Programming

- program: sequence of instructions to the computer
- describing how to perform a task

- programming:

  act of designing and implementing programs

Machine Code

- programs stored as machine instructions
- machine code

- instructions encoded as numbers
- depends on the processor type

Machine Code Example

1. Move contents of memory location 40000 into CPU.
2. If that value is greater than 100, continue with instruction that is stored in memory location 11280.

- example machine code on a PC:

```
161 40000 45 100 127 11280
```

High-Level Language

- machine code is very difficult to write by humans

- write programs in a high-level programming language
- source code

- independent of processor type
- lots of languages with different characteristics

Conversion to Machine Code

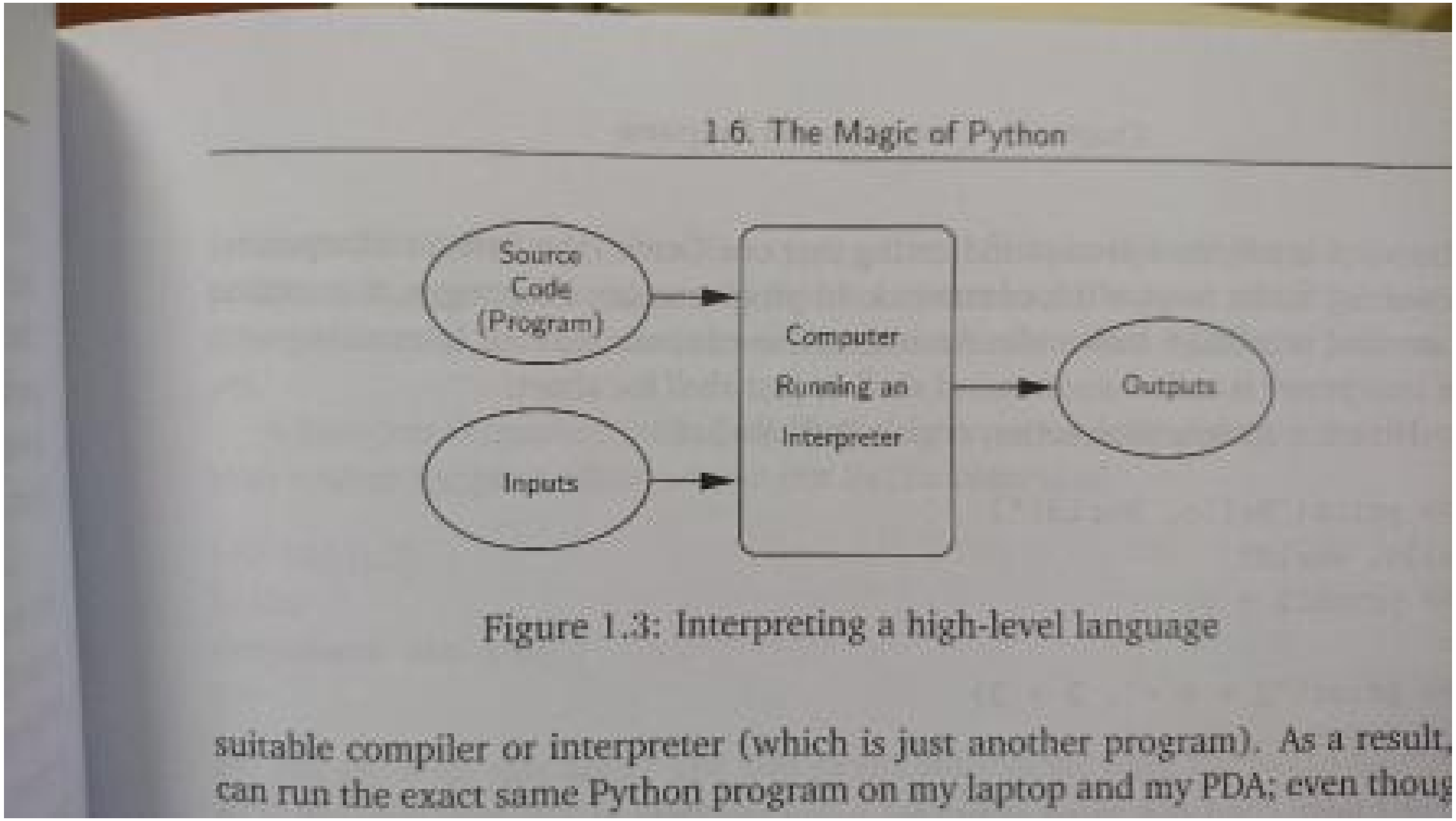
- use programs to convert source code to machine code
- **for a particular processor**

