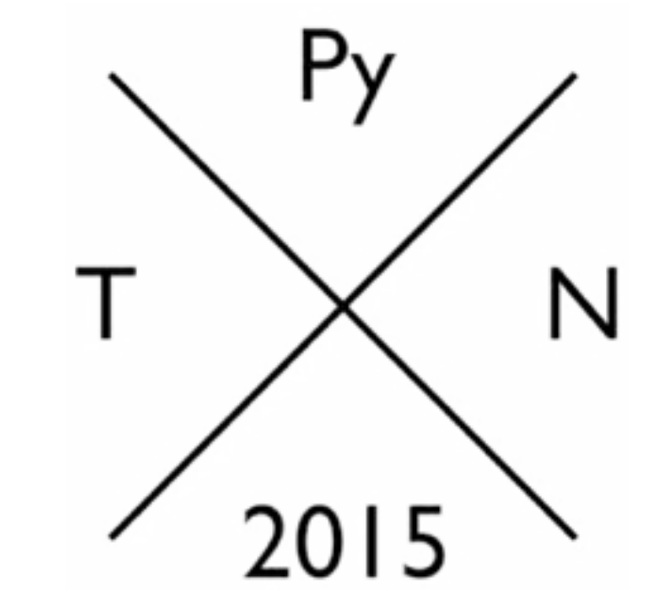
Write a Toy Language in python™ 3



T. Scot Clausing

@tsclausing

Write a Toy Language in  3

New programming languages targeting the JVM, JavaScript, and others, are popping up all over the place. We'll cover the basics of writing your own language in Python 3 to target the Python virtual machine.

Write a Toy Language in  3

New programming languages targeting the JVM, JavaScript, and others, are popping up all over the place. We'll cover the basics of writing your own language in Python 3 to target the Python virtual machine.

Do you have your computer?

Do you have Python 3?

T. Scot Clausing
@tsclausing

Come to "Write a Toy Language" pytennessee.org/schedule/prese… WITH Python 3 already installed and get CANDY :) @PyTennessee

Oh. One more thing.

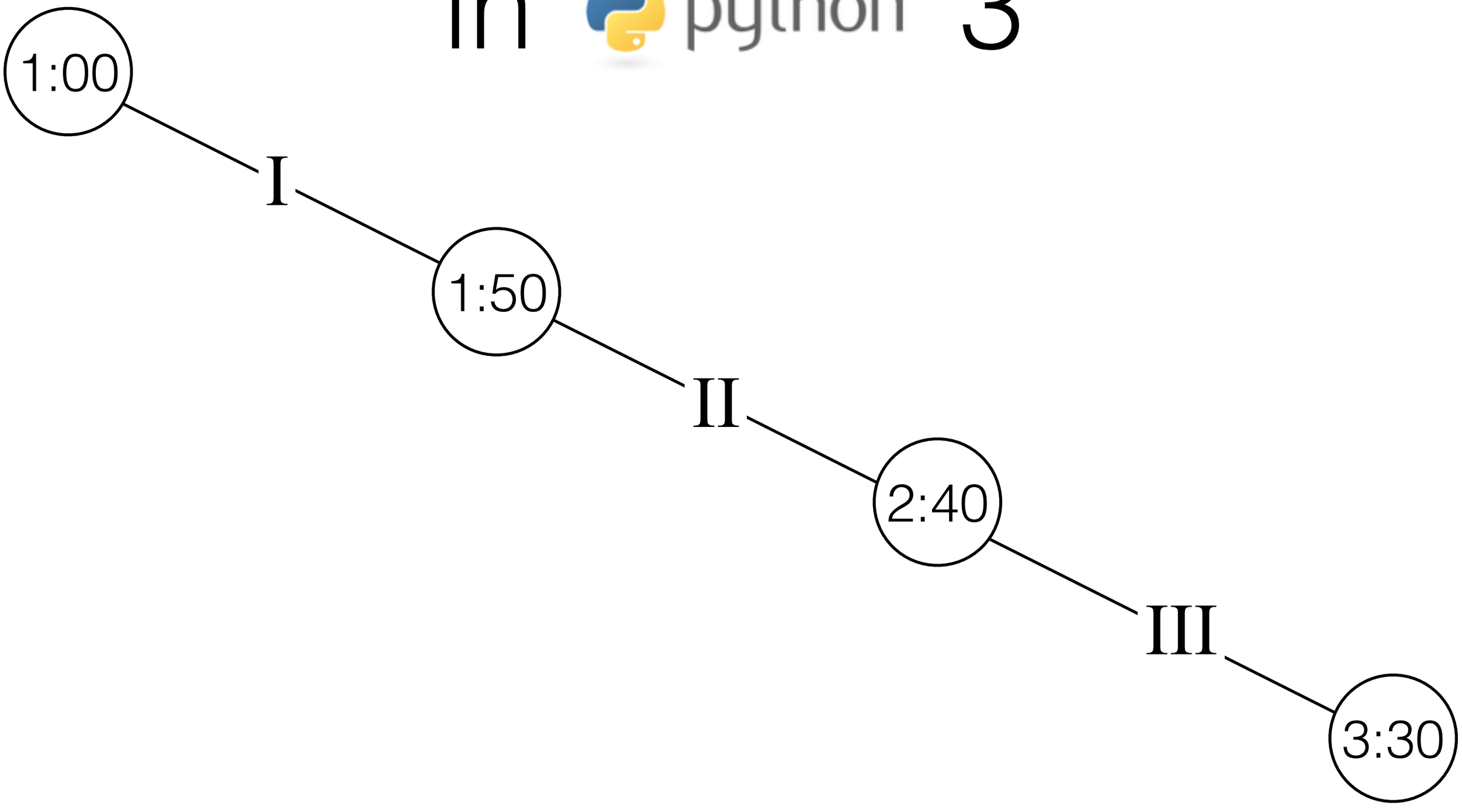
Write a Toy Language in  3

Write a Toy Language in  3

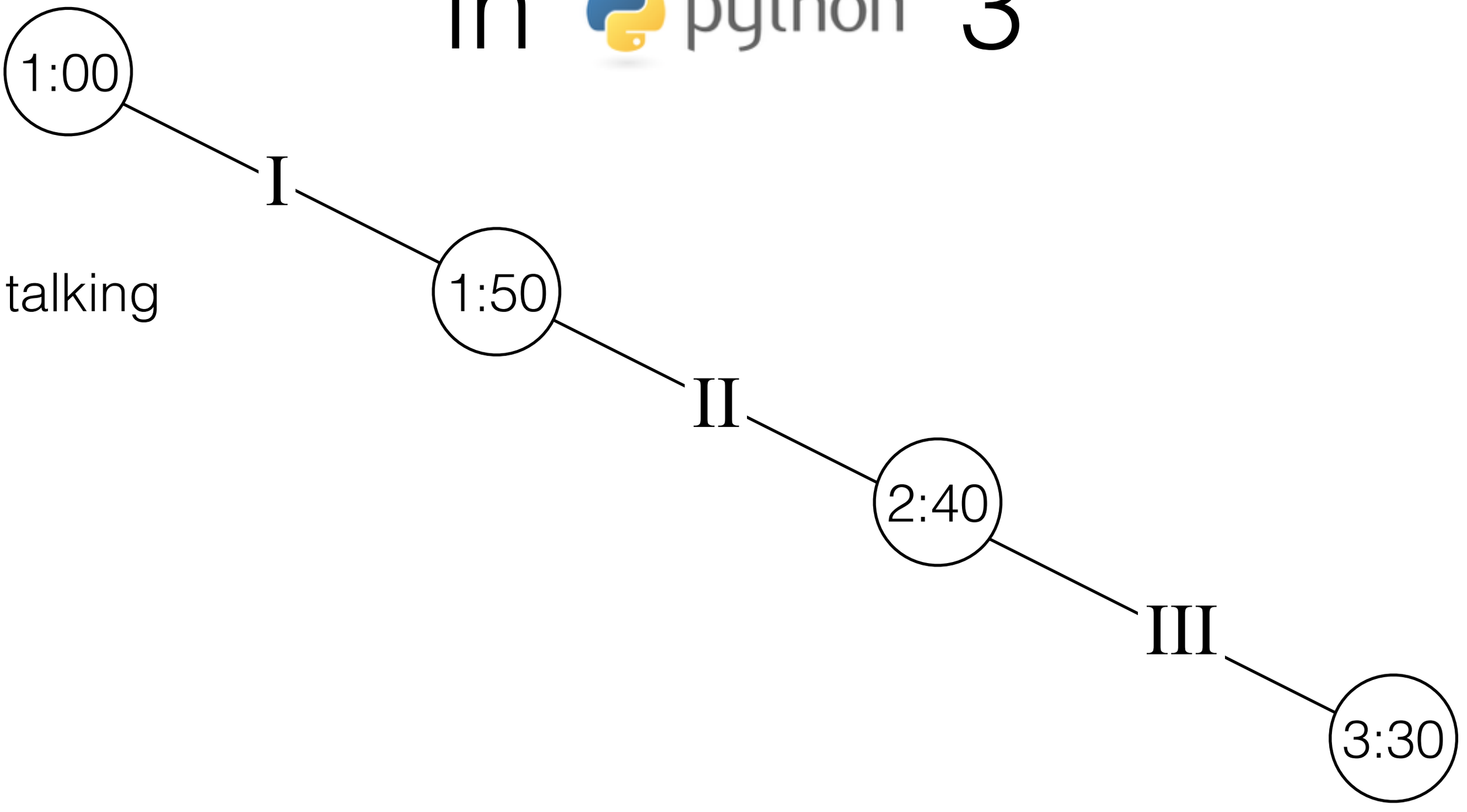
2.5 HOURS

oops.

