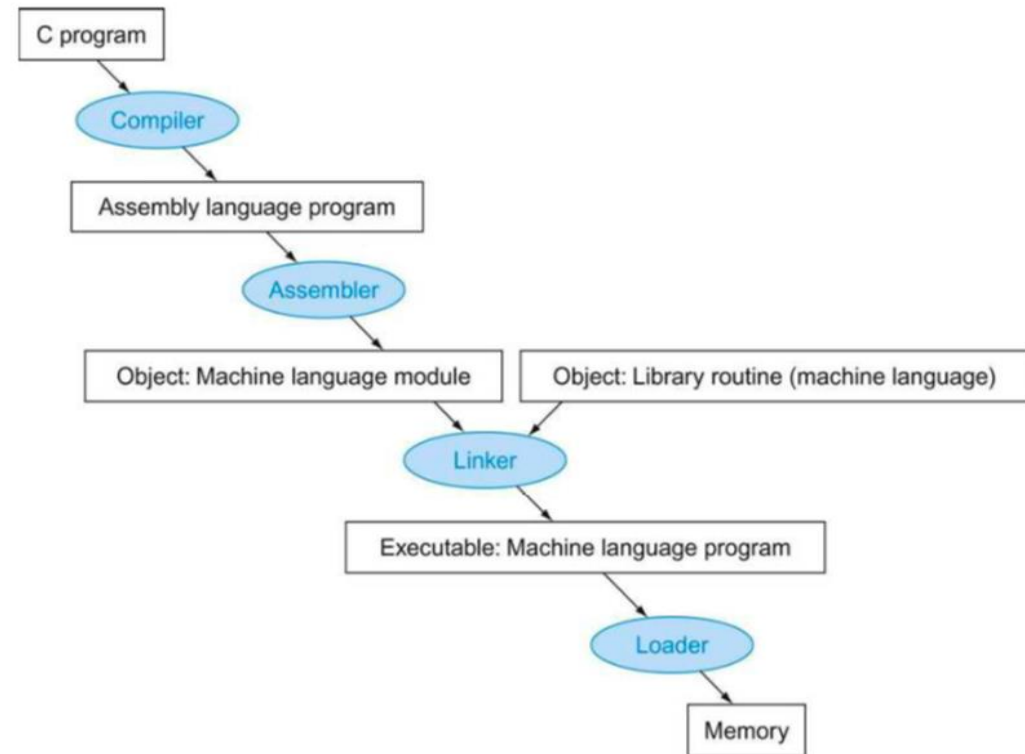
- allows to install a specific version of the package that you may want to use.

```
ex1)
conda install matplotlib=1.4.3
conda install 'matplotlib>=1.4.3'
ex2)
conda install openpyxl '>=2.4.10,<=2.6.0'
conda install "openpyxl>=2.4.10,<3.0.0"
```

# Python Interpreter

# Compiler

- **Compilation** :- Compilation is a process in which a program written in one language get translated into another targeted language. If there is some errors, the compiler will detect them and report it.
- A compiler produces a program written in assembly language.
- A compiled program is not human readable, but instead is in an architecture-specific machine language.





# Interpreter

- In an interpreted program, on the other hand, the source code typically is the program. Programs of this type (often known as scripts) require an interpreter, which parses the commands in the program and then executes them.
- Some interpreters, such as the Unix shells (sh, csh, ksh, etc.), read and then immediately execute each command, while others, such as Perl, analyze the entire script before sending the corresponding machine language instructions.
- The advantage of a script is that it is very portable. Any computer that has the appropriate interpreter installed may run the program more or less unchanged.
- This is a disadvantage as well, because the program will not run at all if the interpreter is not available. In general, interpreted programs are slower than compiled programs, but are easier to debug and revise. Other examples of interpreted languages include JavaScript and Python.