- generated machine code different between processors
- programmer need not worry

Conversion Methods

- interpreting: convert step by step during execution
- compiling: first convert all, then execute

Interpreting



1.6. The Magic of Python

Source Code (Program)
Inputs
Computer Running an Interpreter
Outputs

Figure 1.3: Interpreting a high-level language

suitable compiler or interpreter (which is just another program). As a result,
can run the exact same Python program on my laptop and my PDA; even thoug

Compiling

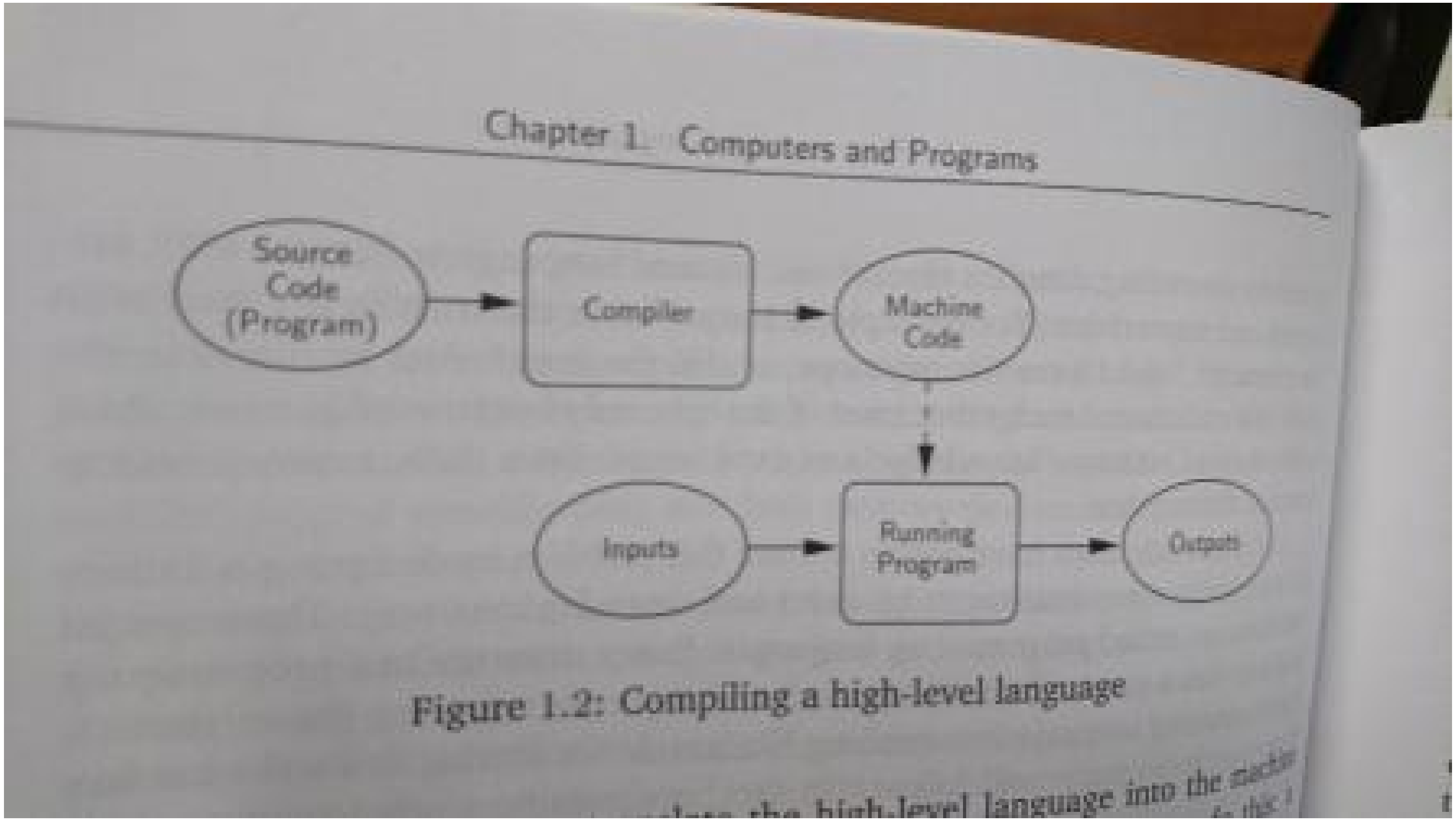
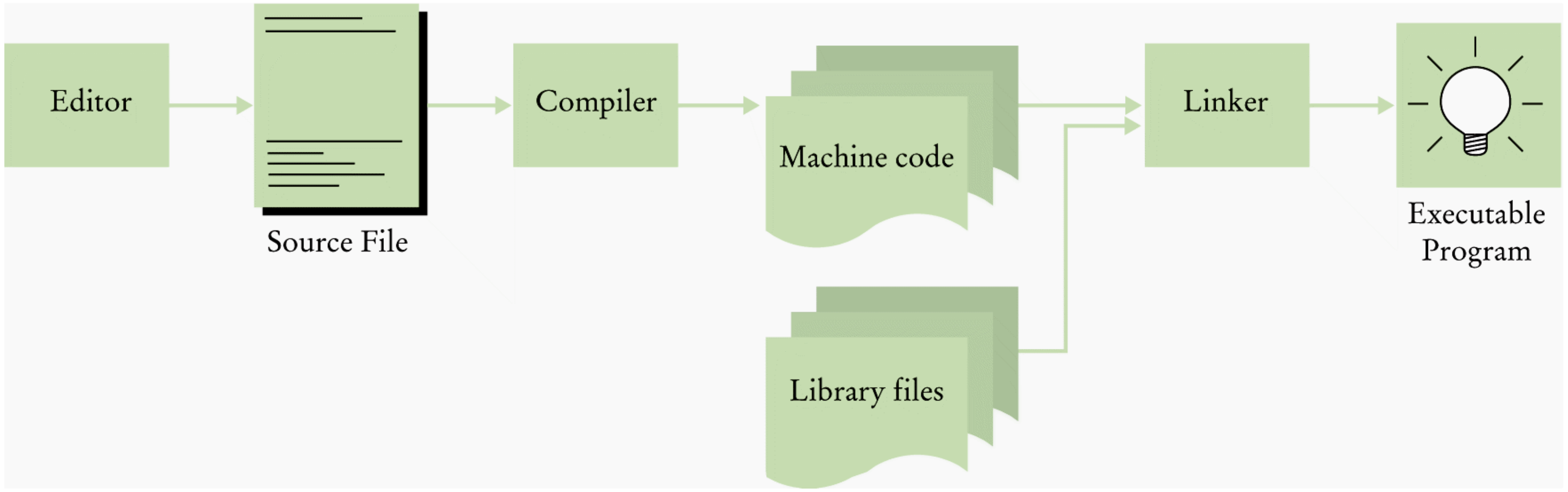

