Introduction to Python
P.1

Agenda

- The Zen of Python
- First steps
- Basic Python
- Fun :)

The Zen of Python

```
import this
```

The Zen of Python

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!

First steps

Interpreter Prompt

First steps

<Interpreter Prompt>

REPL

Read-**E**val-**P**rint **L**oop

First steps

<Interpreter Prompt>

```
>>> print(“Hello World!”)
```

First steps

<Interpreter Prompt>

```
>>> print(“Hello World!”)
```

> "The 'Hello World' example is the traditional incantation to the programming gods and will ensure your quick mastery of the language, so please make sure you actually do this exercise, instead of just reading about it."
>
> -- said Simon Cozens in "Beginning Pearl"

First steps

<Interpreter Prompt>

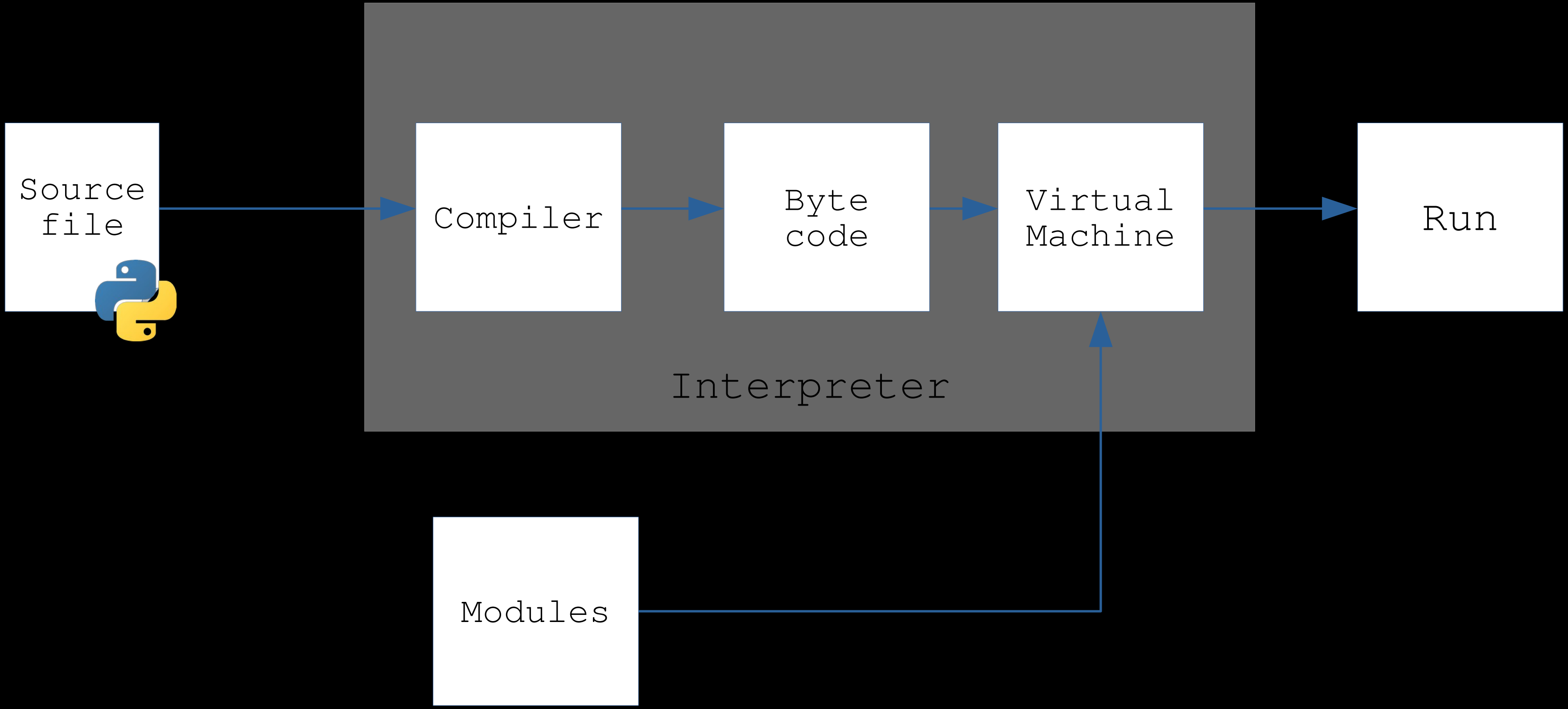
```
>>> exit()
```

First steps

Editor and Source File

First steps

<Source File>


Source file
Compiler
Byte code
Virtual Machine
Run
Interpreter
Modules

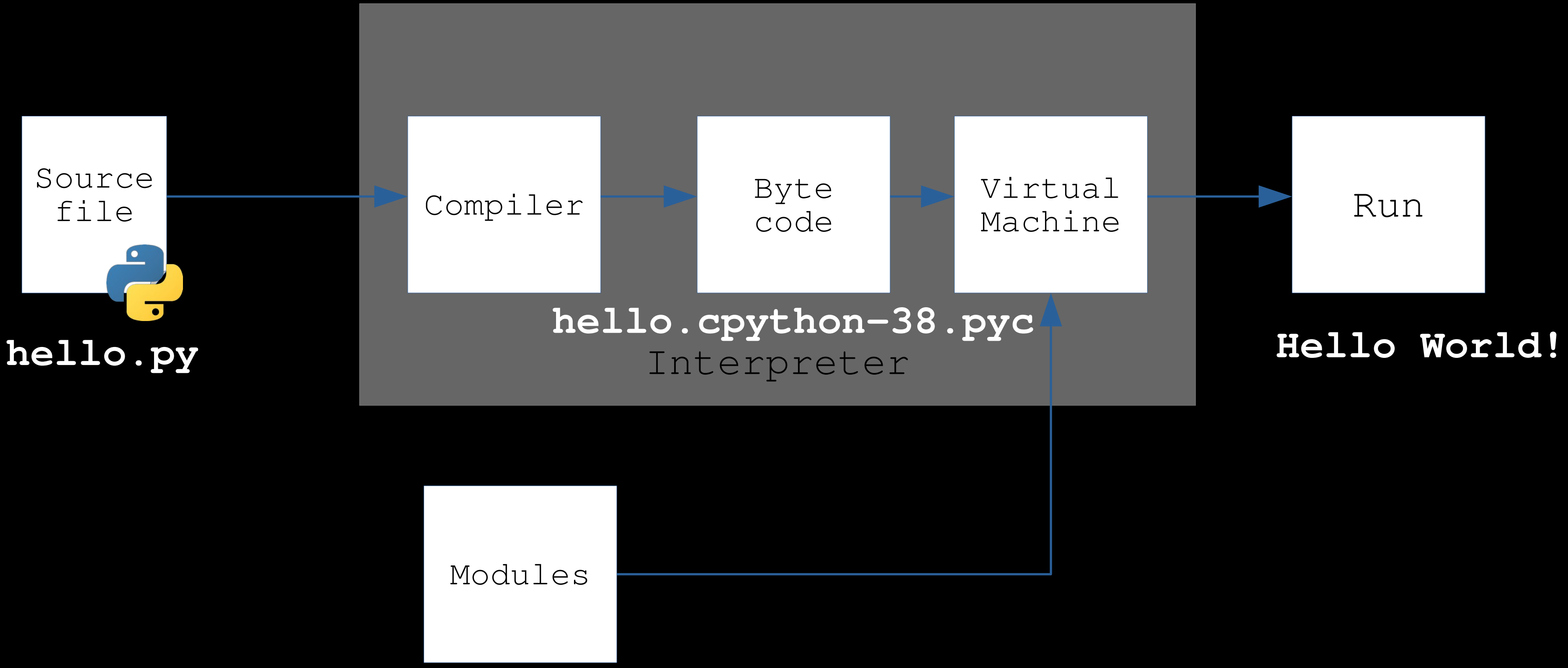
First steps
<Source File>

hello.py

touch this file

First steps

<Source File>


Source file
hello.py
Compiler
Byte code
Virtual Machine
hello.cpython-38.pyc
Interpreter
Run
Hello World!
Modules

