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

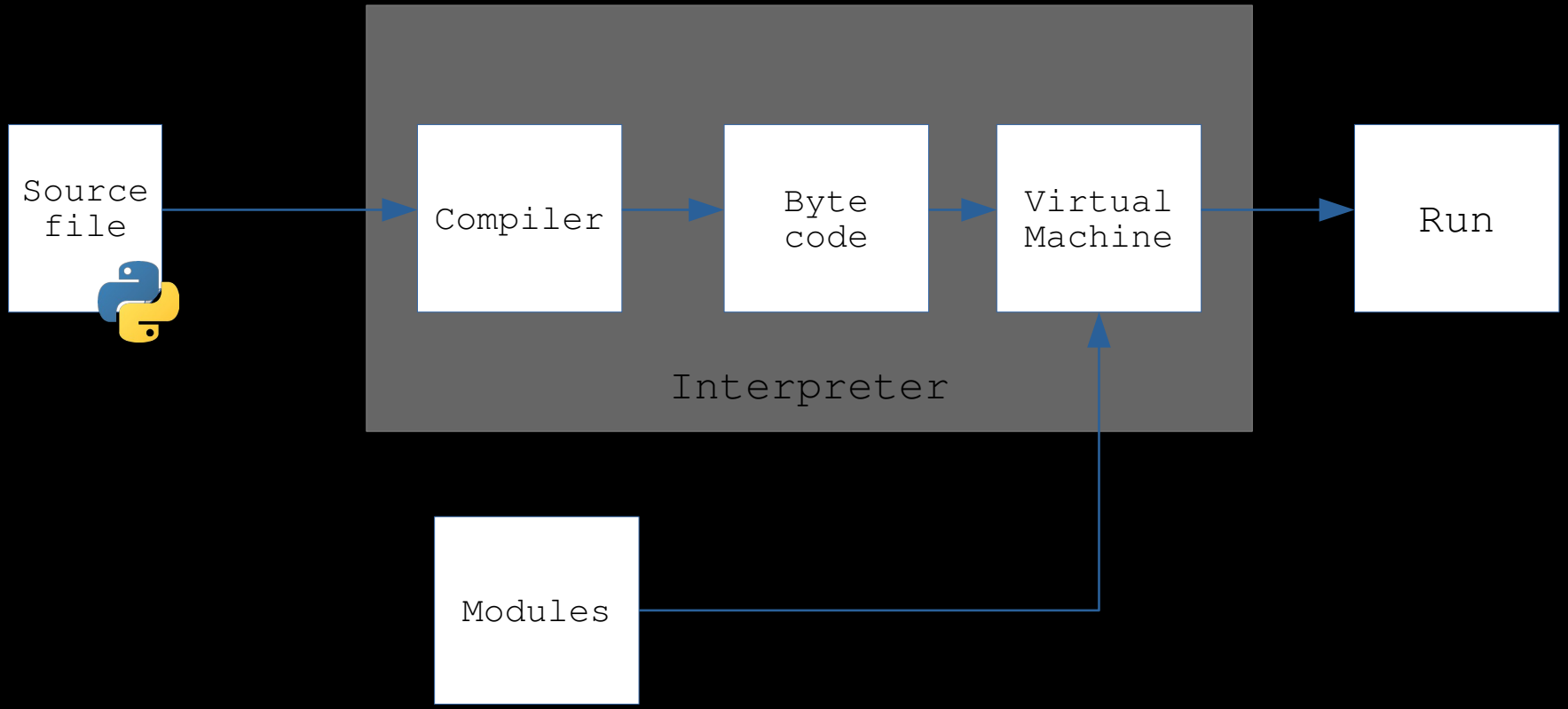
<Interpreter Prompt>

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

First steps

Editor and Source File

First steps
<Source File>



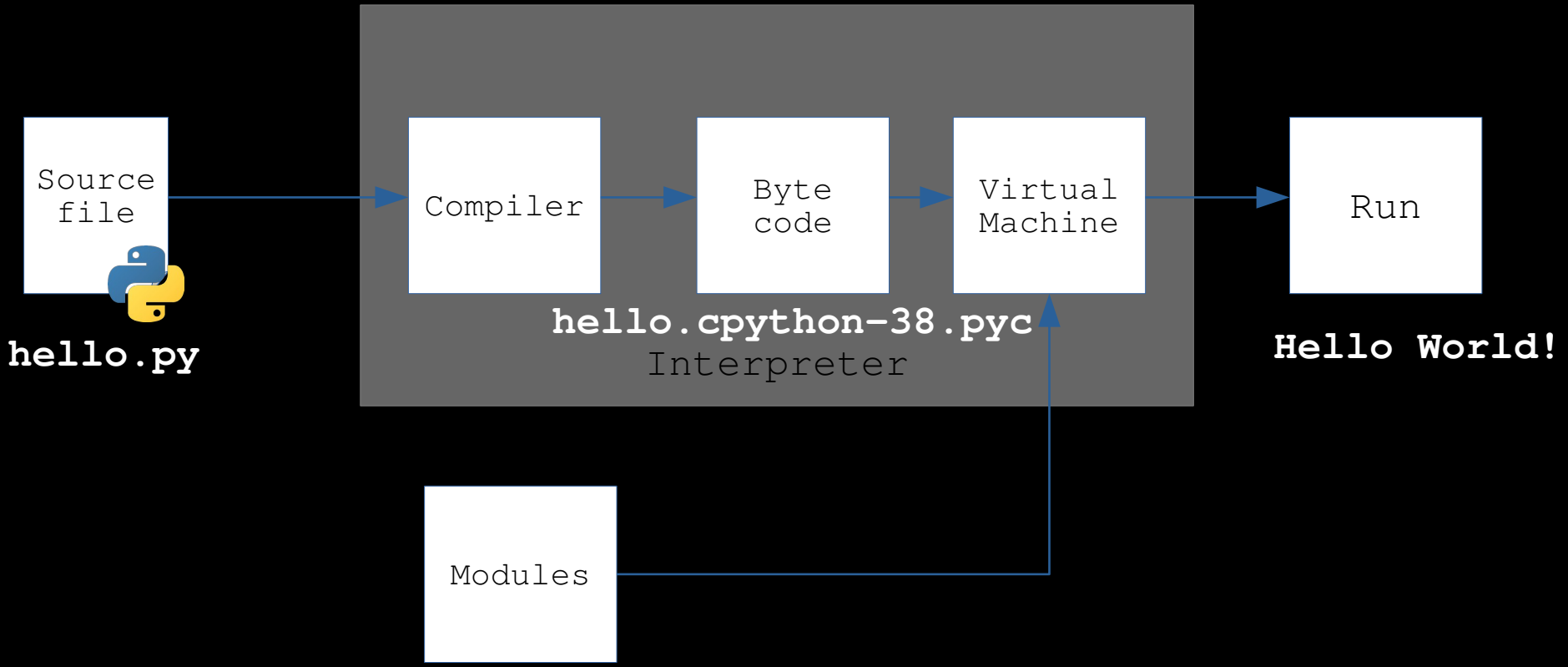
First steps

<Source File>

hello.py