Chapter 1. Computers and Programs

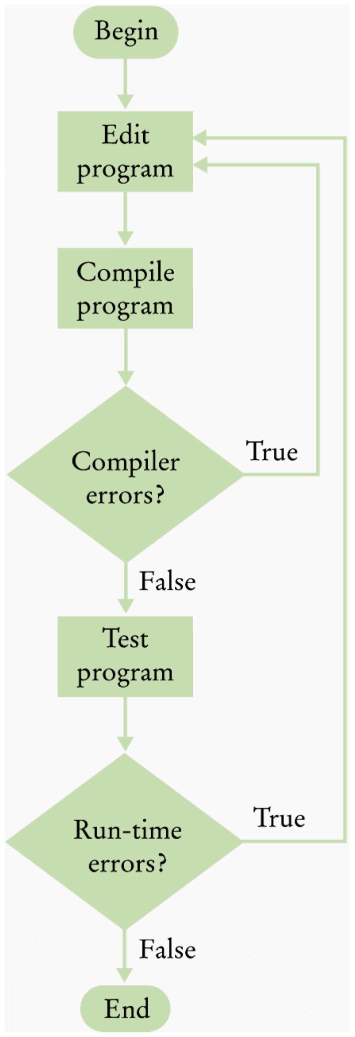
Figure 1.2: Compiling a high-level language

the high-level language into the machine

Compiling: Stages



Compiling: Development Workflow



Python

- created by Guido van Rossum
- in early 1990s

Monty Python

- named after a British comedy group from the 1970s

Popularity

- web and enterprise applications (IEEE):

  The Top Programming Languages 2017

- for teaching programming (ACM):

  Python is Now the Most Popular Introductory Teaching Language at Top U.S. Universities

Popularity - 2

- developing projects (GitHub):

  GitHub Octoverse 2017

- questions and discussions (StackOverflow):

  The Incredible Growth of Python

Who's Using It?

- Youtube, Google
- Dropbox
- Instagram
- Pinterest
- Reddit
- NASA
- IL&M
- …

Application Areas

- web applications
- data science
- scientific computation
- system administration
- …

Source Files

- extension for source files: `.py`
- running a source file:

```
python SOURCE_FILE.py
```

Interactive Mode

- REPL: Read Eval Print Loop
- ask a question, get an answer

- shows prompt, waits for input
- evaluates input
- prints result
- shows prompt, waits for input
- …

Python REPL

- run:

```
python
```

- and you see the prompt:

```
Python 3.6.2 ...
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" ...
>>>
```

Jupyter

- interactive environment for many languages
- Python, R, Julia, JavaScript, Haskell, C++, …

- on the console: `jupyter-console`
- in the browser: `jupyter-notebook`

Development Environments

- any text editor will do

- PyCharm
- Eclipse PyDev
- Spyder
- IDLE
- …

Expressions

- an expression describes a computation
- evaluating it results in a value

- examples:

$$35 + 7$$

$$2^5$$

$$14!$$

Expressions in REPL

- type expression, get result

```
>>> 35 + 7
42
>>> 13 * 3
39
>>> 6 + 7 * 4
34
```

Expression Components

- literals: values written directly
- operators: addition, multiplication, …

- only a literal:

```
42
```

- literals connected with operators:

```
13 * 3
```

Syntax Errors

- source code has to follow language rules
- what happens if:

```
6 +* 7
```

Assignment

- assignment: associate a value with a name
- variable: named value

- variables can be used in expressions
- value substitutes variable

Assignment in Python

- syntax:

```
name = expression
```

1. evaluate expression
2. associate resulting value with name

Statements

- assignment is a statement
- it doesn’t return a result
- not a question

- a source file consists of statements
- and comments: from `#` until end of line

Assignment Examples

```
>>> midterm = 85
>>> final = 78
>>> total = midterm * 0.45 + final * 0.55
>>> total
81.15
```

Assignment and Equality

- assignment is not equality!

```
>>> x = 41
>>> x = x + 1
>>> x
42
```

Name Rules

- start with letters
- can contain letters, digits and underscore
- no punctuation or white-space
- case sensitive: $A \neq a$

Missing Variable

- what happens if:

```
total = midterm * 0.3 + assignment * 0.3 + final * 0.4
```

Types

- every value has a type
- how data is to be interpreted

- numeric: integer (int), real (float)
- literal: if no decimal point then int, else float

- text: string of characters (str)
- literal: surrounded by double or single quotes

Type Examples

literal	type
`42`	`int`
`3.14159`	`float`
`'Hello'`	`str`
`"42"`	`str`

String Delimiters

- a string starting with `"` is only ended by `"`
- a string starting with `'` is only ended by `'`

```
"I said 'hello'."

'I said "hello".'
```

