Imperative Programming: Your First Programs

Introduction to Programming in Python

Outline

Outline

Outline

The Python Language

Programming in Python

Application Programming Interface (API)

Input and Output

Errors in a Program

Outline

Outline

Outline

Outline

Outline

The Python Language

The Python Language

General-purpose, high-level, object-oriented programming language

The Python Language

General-purpose, high-level, object-oriented programming language

Key features:

The Python Language

General-purpose, high-level, object-oriented programming language

Key features:

- Simple, easy-to-read syntax

The Python Language

General-purpose, high-level, object-oriented programming language

Key features:

- Simple, easy-to-read syntax
- Powerful capabilities

The Python Language

General-purpose, high-level, object-oriented programming language

Key features:

- Simple, easy-to-read syntax
- Powerful capabilities
- Wide range of applications

Programming in Python

Programming in Python

Step 1: Create/edit the program (eg, `program.py`)

Programming in Python

Step 1: Create/edit the program (eg, `program.py`)

Step 2: Run the program

```
$ _
```

Programming in Python

Step 1: Create/edit the program (eg, `program.py`)

Step 2: Run the program

```
$ python3 program.py
```

Programming in Python

Step 1: Create/edit the program (eg, `program.py`)

Step 2: Run the program

```
$ python3 program.py
<program output>
$ _
```

Programming in Python

Step 1: Create/edit the program (eg, `program.py`)

Step 2: Run the program

```
$ python3 program.py
<program output>
$ _
```

Repeat steps 1 and 2 until program output matches expected

Programming in Python

Programming in Python

Program (`helloworld.py`):

Programming in Python

Program (`helloworld.py`):

- Standard output: the message `"Hello, World"`

Programming in Python

Program (`helloworld.py`):

- Standard output: the message `"Hello, World"`

```
$ _
```

Programming in Python

Program (`helloworld.py`):

- Standard output: the message `"Hello, World"`

```
$ python3 helloworld.py
```

Programming in Python

Program (`helloworld.py`):

- Standard output: the message `"Hello, World"`

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

Programming in Python

Programming in Python

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

import stdio

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

Application Programming Interface (API)

Application Programming Interface (API)

API is a set of protocols that allows different software applications to communicate with one another

Application Programming Interface (API)

API is a set of protocols that allows different software applications to communicate with one another

Example (API for `stdio` library):

Application Programming Interface (API)

API is a set of protocols that allows different software applications to communicate with one another

Example (API for `stdio` library):

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

Input and Output

Input and Output

Input and Output

Input types:

Input and Output

Input types:

- Command-line input

Input and Output

Input types:

- Command-line input
- Standard input

Input and Output

Input types:

- Command-line input
- Standard input
- File input

Input and Output

Input types:

- Command-line input
- Standard input
- File input

Output types:

Input and Output

Input types:

- Command-line input
- Standard input
- File input

Output types:

- Standard output

Input and Output

Input types:

- Command-line input
- Standard input
- File input

Output types:

- Standard output
- Standard draw

Input and Output

Input types:

- Command-line input
- Standard input
- File input

Output types:

- Standard output
- Standard draw
- Standard audio

Input and Output

Input types:

- Command-line input
- Standard input
- File input

Output types:

- Standard output
- Standard draw
- Standard audio
- File output

Input and Output

Input and Output ▶ Command-line Input

Command-line inputs (aka arguments) are strings listed next to the program name during execution

```
$ _
```

Input and Output ▶ Command-line Input

Command-line inputs (aka arguments) are strings listed next to the program name during execution

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

Input and Output ▶ Command-line Input

Command-line inputs (aka arguments) are strings listed next to the program name during execution

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

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

Input and Output ▶ Command-line Input

Command-line inputs (aka arguments) are strings listed next to the program name during execution

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

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

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

Input and Output ▶ Command-line Input

Example:

Input and Output ▶ Command-line Input

Example:

```
$ _
```

Input and Output ▶ Command-line Input

Example:

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

Input and Output ▶ Command-line Input

Example:

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

sys.argv[0]	sys.argv[1]	sys.argv[2]	sys.argv[3]

Input and Output ▶ Command-line Input

Example:

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

sys.argv[0]	sys.argv[1]	sys.argv[2]	sys.argv[3]
program.py			

Input and Output ▶ Command-line Input

Example:

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

sys.argv[0]	sys.argv[1]	sys.argv[2]	sys.argv[3]
program.py	Galileo		

Input and Output ▶ Command-line Input

Example:

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

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

Input and Output ▶ Command-line Input

Example:

```
$ 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 ▶ Command-line Input

Program (`useargument.py`):

Program (`useargument.py`):

- Command-line input: a name

Input and Output ▶ Command-line Input

Program (`useargument.py`):

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

Input and Output ▶ Command-line Input

Program (`useargument.py`):

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

```
$ _
```

Input and Output ▶ Command-line Input

Program (`useargument.py`):

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

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

Input and Output ▶ Command-line Input

Program (`useargument.py`):

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

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

Input and Output ▶ Command-line Input

Program (`useargument.py`):

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

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

Input and Output ▶ Command-line Input

Program (`useargument.py`):

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

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

Input and Output ▶ Command-line Input

Program (`useargument.py`):

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

```
$ 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 ▶ Command-line Input

Program (`useargument.py`):

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

```
$ 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 ▶ Command-line Input

Input and Output ▶ Command-line Input

