cambridge medical robotics

Exploring Python ByteCode

Hands-On Session

Jan Chwiejczak

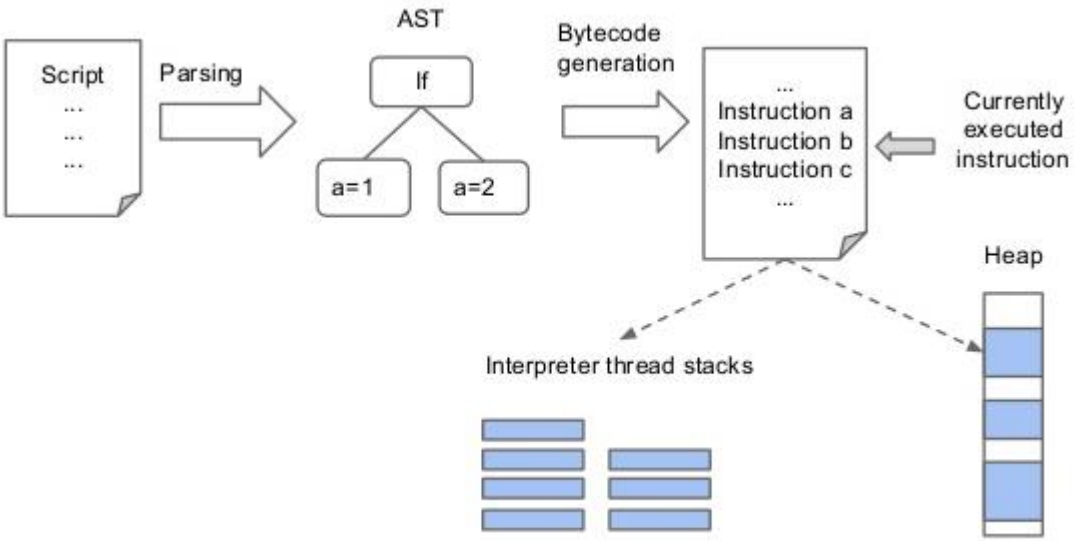
@iamjanhak

www.github.com/janhak/bytecode

What I would like to talk about today

What can you learn

- **Gain insight into how Python executes code**

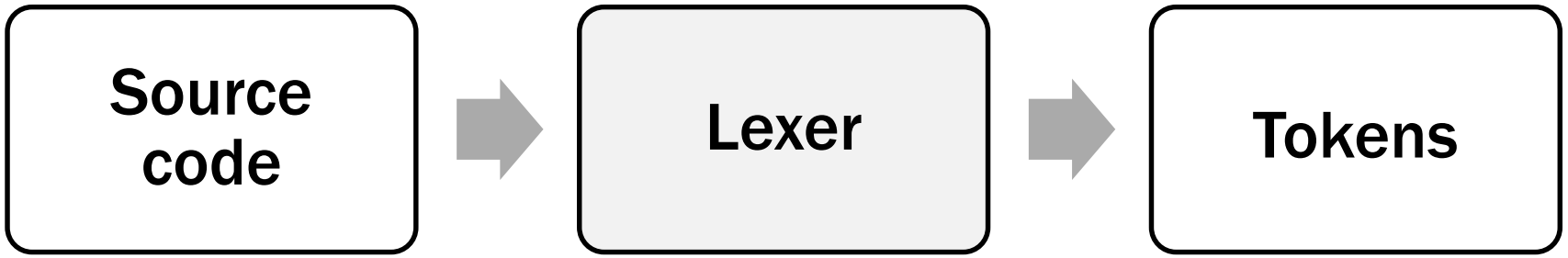


- **Get hands on practice using the dis module to look at Python Bytecode**

b'd\x01\x00}\x00\x00|\x00\x00d\x02\x00k\x00\x00r\x16\x00d\x03\x00Sd\x04\x00Sd\x00\x00S'