First steps

Getting Help

First steps

<Getting Help>

```
>>> help()
```

Basic Python 1

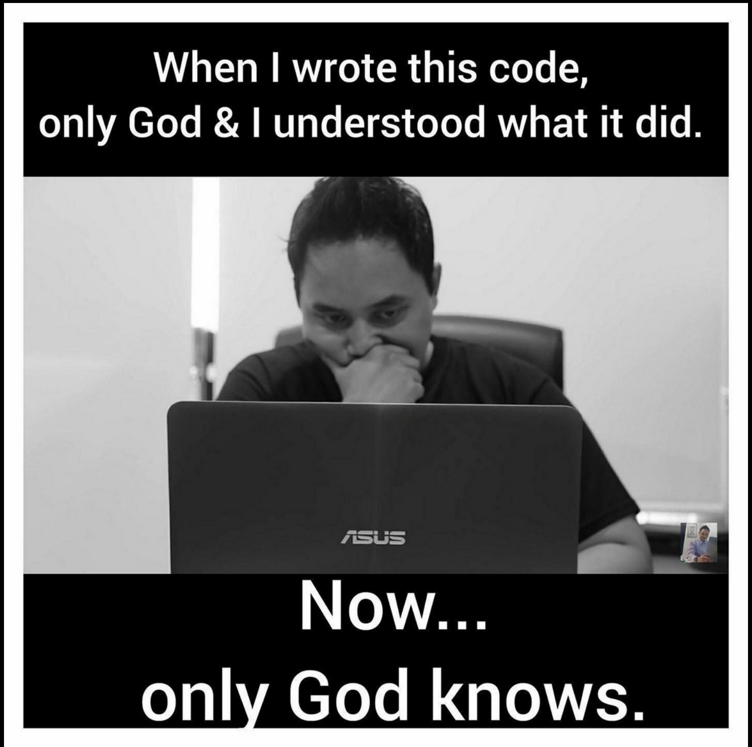
```
print()
```

Basic Python 1

Comments

Basic Python 1

<Comments>



Basic Python 1

Literal Constants

Basic Python 1

Basic Operations with Numbers

Basic Python 1

<Basic Operations with Numbers>

```
>>> 1729**1729
```

Basic Python 1

<Basic Operations with Numbers>

```
>>> # loss-of-precision for float
>>> a = 1/10
>>> print(“{:.50f}”.format(a))
```

Basic Python 1

<Basic Operations with Numbers>

```
>>> # overflow
>>> 1.7e308
>>> 1.8e308
```

Basic Python 1

<Basic Operations with Numbers>

```
>>> # underflow
>>> 5e-324
>>> 1e-325
```

Basic Python 1

Basic Operations with Strings

Basic Python 1

rectangle.py

Given width and height of a triangle 5 and 3 respectively, print out its Perimeter and Area

Sample output:

```
>>> python3 rectangle.py
Perimeter: 16
Area: 15
```

Basic Python 1

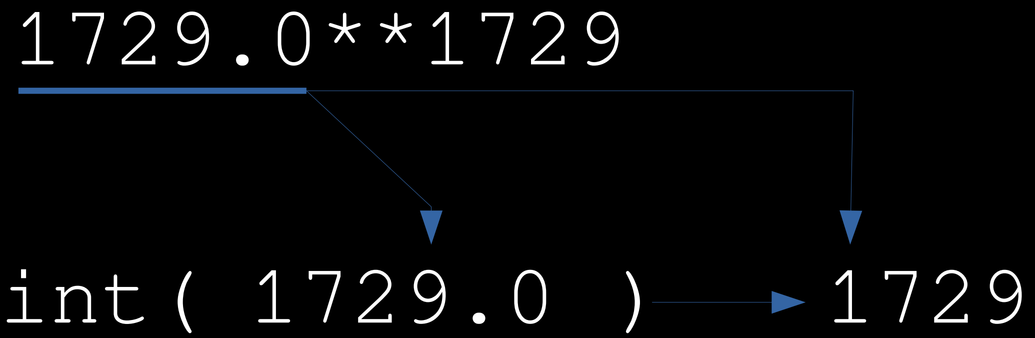
Data Type

Basic Python 1

Casting

Basic Python 1

<Casting>



```
1729.0**1729

int( 1729.0 ) 1729
```

Basic Python 1

<Casting>

```
str(1729)          →  ‘1729’
<class ‘int’>         <class ‘str’>

complex(‘1729’)    →  1729+0j
<class ‘str’>         <class ‘complex’>

int(‘10001’,2)     →  17
<class ‘int’>         <class ‘str’>

int(‘a’)           →  ?
```

