Introudction To Python

```
module = Module(
    code="ELEE1147",
    name="Programming for Engineers",
    credits=15,
    module_leader="Seb Blair BEng(H) PGCAP MIET MIHEEM FHEA"
)
```

UNIVERSITY OF GREENWICH

Download as a PDF

Overview

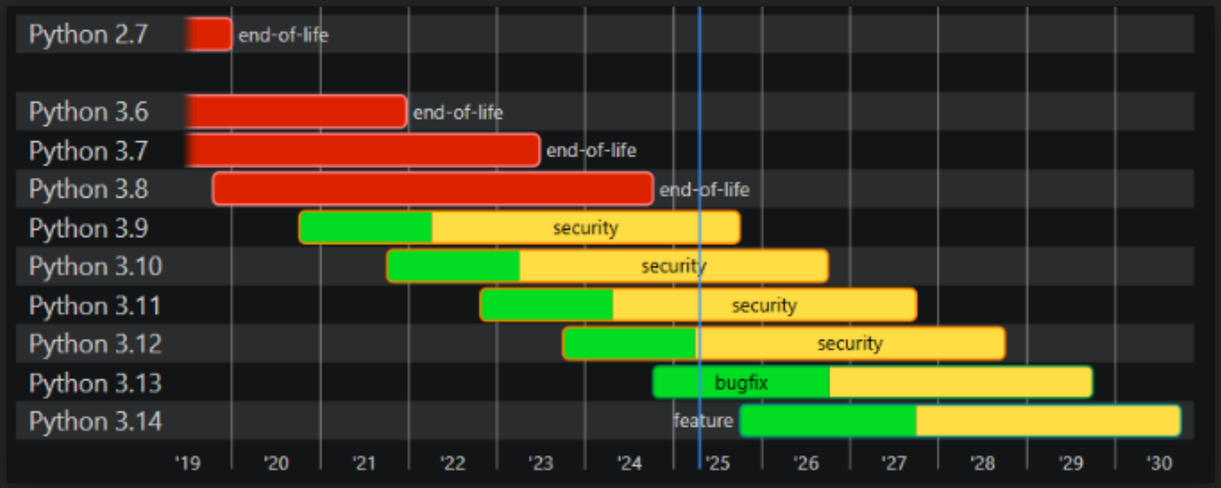
- **Python** is a high-level, interpreted programming language.
- Created by **Guido van Rossum** and first released in **1991**.
- Python's design philosophy emphasizes:
  - **Code readability**
  - Use of significant whitespace.

Why Python?

- **Simple and readable** syntax.
- Large **standard library**.
- **Cross-platform**: Works on Windows, Mac, Linux, and others.
- Popular in:
  - **Web development**
  - **Data science**
  - **Automation**
  - **Machine Learning**

Versions of Python

- **Python 2** (Released in 2000)
  - Legacy, not supported after 2020.
  - Incompatible with Python 3.
- **Python 3** (Released in 2008)
  - Current and actively supported.
  - Many improvements over Python 2.
- **Python 4** (Release -- no time soon)
  - Massive overhaul, breaking change
  - python 2 to 3 was a huge task.



https://devguide.python.org/versions/

Python Enhancement Proposals (PEP)

https://peps.python.org/pep-0000/#

PEP is the style guide for Python code.

- all files have a top level docstring

```python
"""
PEP Compliant Python Code Example
"""

NOUN="Name"

def print_full_name(name):
    """
    Function takes one argument...
    ...
    args name
    """
    print(NOUN + ": " + name)

def main():
    """main function to demonstrate PEP compliances"""
    fullname = "Guido van Rossum"
    print_full_name(name)

if __name__ == "__main__":
    main()
```

PEP is the style guide for Python code:

- all functions should have docstrings
- can be single or multiline
- if you have arguments or returns then they have be to be documented

```python
"""
PEP Compliant Python Code Example
"""

NOUN="Name"

def print_full_name(name):
    """
    Function takes one argument...
    ...
    args name
    """
    print(NOUN + ": " + name)

def main():
    """main function to demonstrate PEP compliances"""
    fullname = "Guido van Rossum"
    print_full_name(name)

if __name__ == "__main__":
    main()
```

PEP is the style guide for Python code:

- naming conventions for functions are lower snake case

```python
"""
PEP Compliant Python Code Example
"""

NOUN="Name"

def print_full_name(name):
    """
    Function takes one argument...
    ...
    args name
    """
    print(NOUN + ": " + name)

def main():
    """main function to demonstrate PEP compliances"""
    fullname = "Guido van Rossum"
    print_full_name(name)

if __name__ == "__main__":
    main()
```

PEP is the style guide for Python code:

- constant variables are uppercase / upper snake case
- local variables are lowercase

```python
"""
PEP Compliant Python Code Example
"""

NOUN="Name"

def print_full_name(name):
    """
    Function takes one argument...
    ...
    args name
    """
    print(NOUN + ": " + name)

def main():
    """main function to demonstrate PEP compliances"""
    fullname = "Guido van Rossum"
    print_full_name(name)

if __name__ == "__main__":
    main()
```

Package Manager: `pip`

- **pip** stands for "Pip Installs Packages".
- Used to install, update, and remove Python packages.