Lexer

Organize your source code into tokens



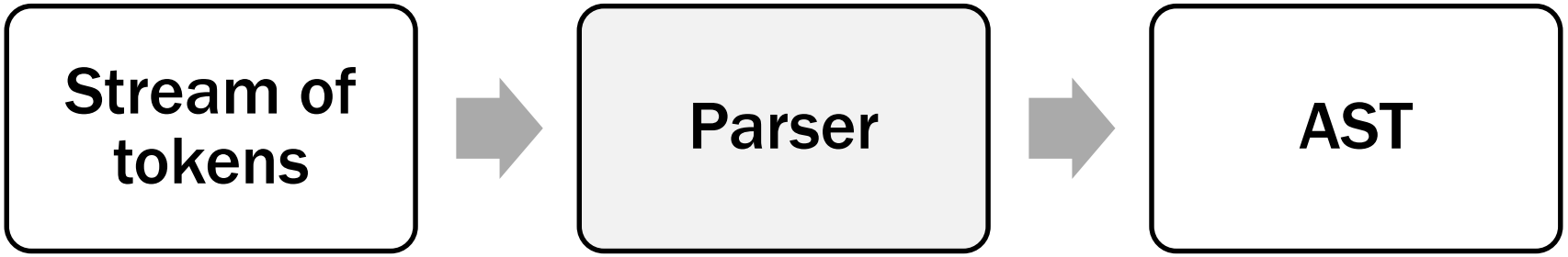
Groups characters together into chunk called tokens and identifies them

Parser

Takes the tokens and creates a structured code object

- Can be explored using the ast module

```
>>> tree = ast.parse("print('hello world')")
>>> tree
<_ast.Module object at 0x9e3df6c>
>>> exec(compile(tree, filename="<ast>", mode="exec"))
hello world
```




```
>>> parseprint("a, *b = it")
Module(body=[
    Assign(targets=[
        Tuple(elts=[
            Name(id='a', ctx=Store()),
            Starred(value=Name(id='b', ctx=Store()), ctx=Store()),
          ], ctx=Store()),
      ], value=Name(id='it', ctx=Load())),
  ])
```

Outputs Abstract Syntax Tree – missing docs https://greentreesnakes.readthedocs.io/

Compiler

Traverses the Abstract Syntax Tree to generate Bytecode

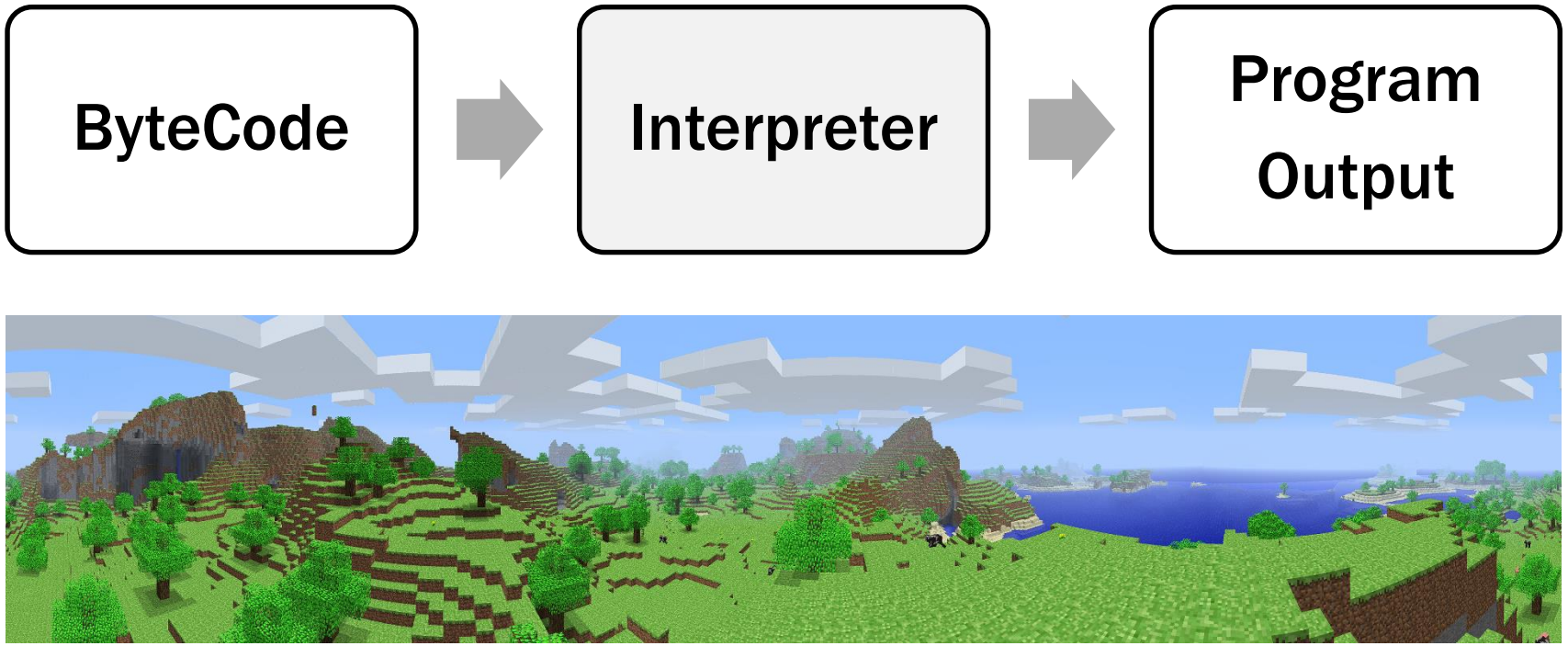
- Output can be disassembled using the dis module
- https://docs.python.org/3/library/dis.html