Basic Python 1

Variable

Basic Python 1

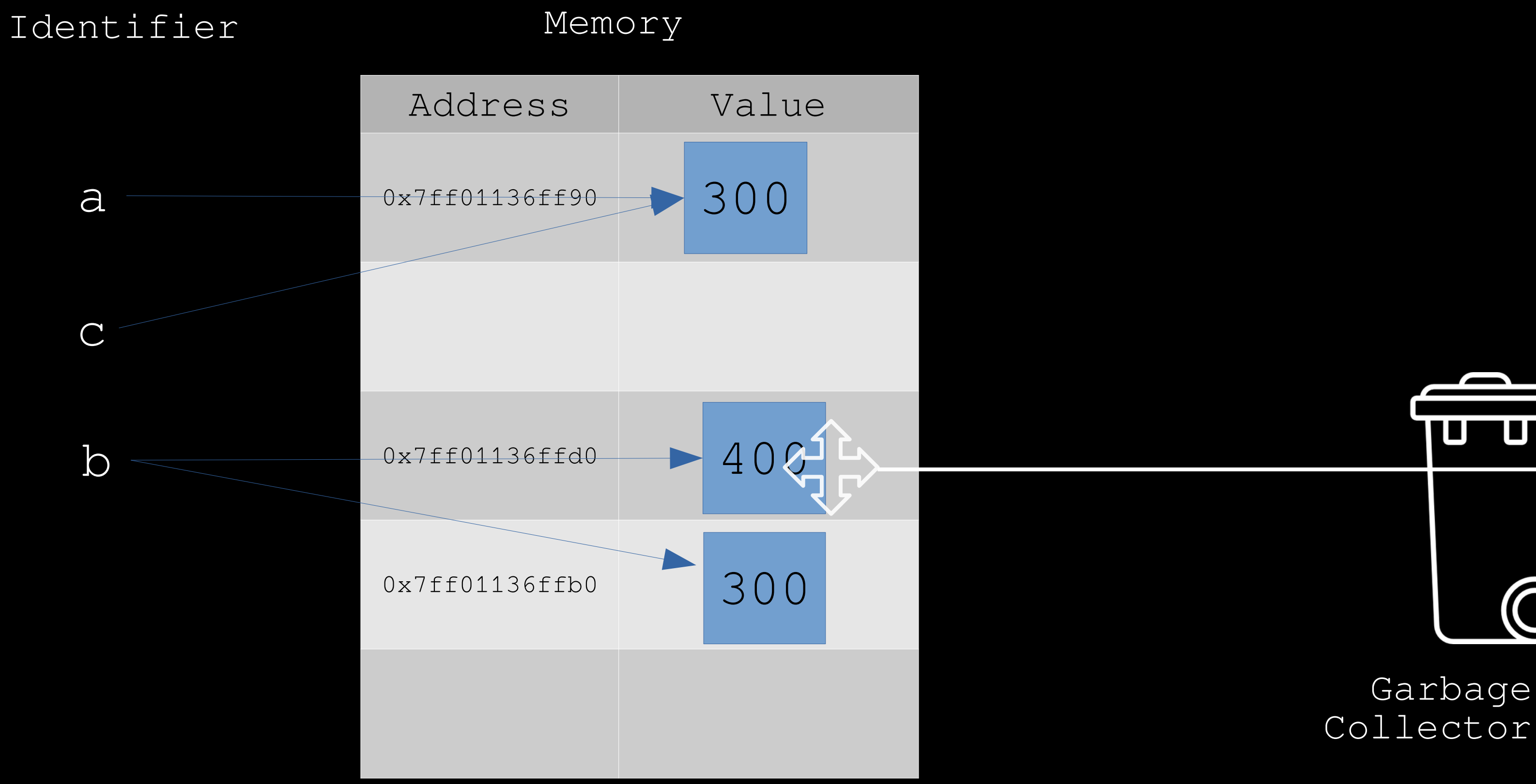
<Variable>

```
a = 300

b = 400

c = a

b = 300
```