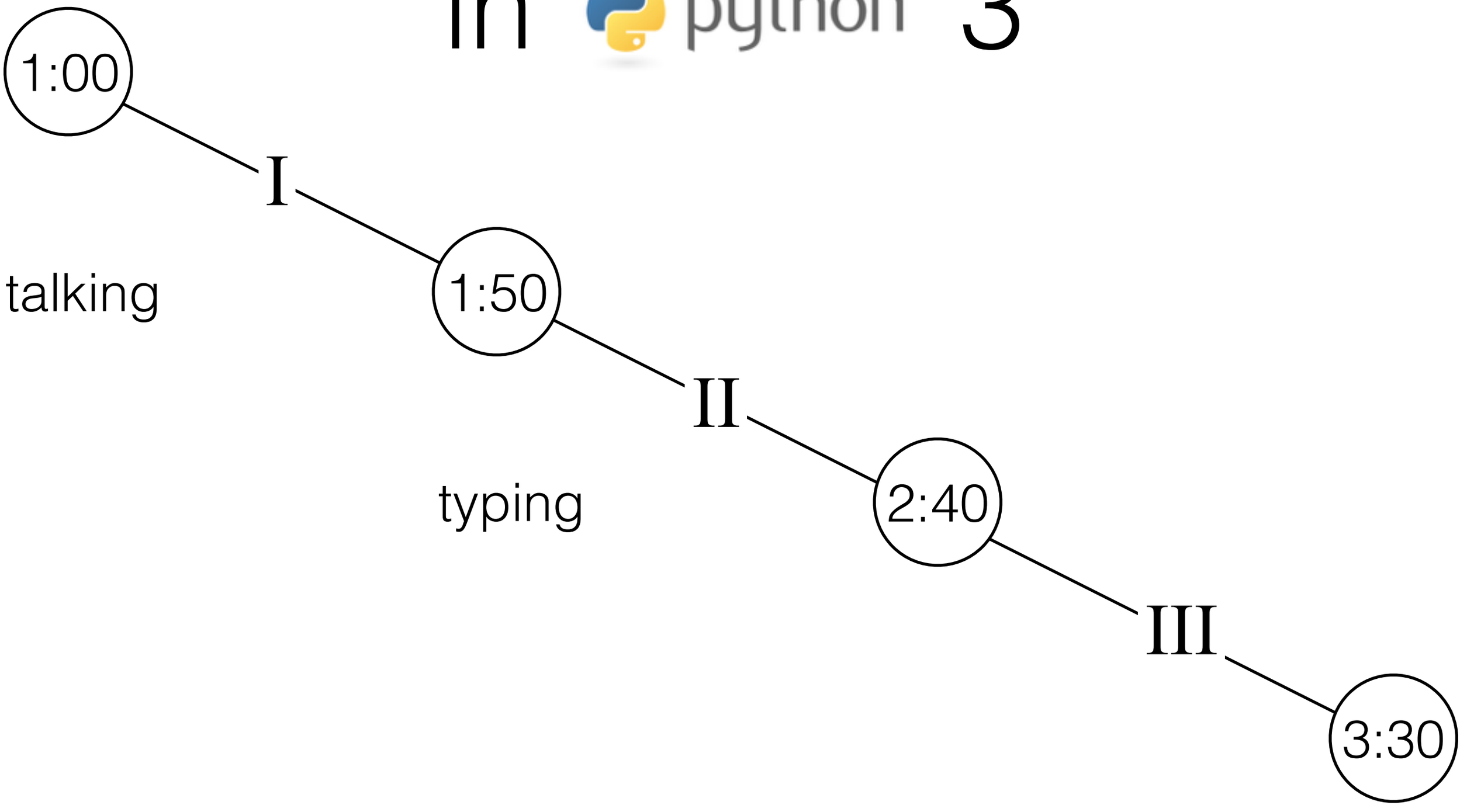
Write a Toy Language in python™ 3



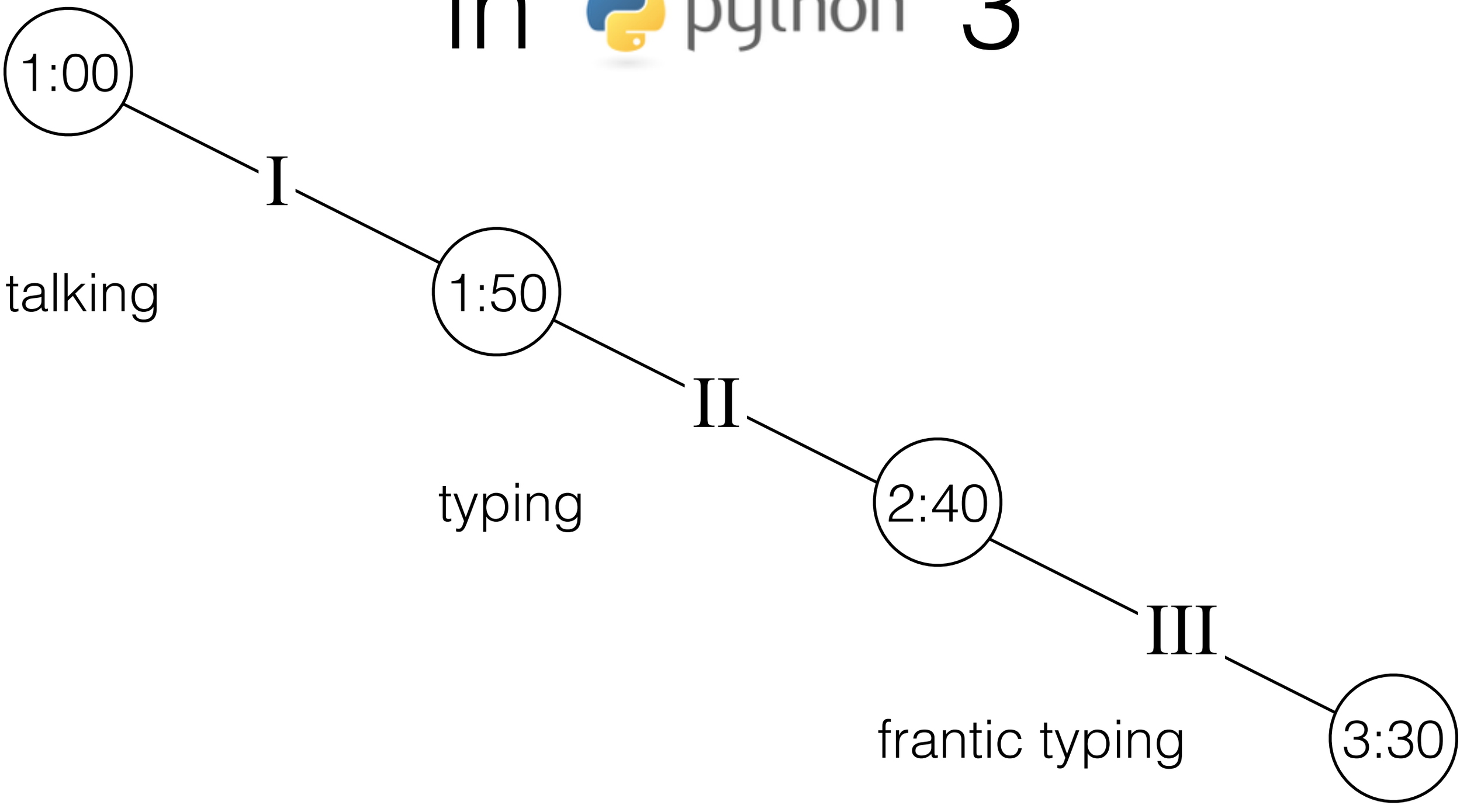
Write a Toy Language in python™ 3



Write a Toy Language in python™ 3



Write a Toy Language in python™ 3



Write a Toy Language in python™ 3

1:00

I

1:50

“An introduction to programming language concepts using Roman Numerals.”

Write a Toy Language in python™ 3

1:50

Writing a toy language: romnomnom

II

“A simple toy language and a dive into the Python AST module.”

2:40

```
$ python3 romnomnom
> I
1
> IV
4
> MMXV
2015
>
```

Write a Toy Language in python™ 3

2:40

III

3:30

Writing a toy language: pypethon

“A less simple toy language and a deeper dive into the Python AST module.”

```
$ python3 pypethon
> 
> = answer -41
> 
> answer
-41
> 
> |= correct abs | inc
> 
> answer | correct
42
```

Write a Toy Language in  3

Language Concepts

PART

I

Write a Toy Language in  3

Write a Toy Language in  3

why?

Template Languages

CSS Preprocessors

Data Serialization

Syntax Highlighters

Programming Languages

Or, write your own!

wheresaddie / **Emojinal**

👋🌎

sferik / **active_emoji**

```
👆 = 🔟.➖ 🔟
☝ = ✊.▶️
✌ = ☝.▶️
👌 = ✌.▶️
✋ = 👌.➕ ✌
☎ = -> (👂) { 📩 👂 % ✋ == ✊ && 👂 % 👌 == ✊ ? "🍺
🐝" : 👂 % 👌 == ✊ ? "🍺" : 👂 % ✋ == ✊ ? "🐝" : 👂 }
☝.👆 (💯) { |👄| ☎.📞(👄) }
```

wheresaddie / **Emojinal**

Hello World

👋🌎

sferik / **active_emoji**

✊ = 🔟.➖ 🔟

☝ = ✊.▶️

✌ = ☝.▶️

👌 = ✌.▶️

✋ = 👌.➕ ✌

☎ = -> (👂) { 📩 👂 % ✋ == ✊ && 👂 % 👌 == ✊ ? "🍺🐝" : 👂 % 👌 == ✊ ? "🍺" : 👂 % ✋ == ✊ ? "🐝" : 👂 }

☝.👆 (💯) { |👄| ☎.📞(👄) }

wheresaddie / **Emojinal**

Hello World

👋🌎

sferik / **active_emoji**

Fizz Buzz

```
✊ = 🔟.➖ 🔟
☝ = ✊.▶️
✌ = ☝.▶️
👌 = ✌.▶️
✋ = 👌.➕ ✌
☎ = -> (👂) { 📩 👂 % ✋ == ✊ && 👂 % 👌 == ✊ ? "🍺
🐝" : 👂 % 👌 == ✊ ? "🍺" : 👂 % ✋ == ✊ ? "🐝" : 👂 }
☝.👆 (💯) { |👄| ☎.📞(👄) }
```

cool.

except ...

$$1+2+3+4+5+6\ldots\ =$$

$$1+2+3+4+5+6\ldots = -1/12$$

helpful > correct

language

start here

Write a Toy Language in  3

WIKIPEDIA
The Free Encyclopedia

Create account Log in

Article Talk Read Edit View history Search

Toy language

From Wikipedia, the free encyclopedia

"… a computer programming language that is not considered to fulfill the robustness or completeness requirement of a computer programming language."

"… a learning or demonstration tool …"

WIKIPEDIA
The Free Encyclopedia

Create account Log in

Article Talk Read Edit View history Search

Programming language

From Wikipedia, the free encyclopedia

"A programming language is a notation for writing programs …"

"There are, broadly, two approaches to programming language implementation …"

meaning: implement

Write a Toy Language in python™ 3

WIKIPEDIA
The Free Encyclopedia

Create account Log in

Article Talk Read Edit View history Search

Programming language implementation

From Wikipedia, the free encyclopedia

"… a system for executing programs written in a programming language.

There are two general approaches to programming language implementation:

Interpretation: An *interpreter* takes as input a program in some language, and performs the actions written in that language …

Compilation: A *compiler* takes as input a program in some language, and translates that program into some other language …

… a compiler does not directly execute the program. Ultimately, in order to execute a program via compilation, it must be translated into a form that can serve as input to an interpreter.

WIKIPEDIA
The Free Encyclopedia

Create account Log in

Article Talk Read Edit View history Search

Programming language implementation

From Wikipedia, the free encyclopedia

"… a system for executing programs written in a programming language.

There are two general approaches to programming language implementation:

Interpretation: An *interpreter* takes as input a program in some language, and performs the actions written in that language …

Compilation: A *compiler* takes as input a program in some language, and translates that program into some other language …

… a compiler does not directly execute the program. Ultimately, in order to execute a program via compilation, it must be translated into a form that can serve as input to an interpreter.

"… a virtual machine (VM) is an emulation of a particular computer system."

"… software running inside is limited to the resources and abstractions provided by the virtual machine …"

"[virtual machines] can serve as an abstraction layer for any computer language."

WIKIPEDIA
The Free Encyclopedia

Create account Log in

Article Talk Read Edit View history Search

Virtual machine

From Wikipedia, the free encyclopedia

"… a virtual machine (VM) is an emulation of a particular computer system."

"… software running inside is limited to the resources and abstractions provided by the virtual machine …"

"[virtual machines] can serve as an abstraction layer for any computer language."

Write a Toy Language in  3

New programming languages targeting the JVM, JavaScript, and others, are popping up all over the place. In a couple of hours, we'll cover the basics of writing your own language in Python 3 to target the Python virtual machine.

Write a Toy Language in  3

New programming languages targeting the JVM, JavaScript, and others, are popping up all over the place. In a couple of hours, we'll cover the basics of writing your own language in Python 3 to target the Python virtual machine.

that was an important slide.


python™

WIKIPEDIA
The Free Encyclopedia

Create account Log in

Article Talk Read Edit View history Search

Read–eval–print loop

From Wikipedia, the free encyclopedia

"… a simple, interactive computer programming environment that takes single user inputs, evaluates them, and returns the result to the user"

"REPLs facilitate exploratory programming …"

WIKIPEDIA
The Free Encyclopedia

Create account Log in

Article Talk Read Edit View history Search

Read–eval–print loop

From Wikipedia, the free encyclopedia

"… a simple, interactive computer programming environment that takes single user inputs, evaluates them, and returns the result to the user"

"REPLs facilitate exploratory programming …"

```
$ python3
>>>
```

```
$ python3
>>> 42
```

```
$ python3
>>> 42
42
```

```
$ python3
>>> 42
42
```

Graphic image of exploding head goes here!

```
$ python3
>>> 42
42
```

1. We wrote a program in the Python *programming language:* "`42`".

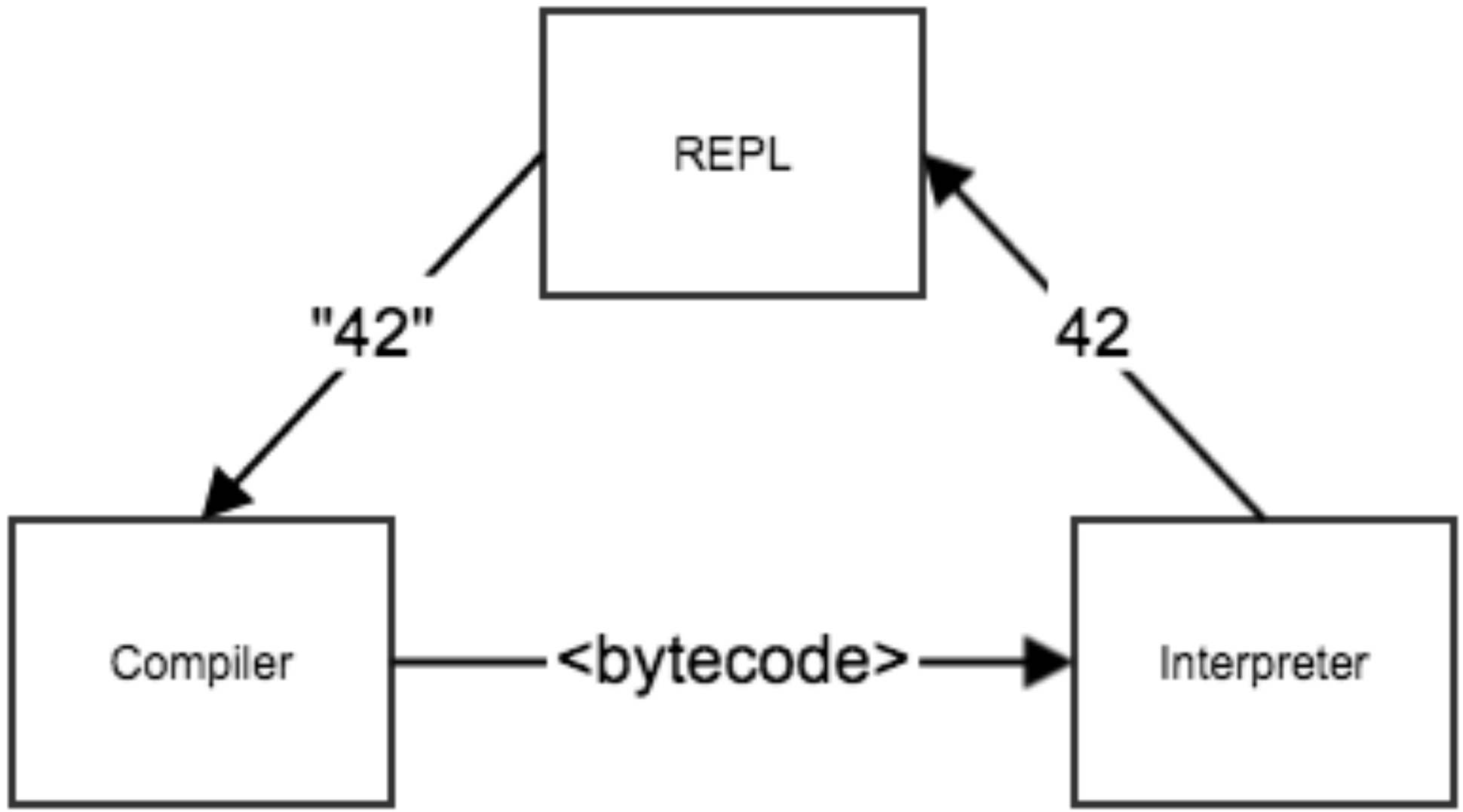
2. Which was submitted via the *REPL* to the *CPython programming language implementation*.