Multiline Strings

- putting a newline into a string: `\n`

```
'Mountain sheep are sweeter,\nvalley sheep are fatter.'
```

- multi-line strings: three quotes (double or single)

```
"""Mountain sheep are sweeter,
valley sheep are fatter."""
```

Arithmetic Operators

- addition: `x + y`
- subtraction: `x - y`
- multiplication: `x * y`
- division: `x / y`

- integer division: `x // y`
- division remainder (mod): `x % y`

- exponentiation: `x ** y`

Arithmetic Operator Examples

operator	expression	result	type
`+`	`6 + 7`	`13`	`int`
`*`	`6 * 7`	`42`	`int`
`/`	`15 / 6`	`2.5`	`float`
`//`	`15 // 6`	`2`	`int`
`%`	`15 % 6`	`3`	`int`
`**`	`4 ** 3`	`64`	`int`

String Concatenation

- addition on strings → concatenation

```
>>> "Hello," + "world!"
"Hello,world!"
>>> name = "Eric"
>>> greeting = "Hello," + " " + name + "!"
>>> greeting
"Hello, Eric!"
```

Type Errors

- operand types must match operation

```
>>> birth_year = 1991
>>> age = 2018 - birth_year
>>> "Python is " + age + " years old."
```

Functions

- take input: parameters (also called “arguments”)
- produce output: return values

Function Examples

- abs: absolute value, 1 parameter
- min: minimum, 2 parameters
- max: maximum, 2 parameters
- round: 2 parameters (value and precision)
- len: length, 1 parameter

Function Usage Examples

```
abs(-3)

min(midterm, final)

max(midterm, final)

round(total, 1)

len(greeting)
```

Functions as Operands

- functions can be operands in expressions
- replace function expression with its return value

```
abs(-3) + 3

min(3, -3) + max(3, -3)
```

Parameter Expressions

- function parameters are expressions

```
min(3 * 9, 4 * 8)

min(abs(-10), abs(3))
```

Type Conversions

- functions to convert values between types

```
>>> str(42)
'42'
>>> int("42")
42
>>> int(42)
42
```

Type Conversion Errors

- what’s the result of `int("Eric")`?

- a syntax error?
- a type error?

Type Errors

```
>>> birth_year = 1991
>>> age = 2018 - birth_year
>>> "Python is " + str(age) + " years old."
```

Input and Output

- interaction with the user
- output: print a string to the screen

```
print(message)
```

- input: read a string from the keyboard

```
variable = input(prompt)
```

Output Example

- a program to print a message

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

Output Example - 2

- a program to get an input and produce an output

```
name = input("What is your name? ")
message = "Hello, " + name + "!"
print(message)
```

Simple Flow

- get inputs from user
- process inputs and produce results
- output results

Simple Flow Example

```
response = input("In which year were you born? ")
birth_year = int(response)
age = 2018 - birth_year
message = "You are " + str(age) + " years old."
print(message)
```

Libraries

- library: collection of code
- functions, constants, ...
- grouped into packages

- import into your code

Importing Libraries

- syntax 1:

```python
from LIBRARY import NAME
```

- syntax 2:

```python
import LIBRARY

# use names as: LIBRARY.NAME
```

Import Example - 1

- importing a constant

```
>>> from math import pi
>>> pi
3.141592653589793
>>> r = 4.2
>>> area = pi * r ** 2
>>> area
55.41769440932395
```

Import Example - 2

- importing a function

```
>>> from math import pi, sqrt
>>> sqrt(area / pi)
4.2
```

Math Library Example

```python
# Given the radius, calculate the area of a circle.

from math import pi

response = input("What's the radius of the circle? ")
radius = float(response)
area = pi * radius ** 2
message = "The area is: " + str(area)
print(message)
```

Math Library Example - 2

```
# Given the area, calculate the radius of a circle.

import math

response = input("What's the area of the circle?" )
area = float(response)
radius = math.sqrt(area / math.pi)
message = "The radius is: " + str(radius)
print(message)
```