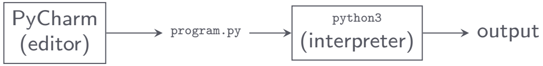
Your First Programs

Outline

Programming in Python

Programming in Python

The Python workflow



Programming in Python

Programming in Python

Program: `helloworld.py`

- Standard output: the message “Hello, World”

Programming in Python

Program: `helloworld.py`

- Standard output: the message “Hello, World”

```
>_ ~/workspace/ipp/programs

$ _
```

Programming in Python

Program: `helloworld.py`

- Standard output: the message “Hello, World”

```
>_ ~/workspace/ipp/programs

$ python3 helloworld.py
```

Programming in Python

Program: `helloworld.py`

- Standard output: the message “Hello, World”

```
~/workspace/ipp/programs
```

```
$ python3 helloworld.py
Hello, World
$ _
```

Programming in Python

Programming in Python

helloworld.py

```python
# Writes the message "Hello, World" to standard output.

import stdio

stdio.writeln("Hello, World")
```

Application Programming Interface

Application Programming Interface

The application programming interface (API) for a library provides a summary of the functions in the library

Application Programming Interface

The application programming interface (API) for a library provides a summary of the functions in the library

Example

stdio	
`writeln(x = "")`	writes `x` followed by newline to standard output
`write(x = "")`	writes `x` to standard output

Errors in a Program

Errors in a Program

Compile-time errors are identified and reported by Python when it compiles a program

Errors in a Program

Compile-time errors are identified and reported by Python when it compiles a program

Example

helloworld.py

```
# Writes the message "Hello, World" to standard output.

import stdio

stdio.writeln("Hello, World"]
```

Errors in a Program

Compile-time errors are identified and reported by Python when it compiles a program

Example

```
helloworld.py

# Writes the message "Hello, World" to standard output.

import stdio

stdio.writeln("Hello, World"]
```

```
~/workspace/ipp/programs

$ _
```

Errors in a Program

Compile-time errors are identified and reported by Python when it compiles a program

Example

helloworld.py

```
# Writes the message "Hello, World" to standard output.

import stdio

stdio.writeln("Hello, World"]
```

~/workspace/ipp/programs

```
$ python3 helloworld.py
```

Errors in a Program

Compile-time errors are identified and reported by Python when it compiles a program

Example

```
helloworld.py
# Writes the message "Hello, World" to standard output.

import stdio

stdio.writeln("Hello, World"]
```

```
~/workspace/ipp/programs
$ python3 helloworld.py
  File "helloworld.py", line 5
    stdio.writeln('Hello, World']
                                ^
SyntaxError: invalid syntax
$ _
```

Errors in a Program

Errors in a Program

Run-time errors are identified and reported by Python when it runs a program

Errors in a Program

Run-time errors are identified and reported by Python when it runs a program

Example

helloworld.py

```
# Writes the message "Hello, World" to standard output.

stdio.writeln("Hello, World")
```

Errors in a Program

Run-time errors are identified and reported by Python when it runs a program

Example

helloworld.py

```
# Writes the message "Hello, World" to standard output.

stdio.writeln("Hello, World")
```

~/workspace/ipp/programs

```
$ _
```

Errors in a Program

Run-time errors are identified and reported by Python when it runs a program

Example

helloworld.py

```
# Writes the message "Hello, World" to standard output.

stdio.writeln("Hello, World")
```

~/workspace/ipp/programs

```
$ python3 helloworld.py
```

Errors in a Program

Run-time errors are identified and reported by Python when it runs a program

Example

```
helloworld.py
# Writes the message "Hello, World" to standard output.

stdio.writeln("Hello, World")
```

```
>_ ~/workspace/ipp/programs
$ python3 helloworld.py
Traceback (most recent call last):
  File "helloworld.py", line 3, in <module>
    stdio.writeln('Hello, World')
NameError: name 'stdio' is not defined
$ _
```

Errors in a Program

Errors in a Program

Logic errors are not identified or reported by Python, but produce unintended output

Errors in a Program

Logic errors are not identified or reported by Python, but produce unintended output

Example

helloworld.py

```
# Writes the message "Hello, World" to standard output.

import stdio

stdio.write("Hello, World")
```

Errors in a Program

Logic errors are not identified or reported by Python, but produce unintended output

Example

```
helloworld.py
```

```python
# Writes the message "Hello, World" to standard output.

import stdio

stdio.write("Hello, World")
```

```
~/workspace/ipp/programs
```

```
$ _
```

Errors in a Program

Logic errors are not identified or reported by Python, but produce unintended output

Example

helloworld.py

```
# Writes the message "Hello, World" to standard output.

import stdio

stdio.write("Hello, World")
```