3. Which *compiled* the program into another language: Python *bytecode*.

4. Which the *interpreter* evaluated to a value: `42`.

5. Which the *REPL* printed a representation of back to the programmer.

so far

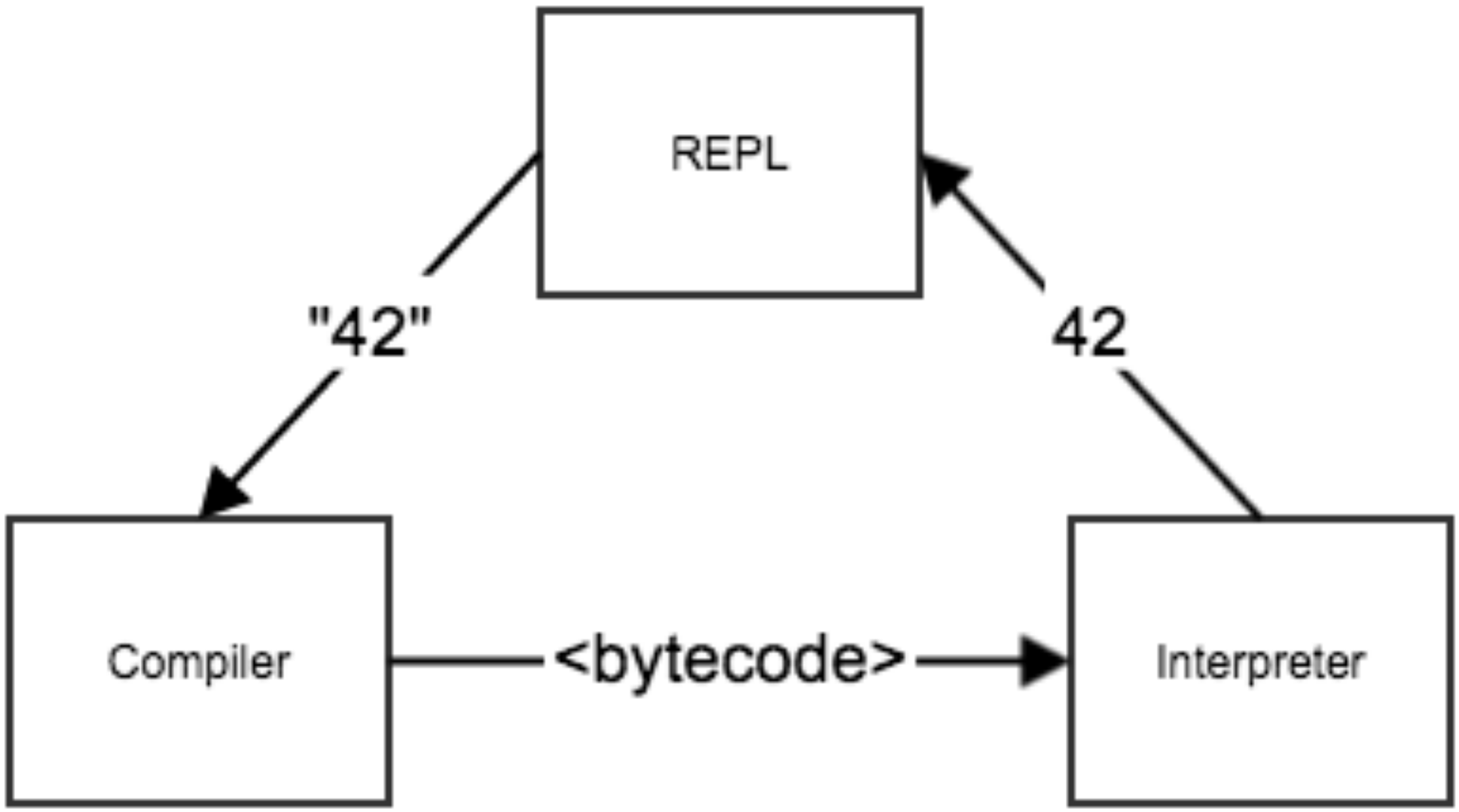

REPL
"42"
42
Compiler
<bytecode>
Interpreter

```
$ python3
>>>
>>> while True:
...     print(eval(input("> ")))
...
> "hello world"
hello world
> 1 + 2 + 3 + 4 + 5
15
> [i**2 for i in range(1, 5)]
[1, 4, 9, 16]
```

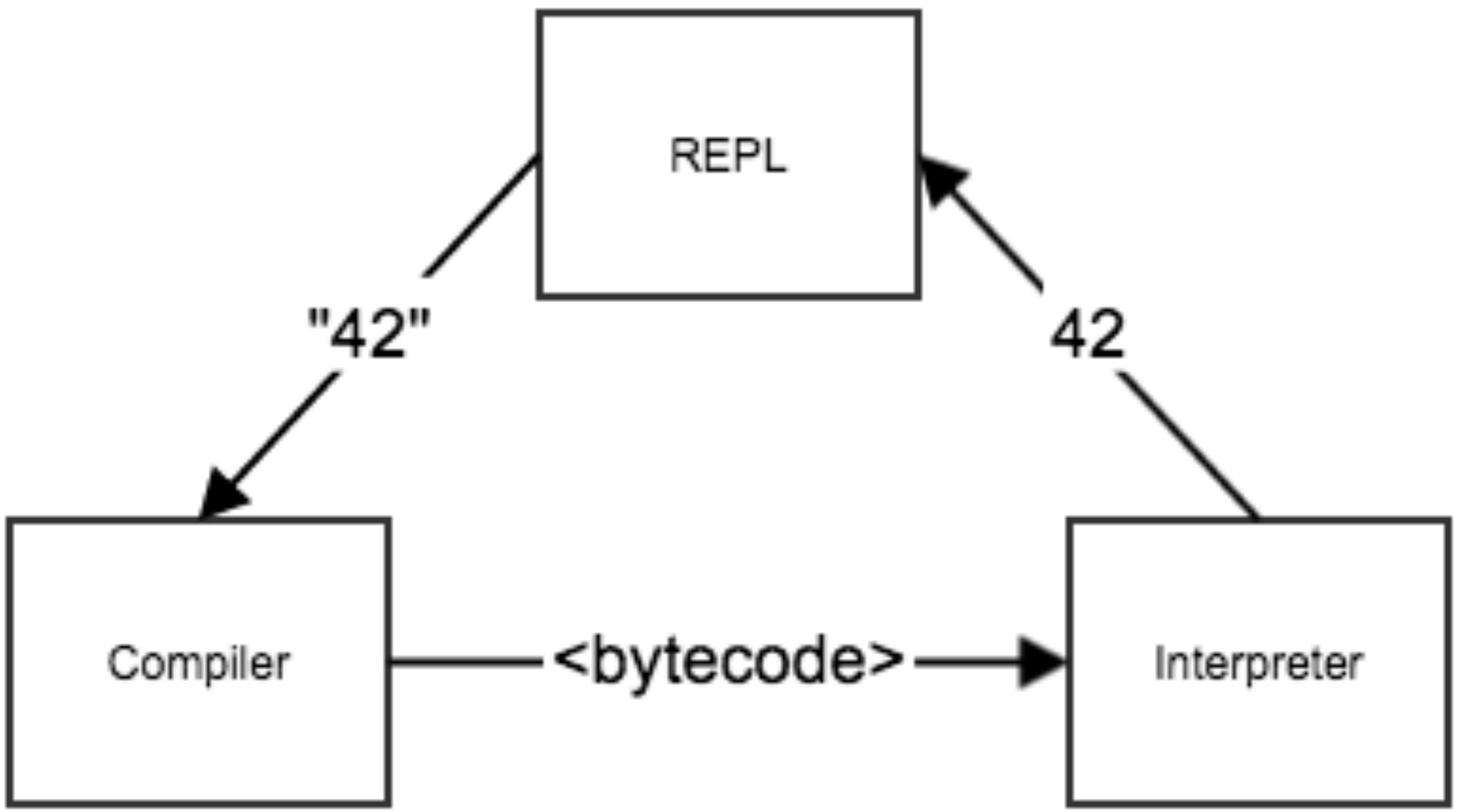
```
$ python3
>>> from code import interact
>>> interact("My REPL",
...     lambda p: input("> "))
>
> “hello world”
hello world
> 1 + 2 + 3 + 4 + 5
15
> [i**2 for i in range(1, 5)]
[1, 4, 9, 16]
```

Write a Toy Language in  3

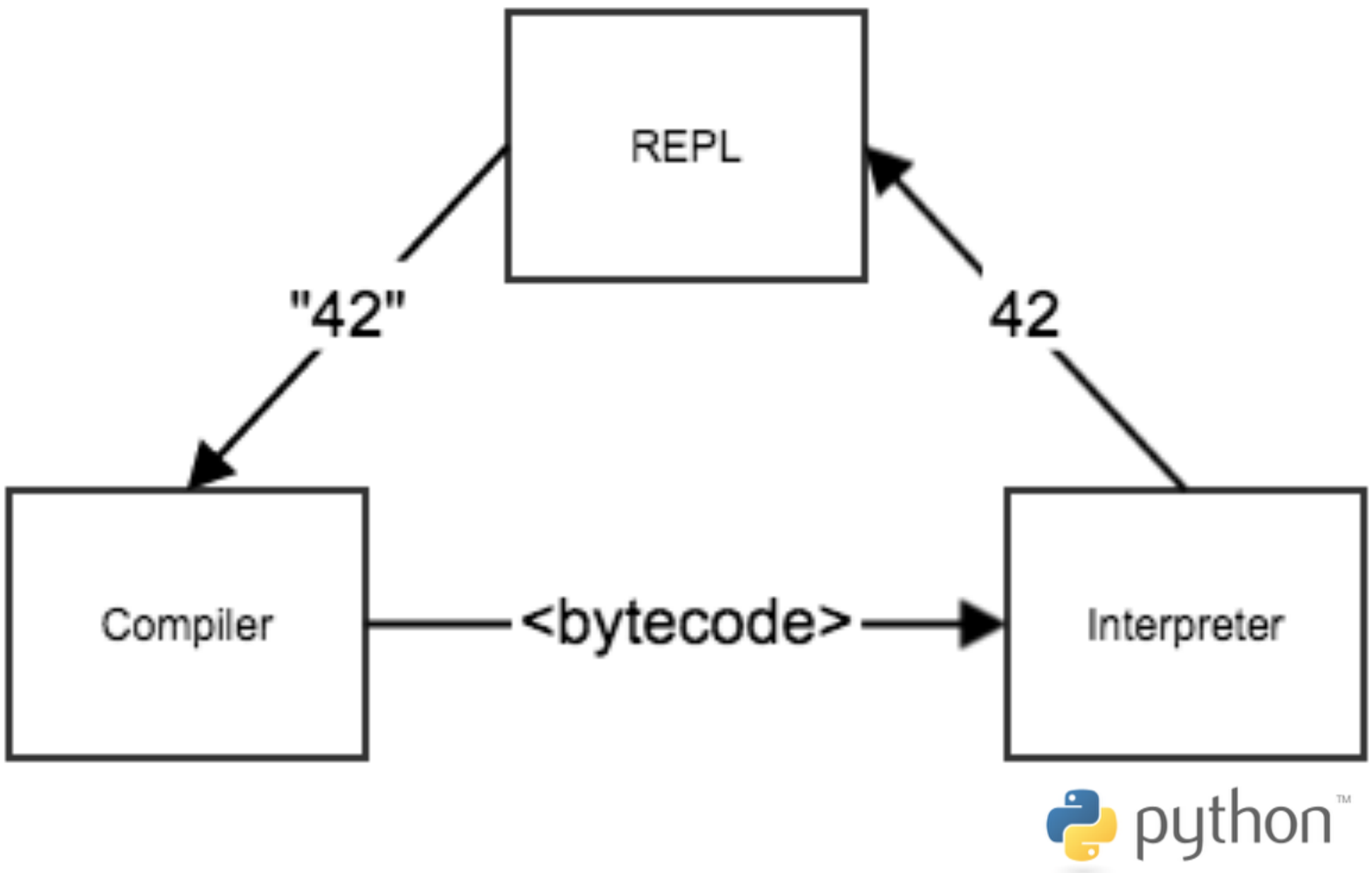
```
>>> while True:
...     print(eval(input("> ")))
```



```
>>> while True:
...     print eval(raw_input("> ")))
```



```
>>> while True:
...     print eval(raw_input("> ")))
```



WIKIPEDIA
The Free Encyclopedia

Create account Log in

Article Talk Read Edit View history Search

High-level programming language

From Wikipedia, the free encyclopedia

"… a programming language with strong abstraction from the details of the computer."