touch this file

First steps
<Source File>



First steps

Getting Help

First steps
<Getting Help>

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

Basic Python 1

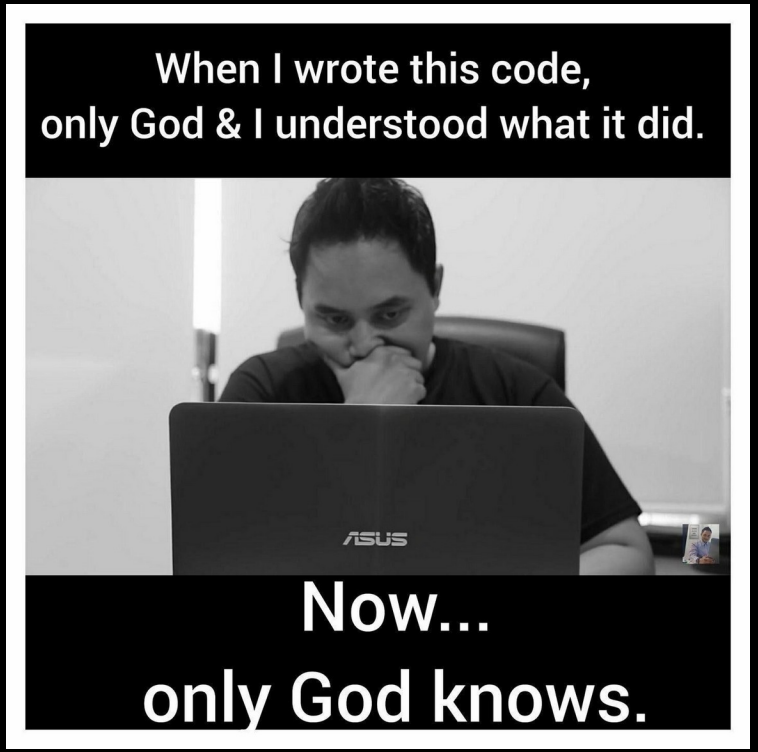
```
print()
```

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, print out its Perimeter and Area