Outputs ByteCode, machine readable instructions for the interpreter

Interpreter

Simple stack based virtual machine executing your code

- Output can be disassembled using the dis module

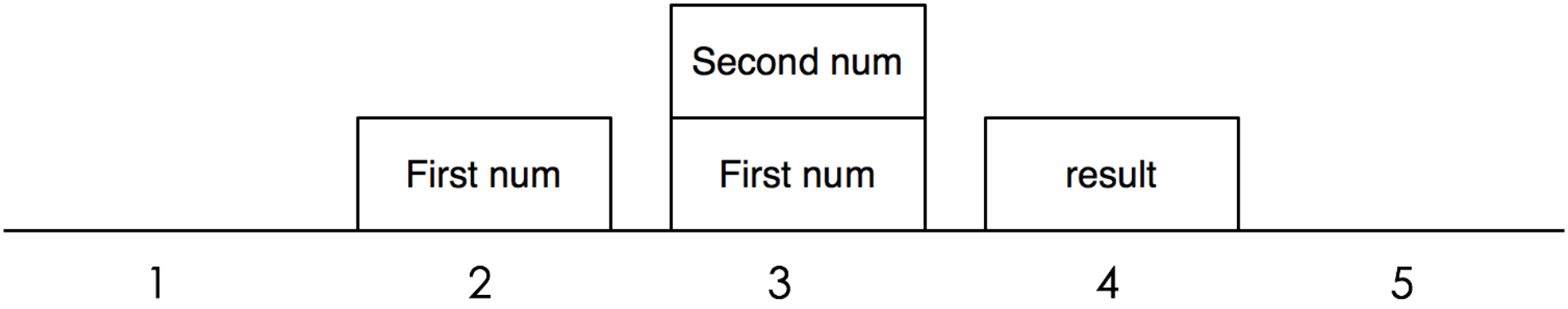


Interpreter

Only understand limited set of instructions (circa ~100)

- Suppose we want to execute 7 + 5
- We could represent it as:
  - INSTRUCTION - ARGUMENT
  - LOAD_VALUE – 0
  - LOAD_VALUE - 1
  - ADD_TWO_VALUES - None
  - PRINT_ANSWER - None

Stack states while executing the code above:



Executes the ByteCode instructions

Bytecode

Machine readable set of instructions for interpreter

- An intermediate representation of your program
- What the interpreter works with when it runs your program
- Machine code for a virtual machine
- A series of instructions for stack operations
- Cached as .pyc files

ByteCode might mean many things...

Dis module

Module used to disassemble Python bytecode

```
In [1]: def greet():
            message = "Hello CAMPUG"
            return message

In [2]: import dis

In [3]: dis.dis(greet)
  2           0 LOAD_CONST               1 ('Hello CAMPUG')
              3 STORE_FAST               0 (message)

  3           6 LOAD_FAST                0 (message)
              9 RETURN_VALUE
```

instruction

line no

index in bytecode

operation name

argument

arg value

https://docs.python.org/3/library/dis.html

Thanks!

Further resources

- A Python Interpreter written in Python:
  - http://www.aosabook.org/en/500L/a-python-interpreter-written-in-python.html
  - https://github.com/nedbat/byterun
- Hand crafted ByteCode:
  - http://multigrad.blogspot.co.uk/2014/06/fun-with-python-bytecode.html
- Anjana Vakil presentation:
  - https://speakerdeck.com/vakila/exploring-python-bytecode
- Docs for the dis module:
  - https://docs.python.org/3/library/dis.html
- Exploring ceval.c at the heart of interpreter:
  - https://tech.blog.aknin.name/category/my-projects/pythons-innards/