"When a language is compiled to an intermediate representation, that representation can be optimized or saved for later execution without the need to re-read the source file. When the intermediate representation is saved, it is often represented as byte code."

"… a form of instruction set designed for efficient execution by a software interpreter."

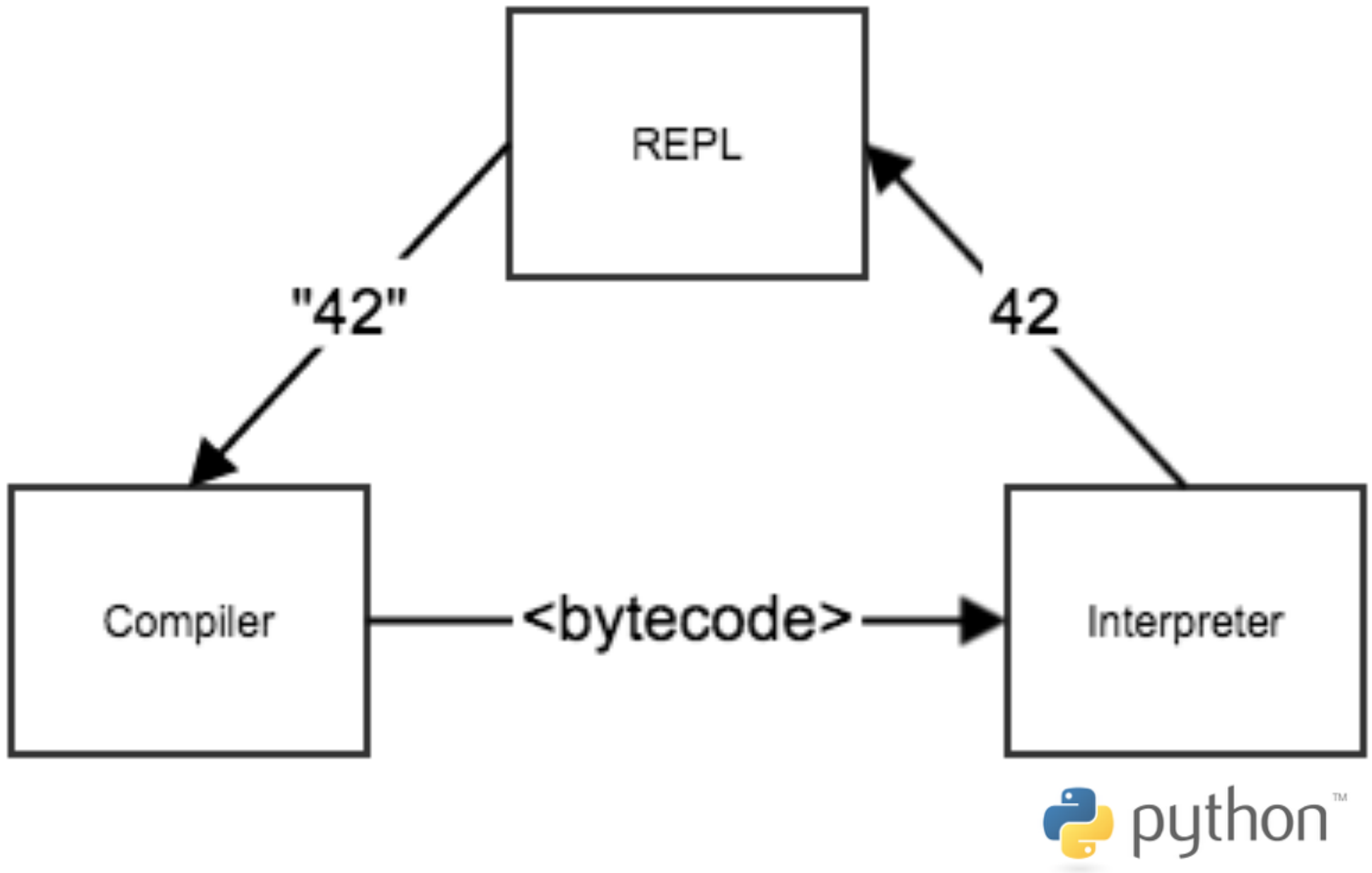
"They therefore allow much better performance than direct interpretation of source code."

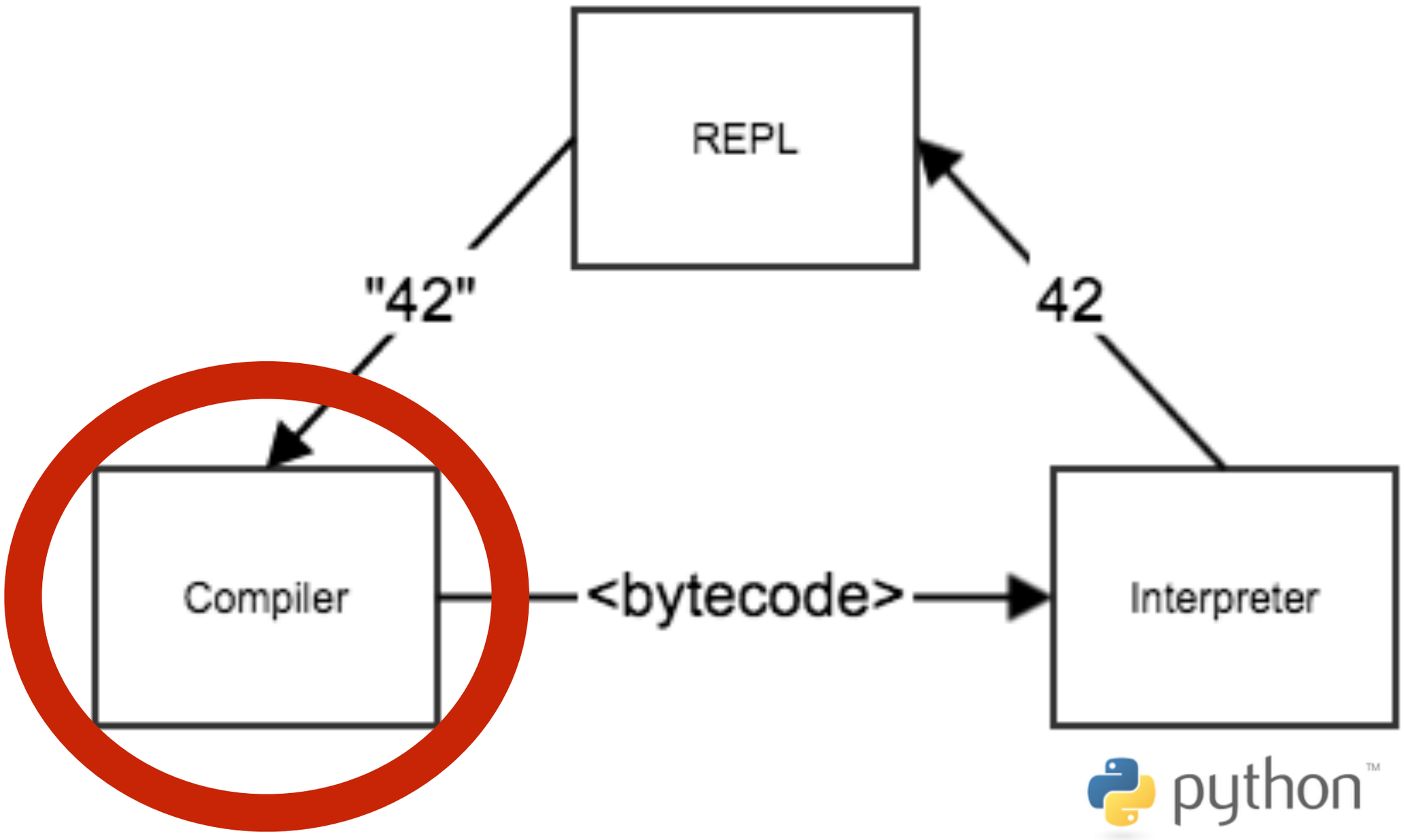
“See Python Run”

Sunday 2:40-3:30

–Jason Orendorff

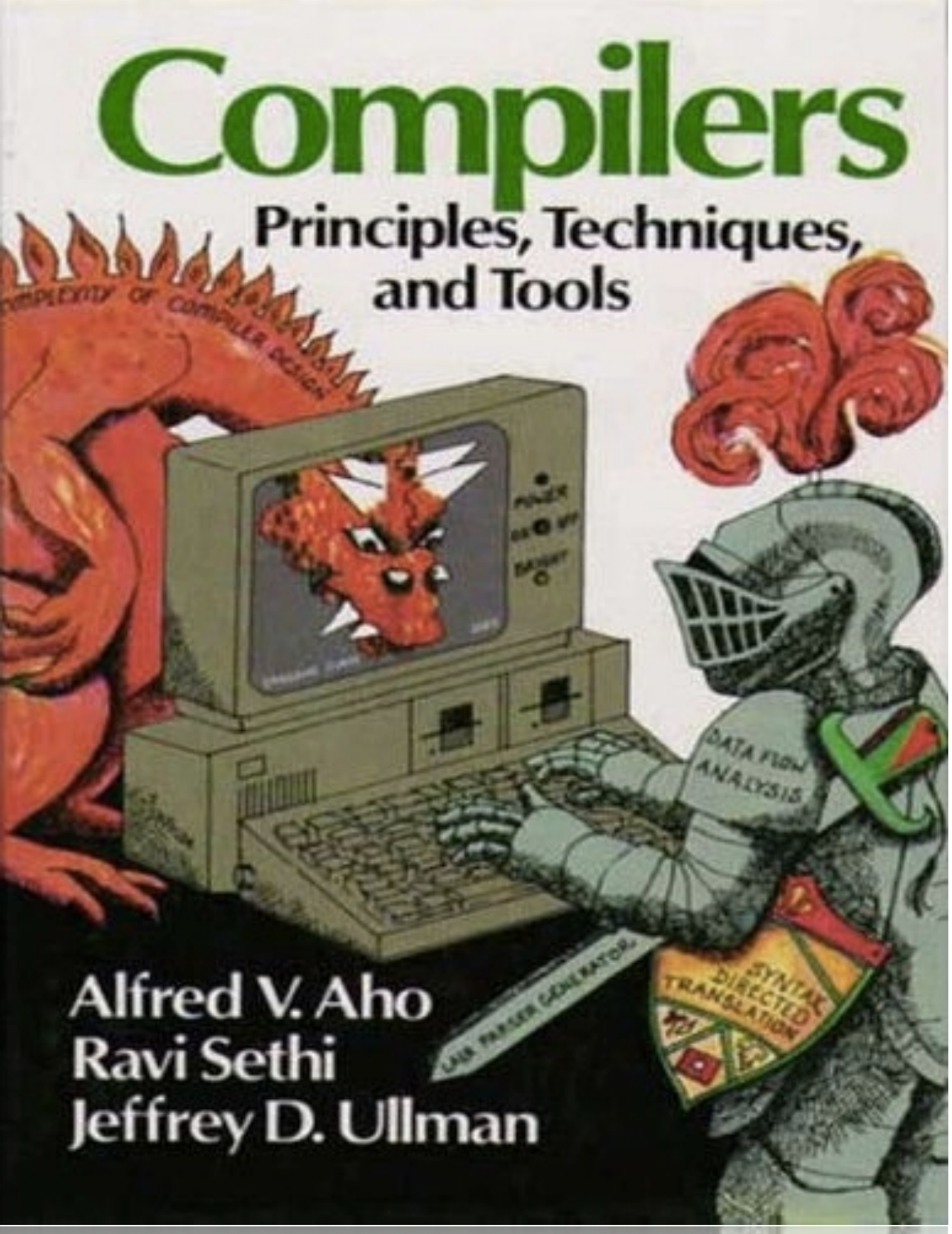
https://www.pytennessee.org/schedule/presentation/68/