```
1 # Receives a name as command-line input; and writes a message containing
2 # that name as standard output.
3
4 import stdio
5 import sys
6
7 stdio.write("Hi, ")
8 stdio.write(sys.argv[1])
9 stdio.writeln(". How are you?")
```

Errors in a Program

Errors in a Program ▶ Compile-time Errors

Errors in a Program ▶ Compile-time Errors

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

Errors in a Program ▶ Compile-time Errors

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

Example:

Errors in a Program ▶ Compile-time Errors

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

Example:

```
1 # Receives a name as command-line input; and writes a message containing
2 # that name as standard output.
3
4 import stdio
5 import sys
6
7 stdio.write("Hi, ")
8 stdio.write(sys.argv[1)
9 stdio.writeln(". How are you?")
```

Errors in a Program ▶ Compile-time Errors

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

Example:

```
1 # Receives a name as command-line input; and writes a message containing
2 # that name as standard output.
3
4 import stdio
5 import sys
6
7 stdio.write("Hi, ")
8 stdio.write(sys.argv[1)
9 stdio.writeln(". How are you?")
```

```
$ _
```

Errors in a Program ▶ Compile-time Errors

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

Example:

```
1 # Receives a name as command-line input; and writes a message containing
2 # that name as standard output.
3
4 import stdio
5 import sys
6
7 stdio.write("Hi, ")
8 stdio.write(sys.argv[1)
9 stdio.writeln(". How are you?")
```

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

Errors in a Program ▶ Compile-time Errors

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

Example:

```
1 # Receives a name as command-line input; and writes a message containing
2 # that name as standard output.
3
4 import stdio
5 import sys
6
7 stdio.write("Hi, ")
8 stdio.write(sys.argv[1)
9 stdio.writeln(". How are you?")
```

```
$ python3 useargument.py Alice
  File "~/tmp/useargument.py", line 8
    stdio.write(sys.argv[1)
                        ^
SyntaxError: closing parenthesis ')' does not match opening parenthesis '['
$ _
```

Errors in a Program

Errors in a Program ▶ Run-time Errors

Errors in a Program ▶ Run-time Errors

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

Errors in a Program ▶ Run-time Errors

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

Example:

Errors in a Program ▶ Run-time Errors

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

Example:

```
1 # Receives a name as command-line input; and writes a message containing
2 # that name as standard output.
3
4 import stdio
5
6 stdio.write("Hi, ")
7 stdio.write(sys.argv[1])
8 stdio.writeln(". How are you?")
```

Errors in a Program ▶ Run-time Errors

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

Example:

```
1 # Receives a name as command-line input; and writes a message containing
2 # that name as standard output.
3
4 import stdio
5
6 stdio.write("Hi, ")
7 stdio.write(sys.argv[1])
8 stdio.writeln(". How are you?")
```

```
$ _
```

Errors in a Program ▶ Run-time Errors

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

Example:

```
1 # Receives a name as command-line input; and writes a message containing
2 # that name as standard output.
3
4 import stdio
5
6 stdio.write("Hi, ")
7 stdio.write(sys.argv[1])
8 stdio.writeln(". How are you?")
```

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

Errors in a Program ▶ Run-time Errors

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

Example:

```
1 # Receives a name as command-line input; and writes a message containing
2 # that name as standard output.
3
4 import stdio
5
6 stdio.write("Hi, ")
7 stdio.write(sys.argv[1])
8 stdio.writeln(". How are you?")
```

```
$ python3 useargument.py Alice
Hi, Traceback (most recent call last):
  File "~/tmp/useargument.py", line 7, in <module>
    stdio.write(sys.argv[1])
                ^^^
NameError: name 'sys' is not defined. Did you forget to import 'sys'?
$ _
```

Errors in a Program

Errors in a Program ▶ Logic Errors

Errors in a Program ▶ Logic Errors

Logic errors are neither identified nor reported by Python, but produce unintended output

Errors in a Program ▶ Logic Errors

Logic errors are neither identified nor reported by Python, but produce unintended output

Example:

Errors in a Program ▶ Logic Errors

Logic errors are neither identified nor reported by Python, but produce unintended output

Example:

```
1 # Receives a name as command-line input; and writes a message containing
2 # that name as standard output.
3
4 import stdio
5 import sys
6
7 stdio.write("Hi, ")
8 stdio.write(sys.argv[0])
9 stdio.writeln(". How are you?")
```

Errors in a Program ▶ Logic Errors

Logic errors are neither identified nor reported by Python, but produce unintended output

Example:

```
1 # Receives a name as command-line input; and writes a message containing
2 # that name as standard output.
3
4 import stdio
5 import sys
6
7 stdio.write("Hi, ")
8 stdio.write(sys.argv[0])
9 stdio.writeln(". How are you?")
```

```
$ _
```

Errors in a Program ▶ Logic Errors

Logic errors are neither identified nor reported by Python, but produce unintended output

Example:

```
1 # Receives a name as command-line input; and writes a message containing
2 # that name as standard output.
3
4 import stdio
5 import sys
6
7 stdio.write("Hi, ")
8 stdio.write(sys.argv[0])
9 stdio.writeln(". How are you?")
```

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

Errors in a Program ▶ Logic Errors

Logic errors are neither identified nor reported by Python, but produce unintended output

Example:

```
1 # Receives a name as command-line input; and writes a message containing
2 # that name as standard output.
3
4 import stdio
5 import sys
6
7 stdio.write("Hi, ")
8 stdio.write(sys.argv[0])
9 stdio.writeln(". How are you?")
```

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