Address	Value
0x7ff01136ff90	300
0x7ff01136ffd0	400
0x7ff01136ffb0	300

Basic Python 1

rectangle.py

Modify the rectangle.py script, now use variable to represent width and height lengths

Sample output:

```
>>> python3 rectangle.py
Perimeter: 16
Area: 15
```

Basic Python 1

Built-in Functions

Basic Python 1

<Built-in Functions>

- abs()
- all()
- any()
- ascii()
- bin()
- bool()
- breakpoint()
- bytearray()
- bytes()
- callable()
- chr()
- classmethod()
- compile()
- complex()
- delattr()
- dict()
- dir()
- divmod()
- enumerate()
- eval()
- exec()
- filter()
- float()
- format()
- frozenset()
- getattr()
- globals()
- hasattr()
- hash()
- help()
- hex()
- id()
- input()
- int()
- isinstance()
- issubclass()
- iter()
- len()
- list()
- locals()
- map()
- max()
- memoryview()
- min()
- next()
- object()
- oct()
- open()
- ord()
- pow()
- print()
- property()
- range()
- repr()
- reversed()
- round()
- set()
- setattr()
- slice()
- sorted()
- staticmethod()
- str()
- sum()
- super()
- tuple()
- type()
- vars()
- zip()
- __import__()

Basic Python 1

rectangle.py

Modify the rectangle.py script, now use user input for width and height and they are both whole numbers.

Sample output:

```
>>> python3 rectangle.py
Type in width: 5
Type in height: 3
Perimeter: 16
Area: 15
```

Basic Python 1

square.py

Given the Perimeter of a Square as user input (can be real number), print out its Area.

Sample output:

```
>>> python3 square.py
Type in Square’s Perimeter: 4
It’s Area: 1
```

Basic Python 1

circle.py

Given the Circumference of a Circle as user input (can be real number), print out its Area, rounded to the 5th decimal point.

Sample output:

```
>>> python3 circle.py
Type in Circle’s Circumference: 1
It’s Area: 0.07958
```

Basic Python 1

regular_polygon.py

Given the Perimeter of a polygon (real number) and its number of vertices (whole number) as user input, print out the polygon's Area.