REPL
"42"
42
Compiler
<bytecode>
Interpreter
python™


REPL
"42"
42
Compiler
<bytecode>
Interpreter
python™

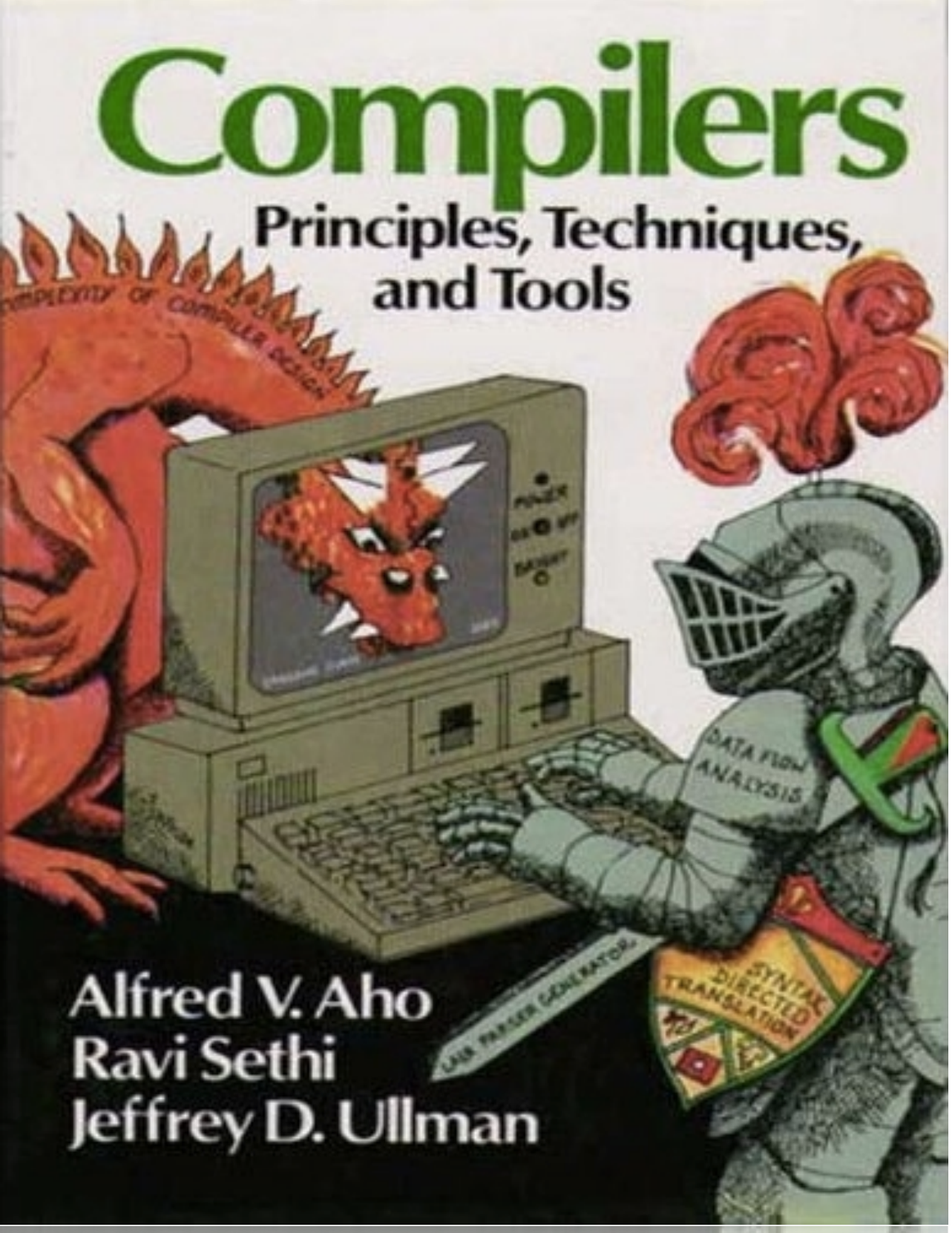
compilers

http://compilers.pydata.org/



"Simply stated, a compiler is a program that reads a program written in one language - the ***source*** language - and translates it into an equivalent program in another language - the ***target*** language."

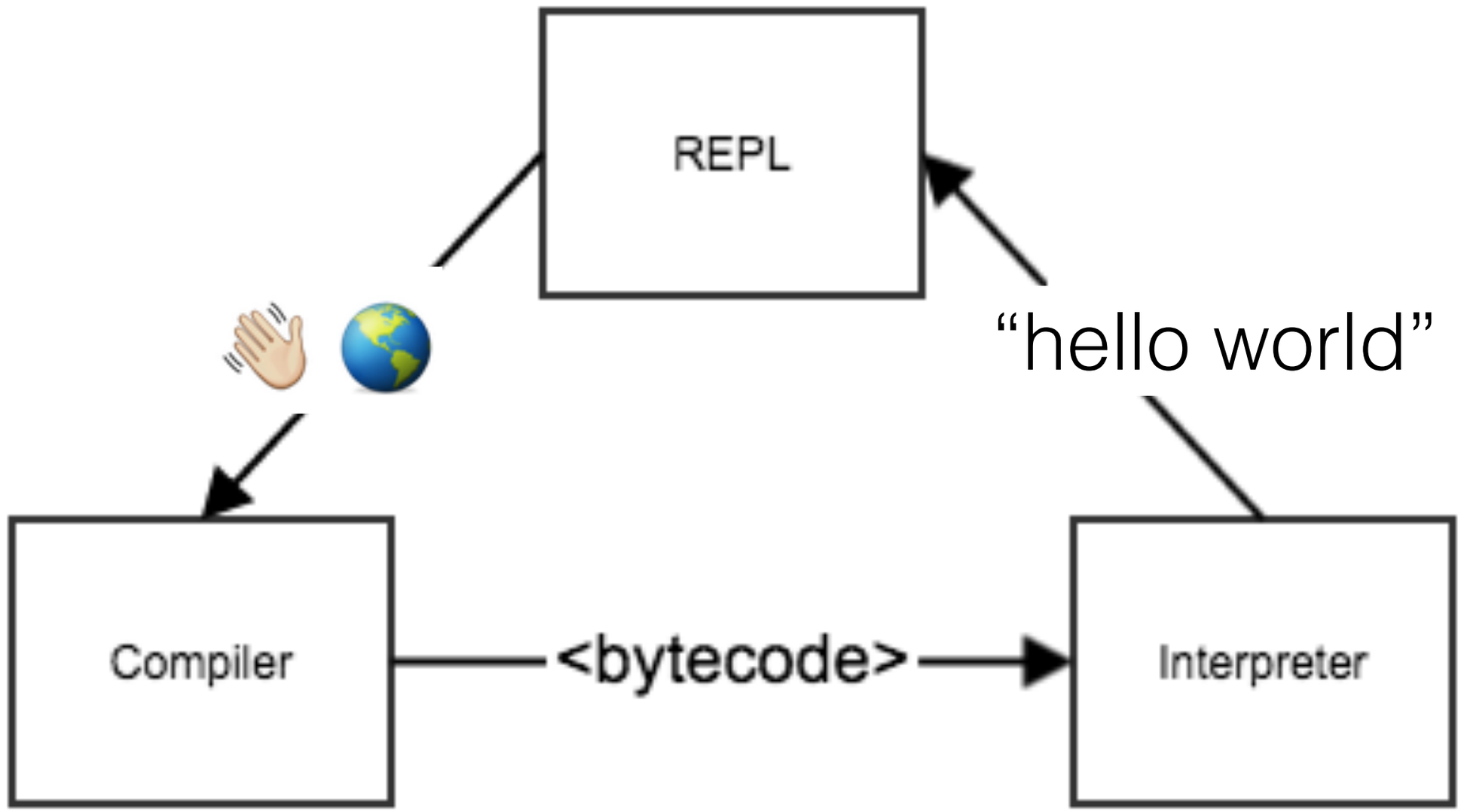
- Compilers
Principles, Techniques, and Tools



"Simply stated, a compiler is a program that reads a program written in one language - the ***source*** language - and translates it into an equivalent program in another language - the ***target*** language."

- Compilers
Principles, Techniques, and Tools

that was an important slide.


REPL
“hello world”
Compiler
<bytecode>
Interpreter

- Compilers
Principles, Techniques, and Tools

language – the *target* language (see Fig. 1.1). As an impo
anslation process, the compiler reports to its user the pr
the source program.

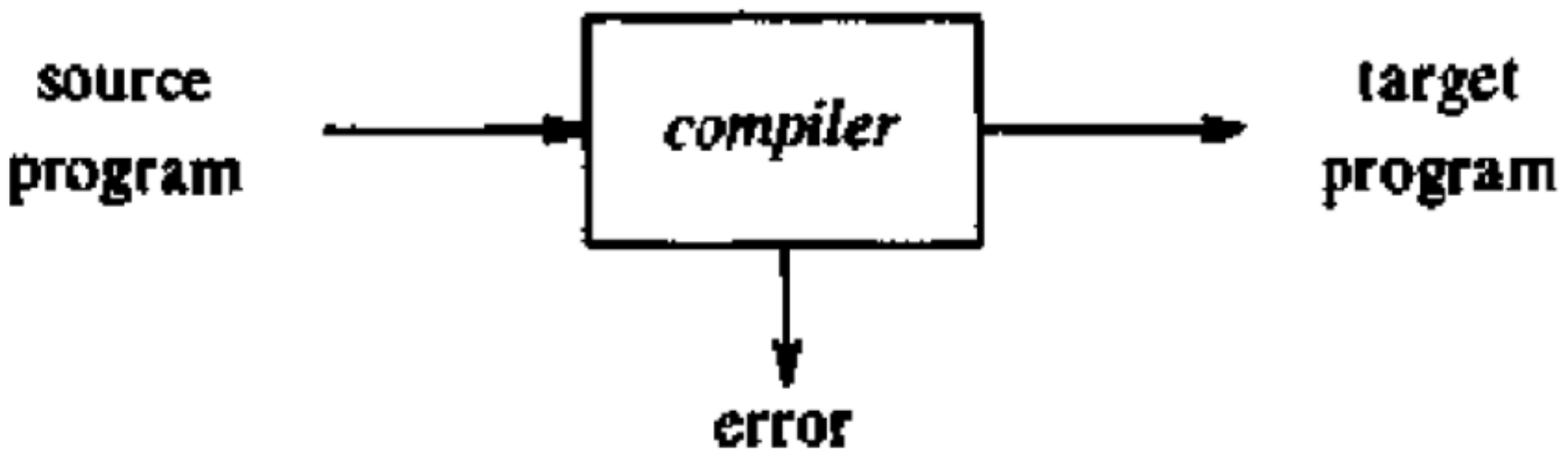


Fig. 1.1. A compiler.

- Compilers
Principles, Techniques, and Tools

t glance, the variety of compile
sands of source languages, ranging from traditional pro

- Compilers
Principles, Techniques, and Tools

- Compilers
Principles, Techniques, and Tools

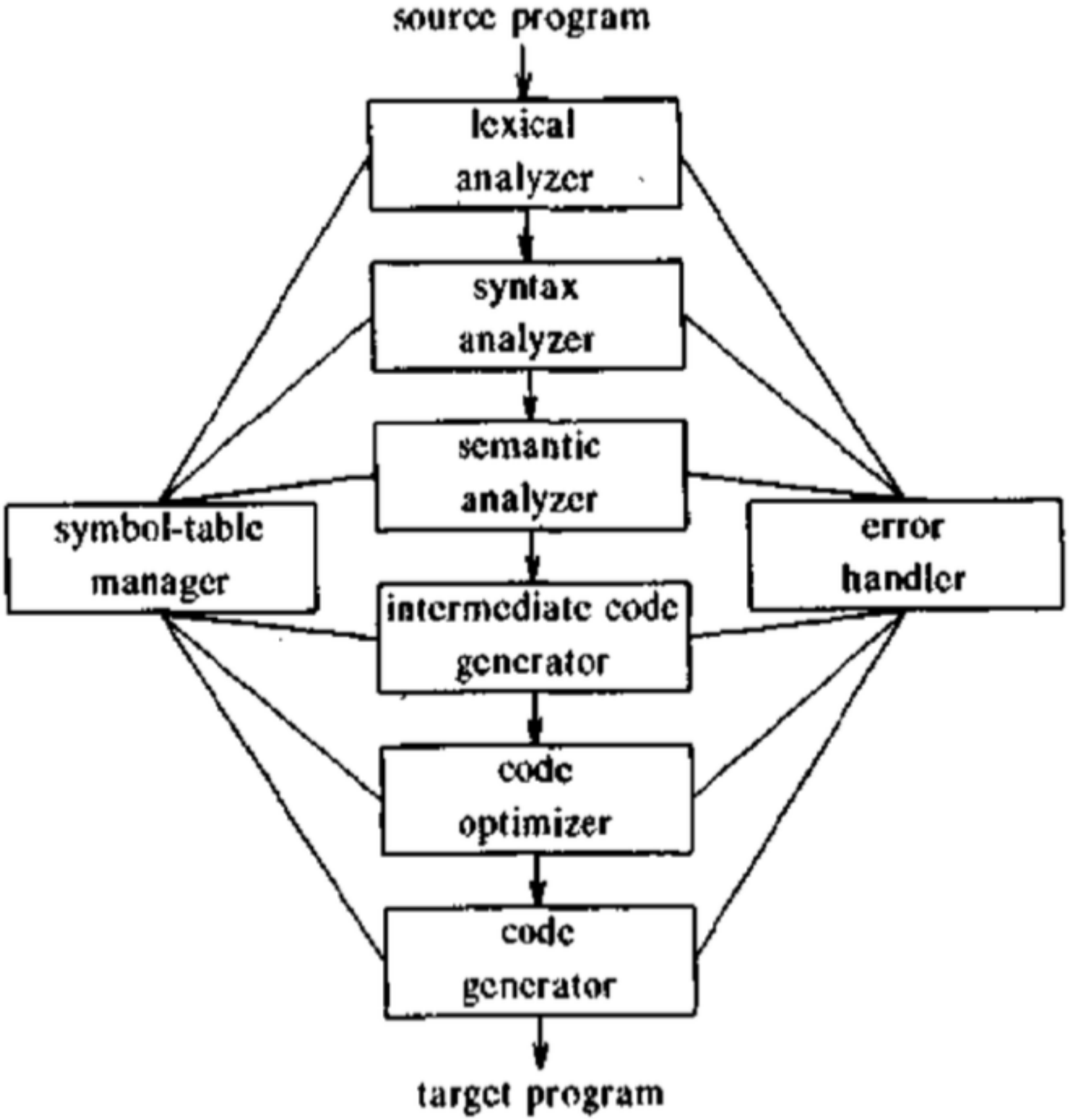


Fig. 1.9. Phases of a compiler.