Basic Python 1

Data Type

Basic Python 1

Casting

Basic Python 1

<Casting>

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')           →  ?
```

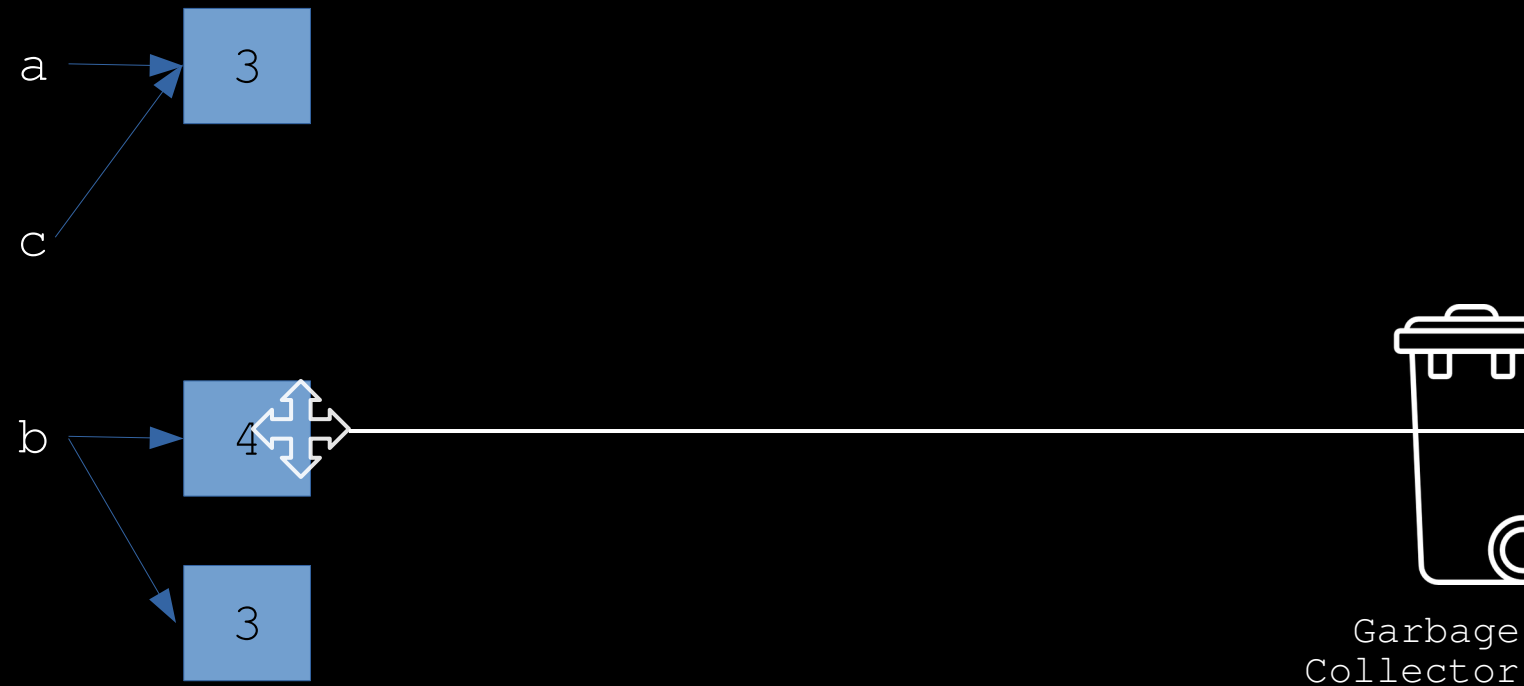
Basic Python 1

Variable

Basic Python 1

<Variable>