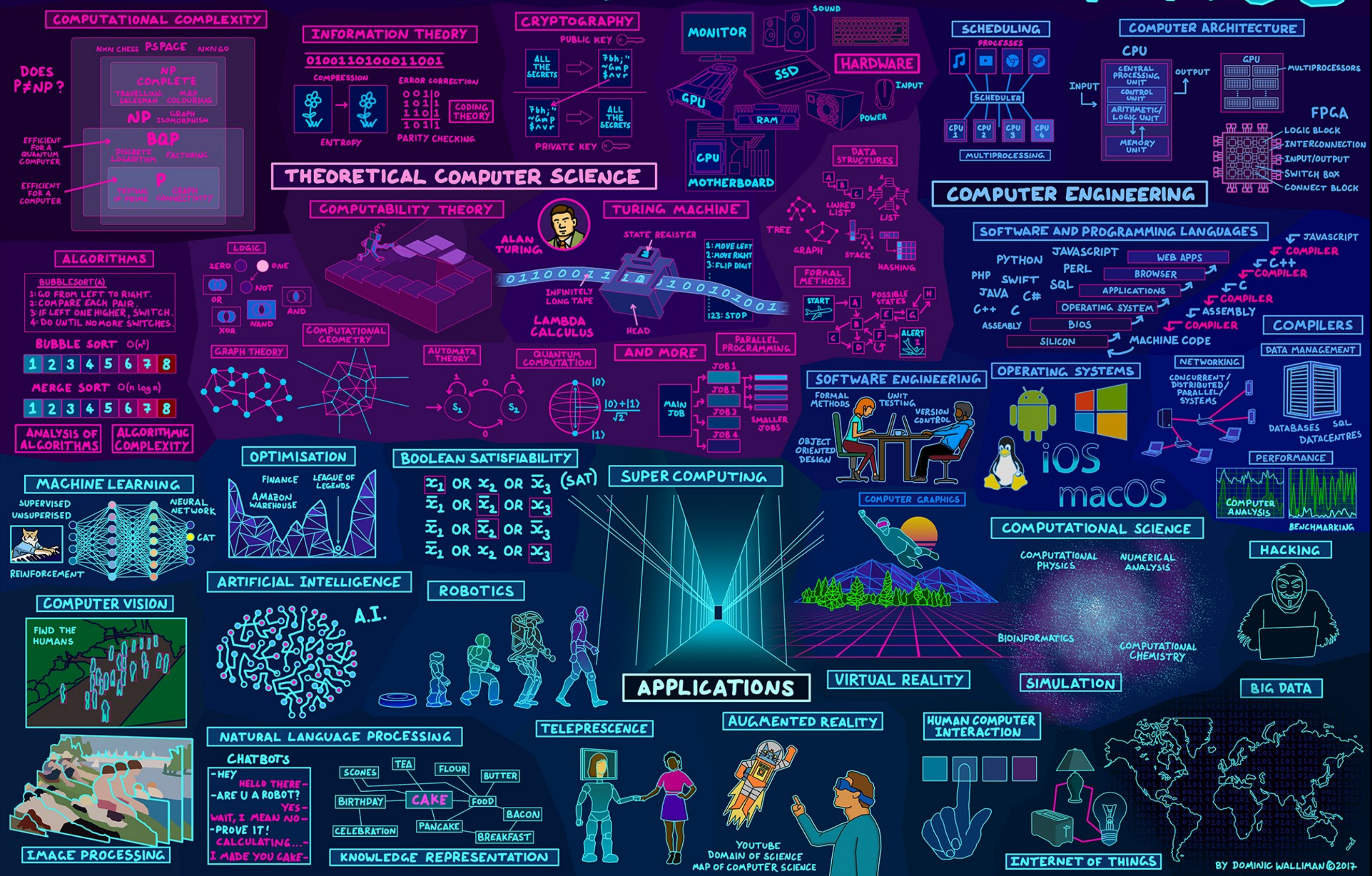
Sample output:

```
>>> python3 regular_polygon.py
Type in Polygon’s Perimeter: 6
Type in number of vertices: 6
It’s Area: 2.598076211353316
```