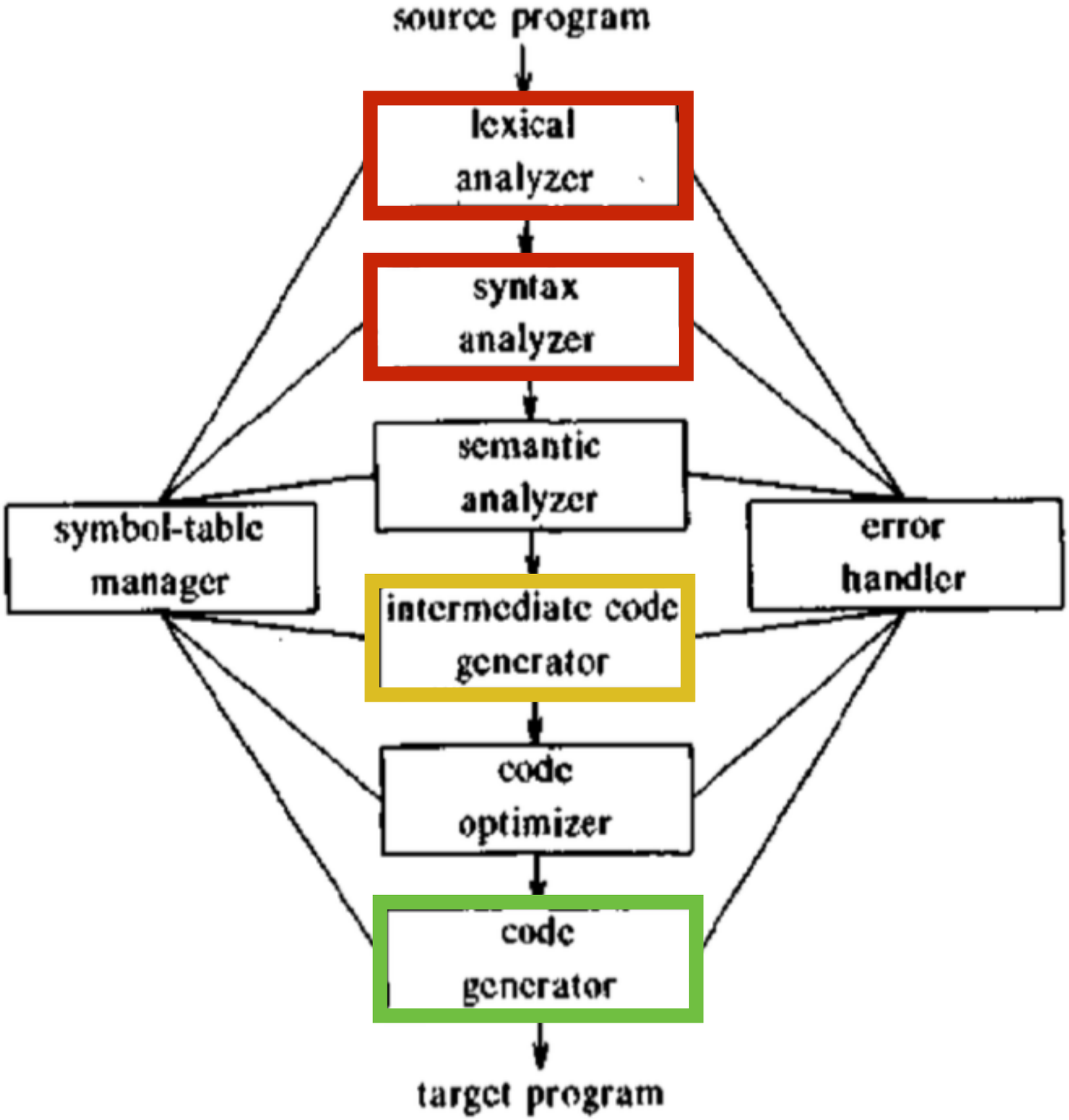


Fig. 1.9. Phases of a compiler.

Most of *our* work

Front End

Python compile()

Back End

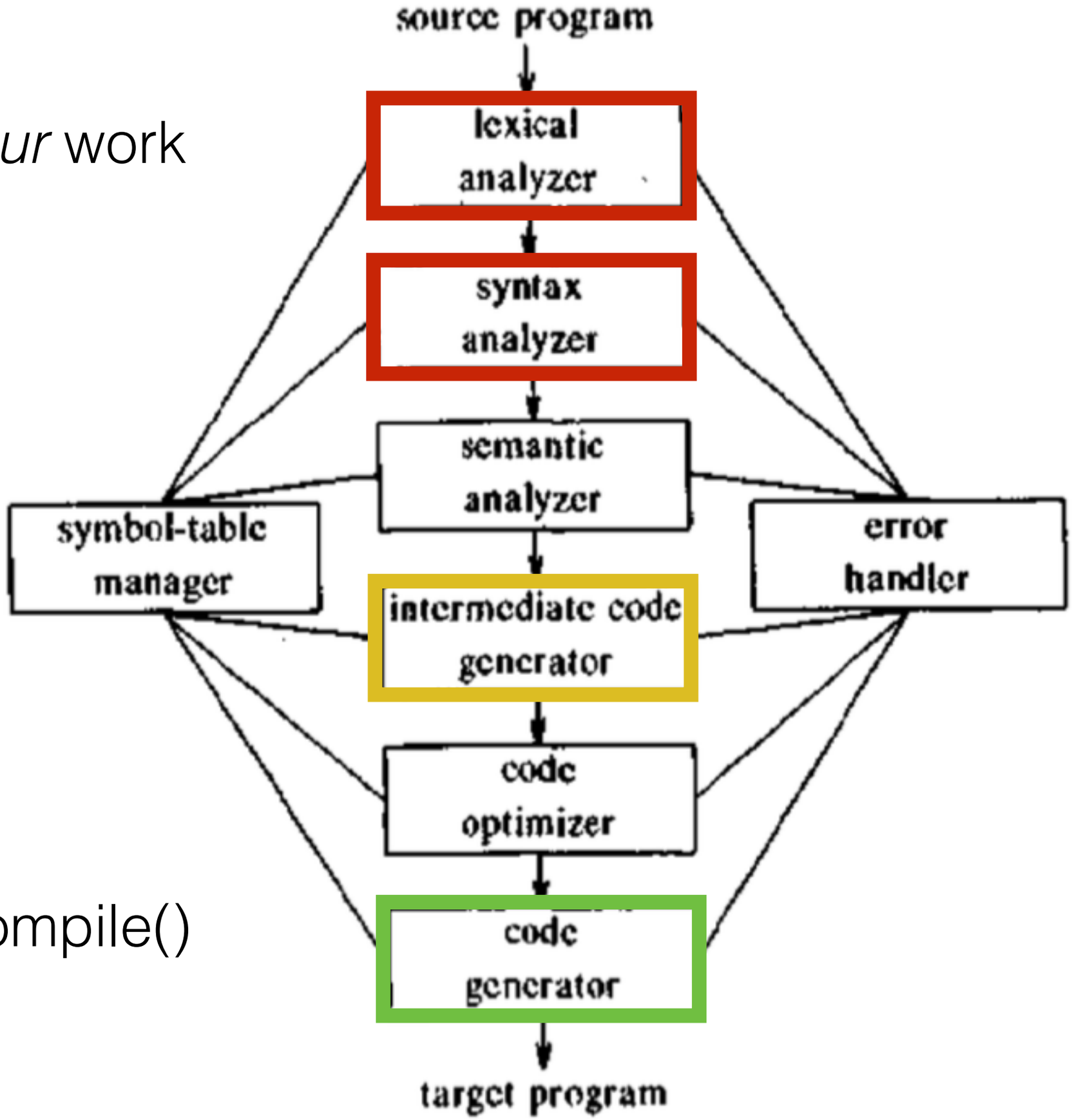


Fig. 1.9. Phases of a compiler.

- Compilers
Principles, Techniques, and Tools

"The first phase of a compiler is called lexical analysis. This phase involves grouping the characters that make up the source program into meaningful sequences [of tokens]."

"The second phase of constructing a compiler is syntax analysis. … which shows the grammatical structure of the tokens. Syntax analysis is also called parsing."