```
a = 3
b = 4
c = a
b = 3
```



Basic Python 1

rectangle.py

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

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

Basic Python 1

square.py

Given the Perimeter of a Square as user input,
print out its Area

Basic Python 1

circle.py

Given the Circumference of a Circle as user input, print out its Area

Basic Python 1

regular_polygon.py

Given the Perimeter of a polygon and its number of vertices as user input, print out the polygon's Area

Basic Python 1

Queen Dido problem

MAP OF COMPUTER SCIENCE

THEORETICAL COMPUTER SCIENCE

COMPUTATIONAL COMPLEXITY
DOES P≠NP?
NXN CHESS PSPACE NXNXN GO
NP COMPLETE
TRAVELLING SALESMAN MAP COLOURING
NP GRAPH ISOMORPHISM
BQP
DISCRETE LOGARITHM FACTORING
P
TESTING IF PRIME GRAPH CONNECTIVITY
EFFICIENT FOR A QUANTUM COMPUTER
EFFICIENT FOR A COMPUTER

INFORMATION THEORY
0100110100011001
COMPRESSION ENTROPY
ERROR CORRECTION
0 0 1 1 0
1 1 0 0 1
1 1 0 0 1
1 0 1 1 1
CODING THEORY
PARITY CHECKING

CRYPTOGRAPHY
PUBLIC KEY
ALL THE SECRETS
7bh;"ngmp$avr
7bh;"ngmp$avr
ALL THE SECRETS
PRIVATE KEY

COMPUTABILITY THEORY
LOGIC
ZERO ONE
OR NOT AND
XOR NAND
GRAPH THEORY
COMPUTATIONAL GEOMETRY

TURING MACHINE
ALAN TURING
STATE REGISTER
1: MOVE LEFT
2: MOVE RIGHT
3: FLIP DIGIT
INFINITELY LONG TAPE
HEAD
123: STOP
LAMBDA CALCULUS

AUTOMATA THEORY
S1 S2

QUANTUM COMPUTATION
|0⟩ |0⟩+|1⟩/√2 |1⟩

AND MORE
MAIN JOB
JOB 1
JOB 2
JOB 3
JOB 4
SMALLER JOBS

PARALLEL PROGRAMMING

DATA STRUCTURES
LINKED LIST
TREE GRAPH STACK HASHING

FORMAL METHODS
START ALERT
POSSIBLE STATES

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)
1 2 3 4 5 6 7 8
MERGE SORT O(n log n)
1 2 3 4 5 6 7 8
ANALYSIS OF ALGORITHMS
ALGORITHMIC COMPLEXITY

COMPUTER ENGINEERING

HARDWARE
MONITOR
SOUND
SSD
GPU
RAM
POWER
INPUT
CPU
MOTHERBOARD

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 WEB APPS
PHP SWIFT PERL BROWSER
SQL APPLICATIONS
JAVA C# C++ C OPERATING SYSTEM
ASSEMBLY BIOS
SILICON MACHINE CODE
JAVASCRIPT COMPILER
C++ COMPILER
C COMPILER
ASSEMBLY COMPILER

COMPILERS

DATA MANAGEMENT
DATABASES SQL DATACENTRES

NETWORKING
CONCURRENT/DISTRIBUTED/PARALLEL SYSTEMS

OPERATING SYSTEMS
iOS
macOS

SOFTWARE ENGINEERING
FORMAL METHODS
UNIT TESTING
VERSION CONTROL
OBJECT ORIENTED DESIGN

PERFORMANCE
COMPUTER ANALYSIS
BENCHMARKING

APPLICATIONS

MACHINE LEARNING
SUPERVISED UNSUPERVISED
NEURAL NETWORK
CAT
REINFORCEMENT

OPTIMISATION
FINANCE
LEAGUE OF LEGENDS
AMAZON WAREHOUSE

BOOLEAN SATISFIABILITY
x_1 OR x_2 OR $\bar{x}_3$
$\bar{x}_1$ OR $\bar{x}_2$ OR x_3
$\bar{x}_1$ OR $\bar{x}_2$ OR $\bar{x}_3$
$\bar{x}_1$ OR x_2 OR x_3
(SAT)

SUPER COMPUTING

COMPUTER GRAPHICS

COMPUTATIONAL SCIENCE
COMPUTATIONAL PHYSICS
NUMERICAL ANALYSIS
BIOINFORMATICS
COMPUTATIONAL CHEMISTRY

HACKING

ARTIFICIAL INTELLIGENCE
A.I.

ROBOTICS

COMPUTER VISION
FIND THE HUMANS

VIRTUAL REALITY

SIMULATION

BIG DATA

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

IMAGE PROCESSING

TELEPRESCENCE

AUGMENTED REALITY
YOUTUBE
DOMAIN OF SCIENCE
MAP OF COMPUTER SCIENCE

HUMAN COMPUTER INTERACTION

INTERNET OF THINGS

BY DOMINIC WALLIMAN ©2017