Common `pip` commands:

- Install a package: `pip install package_name`
- Upgrade a package: `pip install --upgrade package_name`
- List installed packages: `pip list`

```
~/GitHub/Learning_Python via v3.12.3 took 13s
❯ pip --help

Usage:
  pip <command> [options]

Commands:
  install                     Install packages.
  download                    Download packages.
  uninstall                   Uninstall packages.
  freeze                      Output installed packages in requirements format.
  inspect                     Inspect the python environment.
  list                        List installed packages.
  show                        Show information about installed packages.
  check                       Verify installed packages have compatible dependencies.
  config                      Manage local and global configuration.
  search                      Search PyPI for packages.
  cache                       Inspect and manage pip's wheel cache.
  index                       Inspect information available from package indexes.
  wheel                       Build wheels from your requirements.
  hash                        Compute hashes of package archives.
  completion                  A helper command used for command completion.
  debug                       Show information useful for debugging.
  help                        Show help for commands.
```

Writing Code

Variables and Data Types

- **Variables**:
  - Store data values.
  - implicit byt default
  - explicit if you want it to be

- **Basic Data Types** (implicit):
  - `int`: Integer numbers (e.g., 5)
  - `float`: Decimal numbers (e.g., 5.0)
  - `str`: String (e.g., "Hello")
  - `bool`: Boolean (True/False)

```
count = 5
factor = 1.16803
name = "Guido van Rossum"
is_active = True
is_inverse_state = False

---

count: int = 5
factor: float = 1.16803
name: str = "Guido van Rossum"
is_active: bool = True
is_inverse_state: bool = False
```

Indentation and Syntax

- **Python** uses **indentation** to define the structure of code (instead of brackets).
  - Consistent indentation is crucial for Python programs.
- **Syntax** in Python is designed to be clean and readable.
  - No need for semicolons to terminate statements.
  - Code blocks are identified by indentation levels, not braces.

```python
# Function definition with proper indentation
def greet(name: str) -> None:
    if name:
        print(f"Hello, {name}!")
    else:
        print("Hello, World!")

# Calling the function
greet("Guido van Rossum")
greet("")

# Loop with indentation
for i in range(5):
    print(i)

# Conditional statement with indentation
x = 10
if x > 5:
    print("x is greater than 5")
else:
    print("x is 5 or less")
```

Functions

- Functions are defined using the `def` keyword.
- can take arguments
  - implicit and explict data types
  - return values
- docstrings should be implemented

```python
def add(a, b):
    """
    Adds two numbers together.

    Parameters:
    a (int or float): The first number.
    b (int or float): The second number.

    Returns:
    int or float: The sum of the two numbers.
    """
    return a + b

def add_safe(a: int, b: int) -> int:
    """
    Adds two numbers together with explicit data types.

    Parameters:
    a (int): The first number.
    b (int): The second number.

    Returns:
    int: The sum of the two numbers.
    """
    return a + b
```

The `main()` Function

- In Python, scripts can have a `main()` function.

```python
import sys  # Module for system-specific parameters and functions

def main():
    """
    This is the main function.
    """
    print("This is the main function.")
    sys.exit(0)

if __name__ == "__main__":
    main()
```

The `main()` Function

- In Python, scripts can have a `main()` function.

- Ensures that the main() function is executed only when the script is run directly, not when imported as a module.
  - You can use the following idiom to execute code only when the script is run directly (and not imported as a module)

```python
import sys  # Module for system-specific parameters and functions

def main():
    """
    This is the main function.
    """
    print("This is the main function.")
    sys.exit(0)

if __name__ == "__main__":
    main()
```

Conditionals and Loops

- **Conditionals:** `if`, `elif`, `else`

```
if x > 0:
    print("Positive")
elif x == 0:
    print("Zero")
else:
    print("Negative")
```

- **Loops:** `for` and `while`

```
for i in range(5):
    print(i)

x = 10

while x > 0:
    x -= 1
```

Conclusion

- Python is an accessible and versatile programming language.
- Its applications span from web development to data science.
- Python's package manager, `pip`, makes it easy to manage packages.
- Understanding basics like variables, data types, functions, conditionals, PEP standards, and indentation gives a strong foundation.