WIKIPEDIA
The Free Encyclopedia

Create account Log in

Article Talk Read Edit View history Search

Compiler construction

From Wikipedia, the free encyclopedia

`print("hello")` → Lexer →

Parser →

WIKIPEDIA
The Free Encyclopedia

Create account Log in

Article Talk Read Edit View history Search

Compiler construction

From Wikipedia, the free encyclopedia

`print("hello")` → Lexer →

`print`

→ Parser →

Compiler construction

From Wikipedia, the free encyclopedia

`print("hello")` → Lexer → `(` `print`

→ Parser →

WIKIPEDIA
The Free Encyclopedia

Create account Log in

Article Talk Read Edit View history Search

Compiler construction

From Wikipedia, the free encyclopedia

```
print("hello")
```

Lexer

```
"hello"
(
print
```

Parser

WIKIPEDIA
The Free Encyclopedia

Create account Log in

Article Talk Read Edit View history Search

Compiler construction

From Wikipedia, the free encyclopedia

`print("hello")` → Lexer →

```
)
"hello"
(
print
```

Parser →

WIKIPEDIA
The Free Encyclopedia

Create account Log in

Article Talk Read Edit View history Search

Compiler construction

From Wikipedia, the free encyclopedia

`print("hello")` → Lexer →

`)`
`"hello"`
`(`
`print`

Parser →

`print`

WIKIPEDIA
The Free Encyclopedia

Create account Log in

Article Talk Read Edit View history Search

Compiler construction

From Wikipedia, the free encyclopedia

```
print("hello")
```

Lexer

```
)
"hello"
(
print
```

Parser

```
   (
print
```

Create account Log in

Article Talk Read Edit View history Search

Compiler construction

From Wikipedia, the free encyclopedia

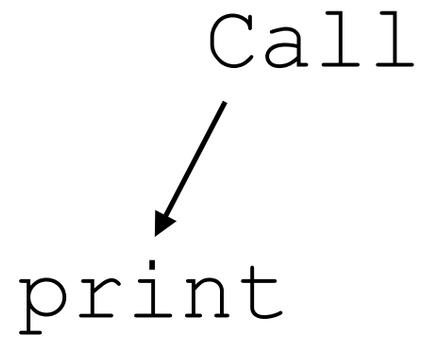
```
print("hello")
```

Lexer

```
)
"hello"
(
print
```

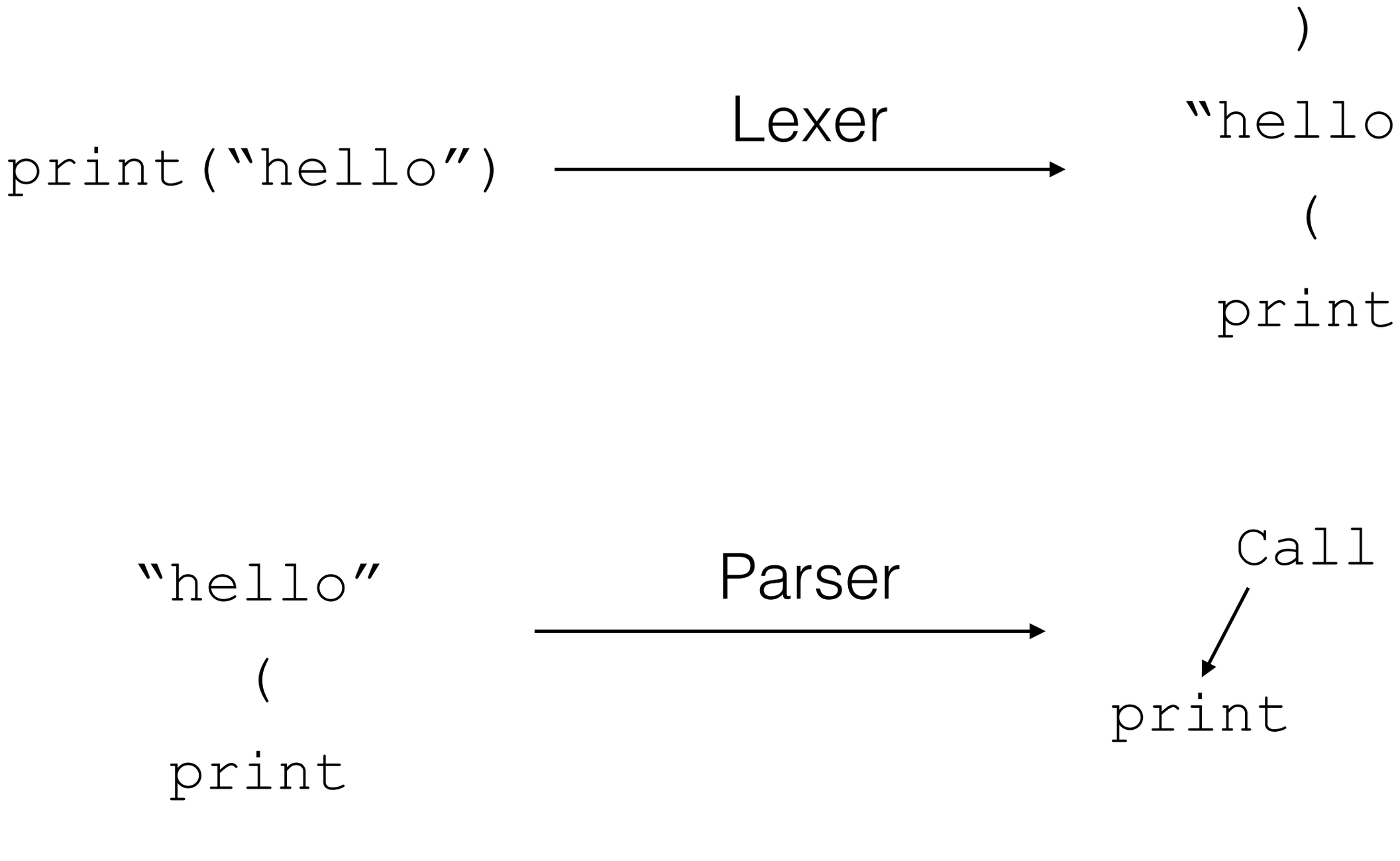
```
(
print
```

Parser



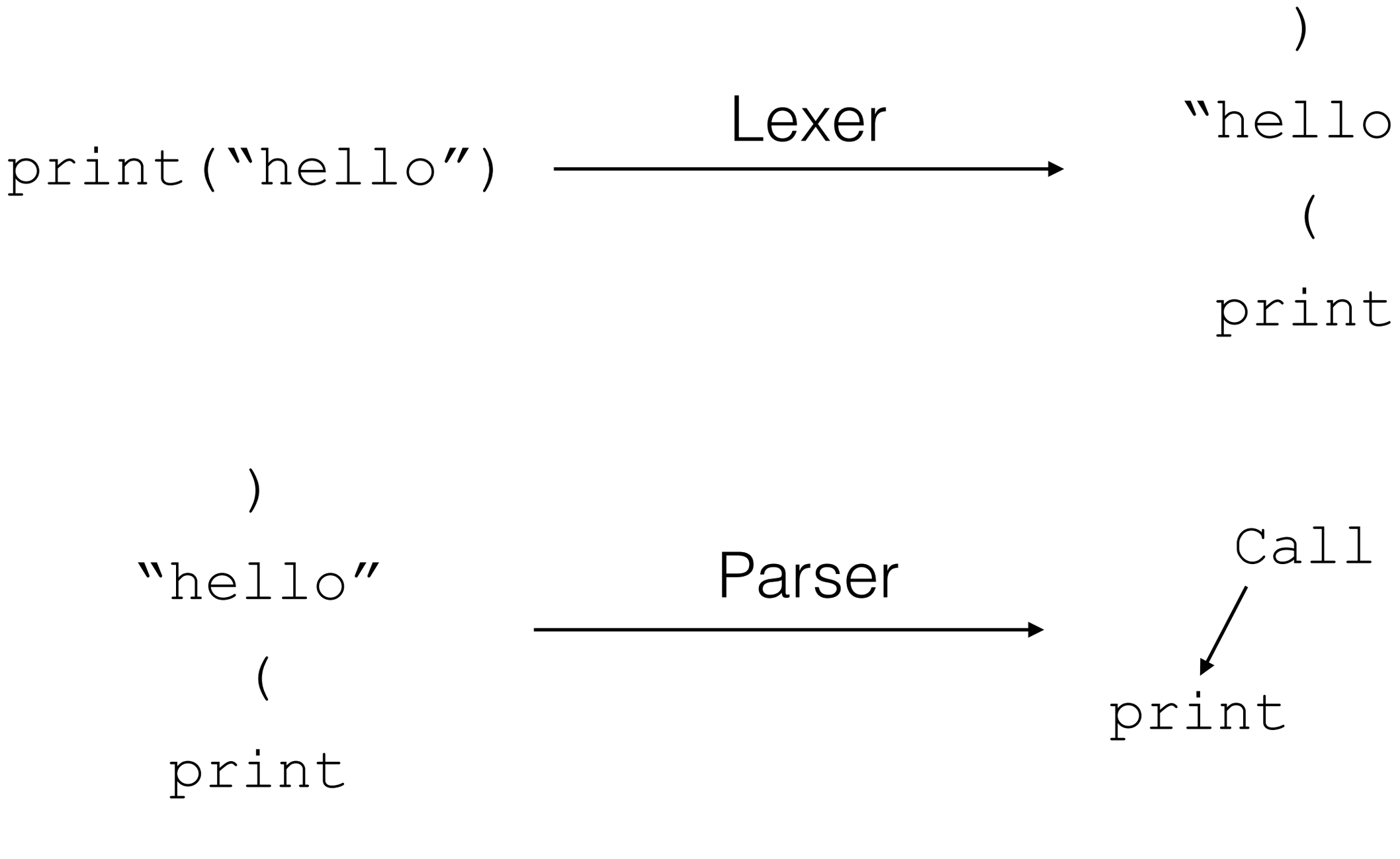
Compiler construction

From Wikipedia, the free encyclopedia



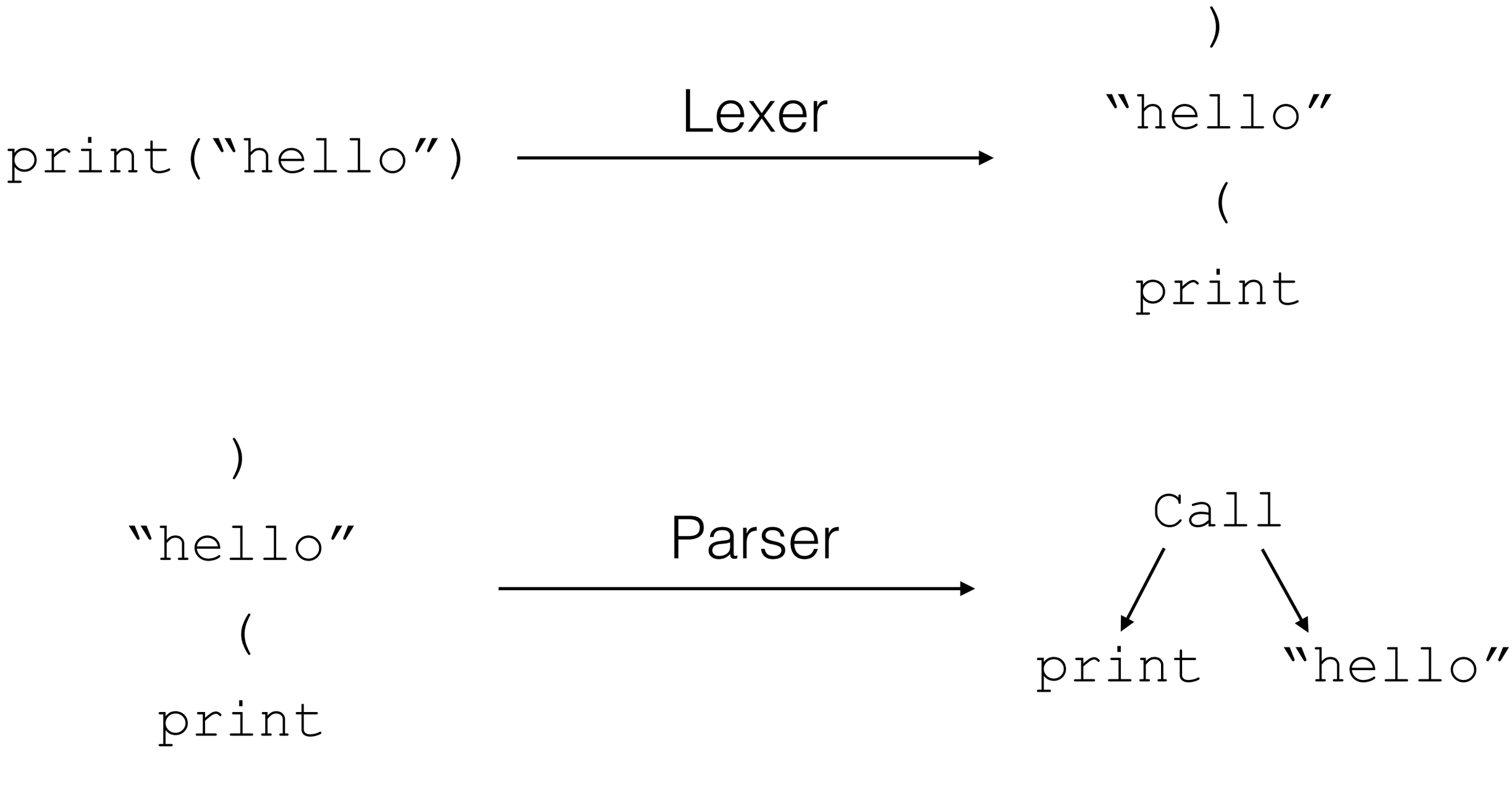
Compiler construction

From Wikipedia, the free encyclopedia



Compiler construction

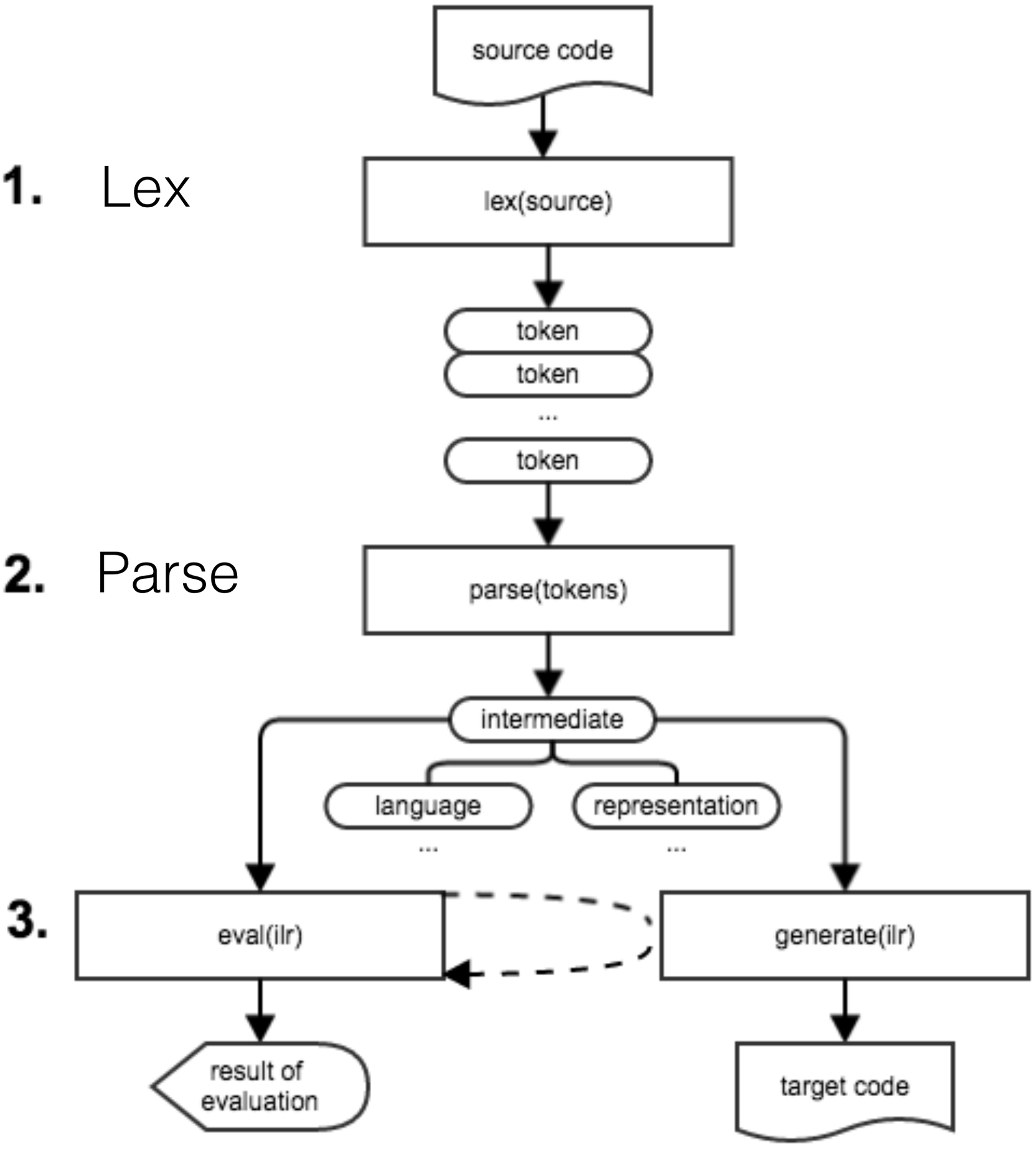
From Wikipedia, the free encyclopedia



1. Lex

2. Parse

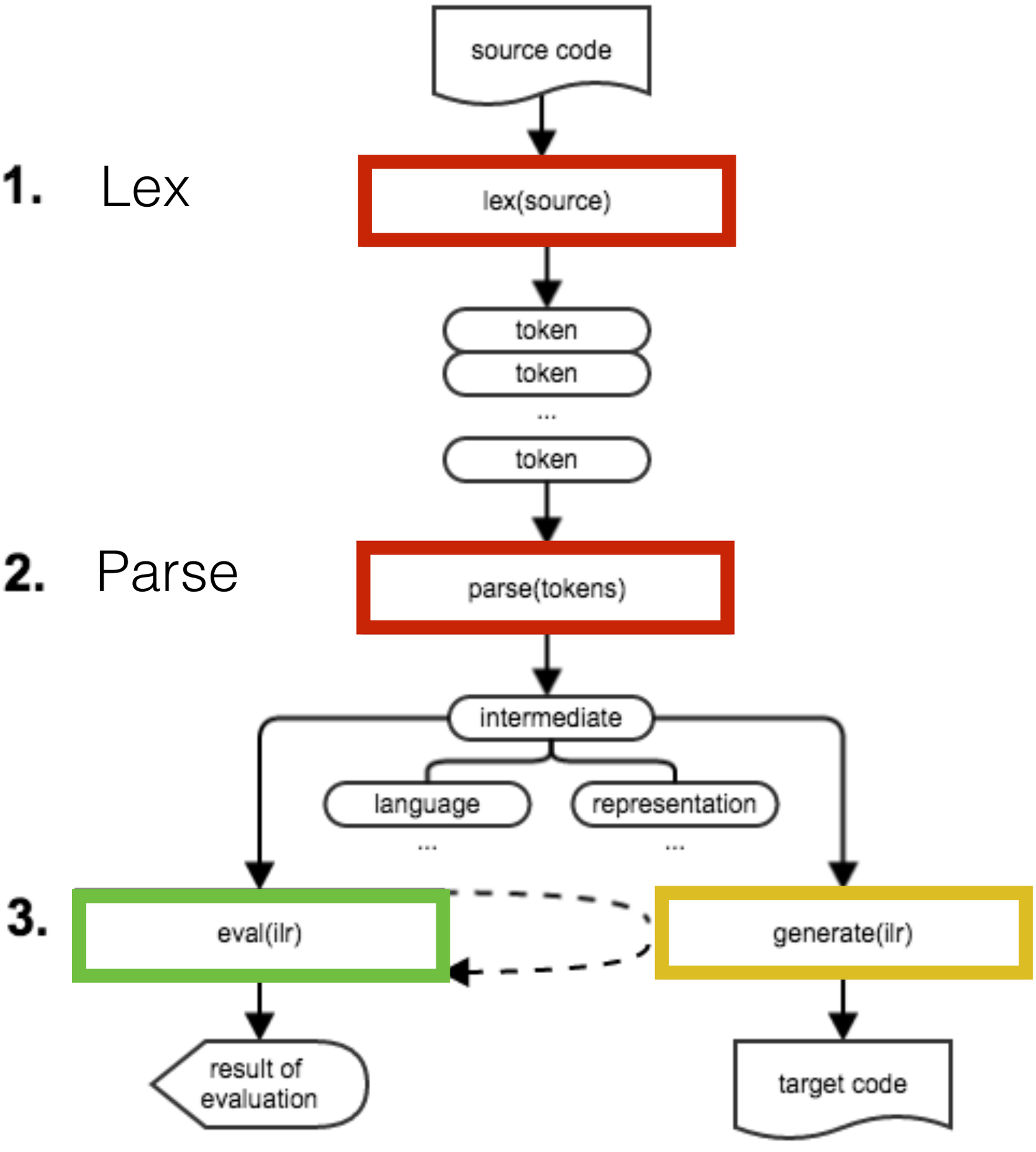
3.



1. Lex

2. Parse

3.



compiling

```
>>> import
```

`>>> import`

- `tokenize`: Interface to Python’s tokenizer.
- `parser`: Interface to Python's internal parser.
- `ast`: Process trees of the Python abstract syntax.
- `py_compile`: Compile a single .pyc file.
- `compileall`: Compile a directory of .py files.
- `dis`: Disassemble Python byte code.

`>>> import`

- `tokenize`: Interface to Python's tokenizer.
- `parser`: Interface to Python's internal parser.
- `ast`: rocess trees of the Python abstract syntax.
- `py_compile`: Compile a single .pyc file.
- `compileall`: Compile a directory of .py files.
- `dis`: Disassemble Python byte code.

“See Python Run”

Sunday 2:40-3:30

–Jason Orendorff

https://www.pytennessee.org/schedule/presentation/68/

```
>>> tokenize.generate_tokens(
  StringIO('print("Hello World")').readline)

TokenInfo(type=1 (NAME), string='print', start=(1, 0), end=(1, 5))
TokenInfo(type=52 (OP), string='(', start=(1, 5), end=(1, 6))