Basic Python 1

Queen Dido problem

MAP OF COMPUTER SCIENCE


THEORETICAL COMPUTER SCIENCE
COMPUTATIONAL COMPLEXITY
DOES P≠NP?
NxN CHESS
PSPACE
NxN GO
NP COMPLETE
TRAVELLING SALESMAN
MAP COLOURING
NP
GRAPH ISOMORPHISM
BQP
DISCRETE LOGARITHM
FACTORING
P
EFFICIENT FOR A QUANTUM COMPUTER
EFFICIENT FOR A COMPUTER
INFORMATION THEORY
010011010001100l
COMPRESSION
ENTROPY
ERROR CORRECTION
PARITY CHECKING
CODING THEORY
CRYPTOGRAPHY
PUBLIC KEY
ALL THE SECRETS
PRIVATE KEY
COMPUTABILITY THEORY
ALAN TURING
TURING MACHINE
STATE REGISTER
1: MOVE LEFT
2: MOVE RIGHT
3: FLIP DIGIT
123: STOP
INFINITELY LONG TAPE
HEAD
LAMBDA CALCULUS
ALGORITHMS
BUBBLESORT(A)
1: GO FROM LEFT TO RIGHT.
2: COMPARE EACH PAIR.
3: IF LEFT ONE HIGHER, SWITCH.
4: DO UNTIL NO MORE SWITCHES.
BUBBLE SORT O(n²)
MERGE SORT O(n log n)
ANALYSIS OF ALGORITHMS
ALGORITHMIC COMPLEXITY
LOGIC
ZERO
ONE
OR
NOT
AND
XOR
NAND
COMPUTATIONAL GEOMETRY
GRAPH THEORY
AUTOMATA THEORY
S1
S2
QUANTUM COMPUTATION
|0⟩
|1⟩
(|0⟩+|1⟩)/√2
AND MORE
PARALLEL PROGRAMMING
MAIN JOB
JOB 1
JOB 2
JOB 3
JOB 4
SMALLER JOBS
DATA STRUCTURES
LINKED LIST
LIST
TREE
GRAPH
STACK
HASHING
FORMAL METHODS
START
POSSIBLE STATES
ALERT
HARDWARE
MONITOR
SOUND
SSD
GPU
RAM
INPUT
POWER
CPU
MOTHERBOARD
COMPUTER ENGINEERING
SCHEDULING
PROCESSES
SCHEDULER
CPU 1
CPU 2
CPU 3
CPU 4
MULTIPROCESSING
COMPUTER ARCHITECTURE
CPU
CENTRAL PROCESSING UNIT
CONTROL UNIT
ARITHMETIC/LOGIC UNIT
MEMORY UNIT
INPUT
OUTPUT
GPU
MULTIPROCESSORS
FPGA
LOGIC BLOCK
INTERCONNECTION
INPUT/OUTPUT
SWITCH BOX
CONNECT BLOCK
SOFTWARE AND PROGRAMMING LANGUAGES
PYTHON
JAVASCRIPT
PHP
SWIFT
PERL
SQL
JAVA
C#
C++
C
ASSEMBLY
WEB APPS
BROWSER
APPLICATIONS
OPERATING SYSTEM
BIOS
SILICON
MACHINE CODE
JAVASCRIPT
COMPILER
C++
COMPILER
C
COMPILER
ASSEMBLY
COMPILER
COMPILERS
SOFTWARE ENGINEERING
FORMAL METHODS
UNIT TESTING
VERSION CONTROL
OBJECT ORIENTED DESIGN
OPERATING SYSTEMS
iOS
macOS
NETWORKING
CONCURRENT/DISTRIBUTED/PARALLEL/SYSTEMS
DATA MANAGEMENT
DATABASES
SQL
DATACENTRES
PERFORMANCE
COMPUTER ANALYSIS
BENCHMARKING
APPLICATIONS
OPTIMISATION
FINANCE
LEAGUE OF LEGENDS
AMAZON WAREHOUSE
BOOLEAN SATISFIABILITY
x₁ OR x₂ OR x̄₃ (SAT)
x̄₁ OR x̄₂ OR x₃
x̄₁ OR x̄₂ OR x̄₃
x̄₁ OR x₂ OR x̄₃
SUPER COMPUTING
MACHINE LEARNING
SUPERVISED
UNSUPERVISED
REINFORCEMENT
NEURAL NETWORK
CAT
ARTIFICIAL INTELLIGENCE
A.I.
ROBOTICS
COMPUTER VISION
FIND THE HUMANS
IMAGE PROCESSING
NATURAL LANGUAGE PROCESSING
CHATBOTS
-HEY
HELLO THERE-
-ARE U A ROBOT?
YES-
WAIT, I MEAN NO-
-PROVE IT!
CALCULATING...-
I MADE YOU CAKE-
KNOWLEDGE REPRESENTATION
SCONES
TEA
FLOUR
BUTTER
BIRTHDAY
CAKE
FOOD
BACON
CELEBRATION
PANCAKE
BREAKFAST
TELEPRESENCE
AUGMENTED REALITY
YOUTUBE
DOMAIN OF SCIENCE
MAP OF COMPUTER SCIENCE
COMPUTER GRAPHICS
VIRTUAL REALITY
COMPUTATIONAL SCIENCE
COMPUTATIONAL PHYSICS
NUMERICAL ANALYSIS
BIOINFORMATICS
COMPUTATIONAL CHEMISTRY
SIMULATION
HUMAN COMPUTER INTERACTION
INTERNET OF THINGS
HACKING
BIG DATA
BY DOMINIC WALLIMAN ©2017