~/workspace/ipp/programs

```
$ python3 helloworld.py
```

Errors in a Program

Logic errors are not identified or reported by Python, but produce unintended output

Example

helloworld.py

```
# Writes the message "Hello, World" to standard output.

import stdio

stdio.write("Hello, World")
```

~/workspace/ipp/programs

```
$ python3 helloworld.py
Hello, World$ _
```

Input and Output

Input and Output

$$\text{input} \xrightarrow{\quad} \boxed{\texttt{program.py}} \xrightarrow{\quad} \text{output}$$

Input and Output

$$\begin{tikzcd} \text{input} \arrow[r] & \boxed{\texttt{program.py}} \arrow[r] & \text{output} \end{tikzcd}$$

Input types:

- Command-line input
- Standard input
- File input

Input and Output

$$\begin{tikzcd}\text{input} \arrow[r] & \boxed{\texttt{program.py}} \arrow[r] & \text{output}\end{tikzcd}$$

Input types:

- Command-line input
- Standard input
- File input

Output types:

- Standard output
- Graphical output
- Audio output
- File output

Input and Output

Input and Output

Command-line inputs are strings listed right next to the program name during execution

```
>_ ~/workspace/ipp/programs

$ python3 program.py input1 input2 input3 ...
```

Input and Output

Command-line inputs are strings listed right next to the program name during execution

```
>_ ~/workspace/ipp/programs

$ python3 program.py input1 input2 input3 ...
```

The inputs are accessed within the program as `sys.argv[1]`, `sys.argv[2]`, `sys.argv[3]`, and so on

Input and Output

Command-line inputs are strings listed right next to the program name during execution

```
>_ ~/workspace/ipp/programs

$ python3 program.py input1 input2 input3 ...
```

The inputs are accessed within the program as `sys.argv[1]`, `sys.argv[2]`, `sys.argv[3]`, and so on

`sys.argv[0]` stores the name of the program

Input and Output

Command-line inputs are strings listed right next to the program name during execution

```
>_ ~/workspace/ipp/programs

$ python3 program.py input1 input2 input3 ...
```

The inputs are accessed within the program as `sys.argv[1]`, `sys.argv[2]`, `sys.argv[3]`, and so on

`sys.argv[0]` stores the name of the program

Example

```
>_ ~/workspace/ipp/programs

$ python3 program.py Galileo "Isaac Newton" Einstein
```

sys.argv[0]	sys.argv[1]	sys.argv[2]	sys.argv[3]
"program.py"	"Galileo"	"Isaac Newton"	"Einstein"

Input and Output

Input and Output

Program: `useargument.py`

- Command-line input: a name
- Standard output: a message containing the name

Input and Output

Program: `useargument.py`

- Command-line input: a name
- Standard output: a message containing the name

```
>_ ~/workspace/ipp/programs

$ _
```

Input and Output

Program: `useargument.py`

- Command-line input: a name
- Standard output: a message containing the name

```
>_ ~/workspace/ipp/programs
```

```
$ python3 useargument.py Alice
```

Input and Output

Program: `useargument.py`

- Command-line input: a name
- Standard output: a message containing the name

```
>_ ~/workspace/ipp/programs
```

```
$ python3 useargument.py Alice
Hi, Alice. How are you?
$ _
```

Input and Output

Program: `useargument.py`

- Command-line input: a name
- Standard output: a message containing the name

```
>_ ~/workspace/ipp/programs
```

```
$ python3 useargument.py Alice
Hi, Alice. How are you?
$ python3 useargument.py Bob
```

Input and Output

Program: `useargument.py`

- Command-line input: a name
- Standard output: a message containing the name

```
>_ ~/workspace/ipp/programs
```

```
$ python3 useargument.py Alice
Hi, Alice. How are you?
$ python3 useargument.py Bob
Hi, Bob. How are you?
$ _
```

Input and Output

Program: `useargument.py`

- Command-line input: a name
- Standard output: a message containing the name

```
>_ ~/workspace/ipp/programs

$ python3 useargument.py Alice
Hi, Alice. How are you?
$ python3 useargument.py Bob
Hi, Bob. How are you?
$ python3 useargument.py Carol
```

Input and Output

Program: `useargument.py`

- Command-line input: a name
- Standard output: a message containing the name

```
>_ ~/workspace/ipp/programs
```

```
$ python3 useargument.py Alice
Hi, Alice. How are you?
$ python3 useargument.py Bob
Hi, Bob. How are you?
$ python3 useargument.py Carol
Hi, Carol. How are you?
$ _
```

Input and Output

Input and Output

useargument.py

```python
# Accepts a name as command-line argument; and writes a message containing that name to standard
# output.

import stdio
import sys

stdio.write("Hi, ")
stdio.write(sys.argv[1])
stdio.writeln(". How are you?")
```