TokenInfo(type=3 (STRING), string='"Hello World"', start=(1, 6))
TokenInfo(type=52 (OP), string=')', start=(1, 19), end=(1, 20))
TokenInfo(type=0 (ENDMARKER), string='', start=(2, 0), end=(2, 0))

>>> ast.parse('print("Hello World")')

Module(body=[
  Expr(
    value=Call(
      func=Name(id='print',
      ctx=Load()),
      args=[Str(s='Hello World')], keywords=[], starargs=None,
      kwargs=None))])

>>> compile(ast.parse('print("Hello World")'),
  '<example>', 'exec')

<code object <module> at 0x10439c780, file "<example>", line 1>
```

examples?

time permitting

Write a Toy Language in  3

romnomnom

PART

II

```
$ python3 romnomnom
> I
1
> IV
4
> MMXV
2015
>
```

https://github.com/tsclausing/romnomnom

https://github.com/tsclausing/romnomnom

Tutorial Goals … Implement:

- **All the Numerals!** The template works for the Numeral I out of the box, but what about the rest?
- **All the rules!** The template doesn't enforce any rules (characters, token order, frequency ...).
- **Compiler Exceptions!** What should happen when a rule is broken? Can your error message point to where it happened?
- **A clean exit!** Once the REPL starts ... can we exit it gracefully when we're done?

Write a Toy Language in  3

pypethon

PART

III

```
$ python3 pypethon
> 
> = answer -41
> 
> answer
-41
> 
> |= correct abs | inc
> 
> answer | correct
42
```

state.

WIKIPEDIA
The Free Encyclopedia

Create account Log in

Article Talk Read Edit View history Search

State (computer science)

From Wikipedia, the free encyclopedia

"… a computer program stores data in variables, which represent storage locations in the computer's memory. The contents of these memory locations, at any given point in the program's execution, is called the program's state."

WIKIPEDIA
The Free Encyclopedia

Create account Log in

Article Talk Read Edit View history Search

State (computer science)

From Wikipedia, the free encyclopedia

"… a computer program stores data in variables, which represent storage locations in the computer's memory. The contents of these memory locations, at any given point in the program's execution, is called the program's state."

"… a virtual machine (VM) is an emulation of a particular computer system."

"… software running inside is limited to the resources and abstractions provided by the virtual machine …"

"All of them can serve as an abstraction layer for any computer language."

WIKIPEDIA
The Free Encyclopedia

Create account Log in

Article Talk Read Edit View history Search

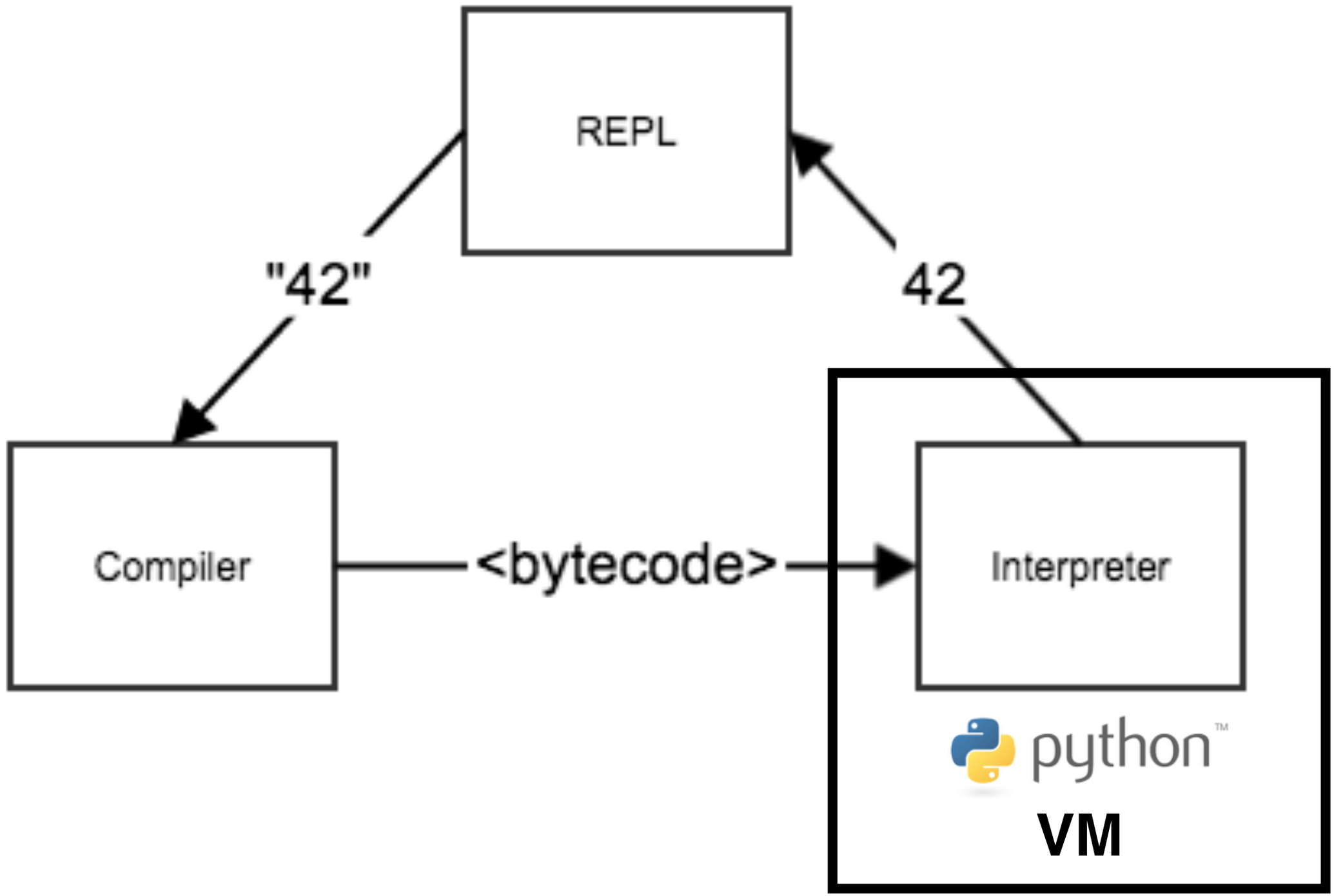
Virtual machine

From Wikipedia, the free encyclopedia

"… a virtual machine (VM) is an emulation of a particular computer system."

"… software running inside is limited to the resources and abstractions provided by the virtual machine …"

"All of them can serve as an abstraction layer for any computer language."


REPL
"42"
42
Compiler
<bytecode>
Interpreter
python™
VM

tsclausing / **pypethon**

https://github.com/tsclausing/pypethon

Write a Toy Language in  3

FIN

Write a Toy Language in  3

T. Scot Clausing

@tsclausing

FIN