Variables:  
Strings and numbers

# Variables

- Variables are containers for storing data values.
- Variables are used for the logical flow of the program and not shown to the users.

```
print("Hello Python world!")
```

```
message = "Hello Python world!"  
print(message)
```



```
Hello Python world!
```

```
(p37) 2022863_14080@Ubuntu-147:~$ python  
Python 3.7.13 (default, Mar 29 2022, 02:18:16)  
[GCC 7.5.0] :: Anaconda, Inc. on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> print("Hello Python World!")  
Hello Python World!  
>>> message = "Hello Python world!"  
>>> print(message)  
Hello Python world!  
>>>
```

# Creating variables

- Python has no command for declaring a variable.
- A variable is created the moment you first assign a value to it.
- Variables do not need to be declared with any particular type, and can even change type after they have been set.

```
x = 5
y = "John"
print(x)
print(y)

5
John
```

```
x = 4          # x is of type int
x = "Sally"    # x is now of type str
print(x)

Sally
```

# Variable Names

- A variable can have a short name (like x and y) or a more descriptive name (age, carname, total\_volume).
- Follow Rules for Python variables:
  - A variable name must start with a letter or the underscore character
  - A variable name cannot start with a number
  - A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_)
  - Variable names are case-sensitive (age, Age and AGE are three different variables). To avoid unnecessary errors, Lowercases only policy is recommended.
- Or, you will see an syntax error.
- Most errors derived from your typo.

```
>>> my_message='Web Programming'
>>> print(my_mesage)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'my_mesage' is not defined
```

# String Concatenation

- print() function is often used to output variables. And the following 3 cases will return the same sentence of "Python is awesome."

```
x = "Python is awesome."  
print(x)
```

```
x = "Python"  
y = "is"  
z = "awesome."  
print(x, y, z)
```

```
x = "Python "  
y = "is "  
z = "awesome."  
print(x + y + z)
```

# String methods

- With escape characters

- Tab: `\t`
- NL: `\n`

```
mytext = "The list of famous machine learning  
frameworks:\n\tPytorch\n\tTensorflow"  
print(mytext)
```

```
The list of famous machine learning frameworks:  
    Pytorch  
    Tensorflow
```

- With space removed

```
left = "["  
right = "]"  
mytext = "  python  "  
print(left + mytext + right)  
print(left + mytext.lstrip() + right)  
print(left + mytext.rstrip() + right)  
print(left + mytext.strip() + right)
```

```
[  python  ]  
[python  ]  
[  python]  
[python]
```

# String methods (Cont'd)

```
orig_str = "Harry Potter and the Philosopher's Stone"  
print(orig_str.lower())  
print(orig_str.upper())  
print(orig_str.capitalize())  
print(orig_str.title())
```

```
harry potter and the philosopher's stone  
HARRY POTTER AND THE PHILOSOPHER'S STONE  
Harry potter and the philosopher's stone  
Harry Potter And The Philosopher'S Stone
```



# Numbers

- There are three numeric types in Python:
  - Int: a whole number, positive or negative, without decimals, of unlimited length.
  - Float: a number, positive or negative, containing one or more decimals.
  - Complex: Complex numbers are written with a "j" as the imaginary part
- Variables of numeric types are created when you assign a value to them

```
x = 1
y = 2.8
z = 1j
print(type(x))
print(type(y))
print(type(z))

<class 'int'>
<class 'float'>
<class 'complex'>
```

# Casting datatypes

- There may be times when you want to specify a type on to a variable. This can be done with casting.
  - `int()` - constructs an integer number from an integer literal, a float literal (by removing all decimals), or a string literal (providing the string represents a whole number)
  - `float()` - constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer)
  - `str()` - constructs a string from a wide variety of data types, including strings, integer literals and float literals

```
x = int(1)      # x will be 1
y = int(2.8)    # y will be 2
z = int("3")    # z will be 3
```

```
x = float(1)    # x will be 1.0
y = float(2.8)  # y will be 2.8
z = float("3")  # z will be 3.0
w = float("4.2") # w will be 4.2
```

```
x = str("s1")   # x will be 's1'
y = str(2)      # y will be '2'
z = str(3.0)    # z will be '3.0'
```

# Operators

# Python Arithmetic Operators

Operator	Name	Example
+	Addition	$x + y$
-	Subtraction	$x - y$
*	Multiplication	$x * y$
/	Division	$x / y$
%	Modulus	$x \% y$
**	Exponentiation	$x ** y$
//	Floor division	$x // y$

# Python Assignment Operators

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3

# Python Comparison Operators

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

# Python Logical Operators

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)