

# Why Does My Computer Do That? Intro to Coding with Python—Numbers

Dr. Ab Mosca (they/them)


Slides based off slides courtesy of Jordan Crouser (<https://jcrouser.github.io/>)

# Plan for Today

- More variables
- Numeric values and basic operations

## (RECAP) Core concept 1: variables

- In CS, a **variable** is a place to store a piece of data
- In Python, variables are:
  - **declared** by giving them a name
  - **assigned** using the equals sign
- Example:

declaring  
a variable `x`      `x = 3`  
         assigning  
the value 3 to `x`

# Keywords

- We want to use descriptive variable names
- Some words in Python\* are reserved as **keywords**, and cannot be used as a variable name:

```
and as assert break class continue def del  
elif else except exec finally for from  
global if import in is lambda not or pass  
raise return try while with yield
```

\* other languages have their own set of reserved words

## More about naming variables

- **Rule 1:** variable name must be at least 1 character long
- **Rule 2:** 1<sup>st</sup> character must be alphabetic  
(uppercase letter, lowercase letter, or underscore)
- **Rule 3:** variable names can contain letters, numbers, and underscores (but not spaces or other punctuation)

## Naming conventions

- These aren't *rules* (i.e. Python won't throw an error), but they make life a lot easier if you follow them

## Naming conventions

- These aren't *rules* (i.e. Python won't throw an error), but they make life a lot easier
- **Convention 1:** the name of the variable should tell you something about what the variable contains, e.g.

```
name = "Jordan"
```

is better than

```
blah = "Jordan"
```

## Naming conventions

- These aren't *rules* (i.e. Python won't throw an error), but they make life a lot easier
- **Convention 1:** the name of the variable should tell you something about what the variable contains, e.g.
- **Convention 2:** if you want to use multiple words as a variable name, separate them using `_underscores_`, or camel case e.g.

```
first_name = "Ab"
```

```
lastName = "Mosca"
```

(but stick to one convention)



# Naming conventions

- These aren't *rules* (i.e. Python won't throw an error), but they make life a lot easier
- **Convention 1:** the name of the variable should tell you something about what the variable contains, e.g.
- **Convention 2:** if you want to use multiple words as a variable name, separate them using `_underscores_`, or camel case e.g.
- **Convention 3:** if the value isn't going to change (i.e. the variable is a constant), use ALL CAPS, e.g.

```
PI = 3.14159
```

DEMO  
TIME

(RECAP) Core  
concept 2:  
numeric values

- Two kinds of **numbers** in CS:
  - integers (“whole numbers”)
  - floats (“decimals” or “floating point numbers”)
- In Python, the kind of number is implied by whether or not the number contains a **decimal point**
- Example:

`x = 3`

`x = 3.0`

# Discussion

Why do we care about whether or not a number has a **decimal point**?

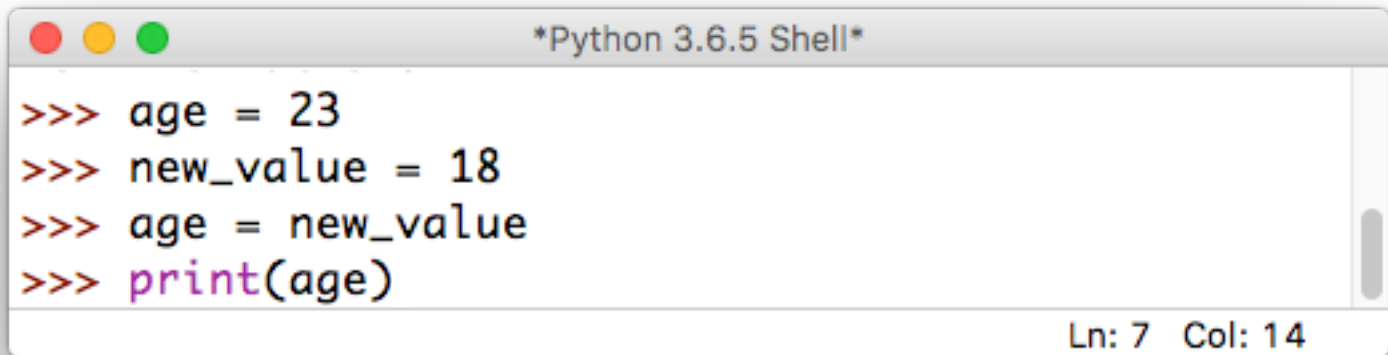
# Math

- Basic operators:
  - addition: +
  - subtraction: -
  - multiplication: \*
  - division: /
  - exponentiation: \*\* (power)
  - modular arithmetic: % (modulo)
- **Negative** values are allowed!

$$x = -3$$

# Overwriting variables

What happens if we do the following?

A screenshot of a Python 3.6.5 Shell window. The window has a title bar with three colored buttons (red, yellow, green) and the text '\*Python 3.6.5 Shell\*'. The main area contains four lines of Python code: '>>> age = 23', '>>> new\_value = 18', '>>> age = new\_value', and '>>> print(age)'. The 'print' function is highlighted in purple. The bottom right corner of the window shows 'Ln: 7 Col: 14'.

```
>>> age = 23
>>> new_value = 18
>>> age = new_value
>>> print(age)
```

Ln: 7 Col: 14

# Incrementing variables


What about this?

A screenshot of a Python 3.6.5 Shell window. The window has a title bar with three colored buttons (red, yellow, green) and the text '\*Python 3.6.5 Shell\*'. The main area contains three lines of code: '>>> age = 21', '>>> age = age + 1', and '>>> print(age)'. The 'print' function is highlighted in purple. The bottom right corner shows 'Ln: 13 Col: 14'.

```
>>> age = 21
>>> age = age + 1
>>> print(age)
```

# Incrementing variables

There's a shorthand for this!

A screenshot of a Python 3.6.5 Shell window. The window has a title bar with three colored buttons (red, yellow, green) and the text '\*Python 3.6.5 Shell\*'. The main area contains three lines of code: '>>> age = 21', '>>> age += 1', and '>>> print(age)'. The 'print' function is highlighted in purple. The bottom right corner shows 'Ln: 21 Col: 14'.

```
>>> age = 21
>>> age += 1
>>> print(age)
```

Ln: 21 Col: 14



## Quick exercise

```
a = 10
b = 20
c = 30
a = b          # a =
b = a          # b =
c = c * 2      # c =
d = d - 10     # d =
# NameError:
# name 'd' is not
# defined
```

these are called

**comments:**

they are not executed  
by the interpreter,  
but are useful  
for making  
code readable



Some more  
useful  
shorthand

- Simultaneous assignment:

```
a = 10
```

```
b = 20
```

```
c = 30
```

```
a = b
```

```
b = a
```

```
c = c * 2
```

## Some more useful shorthand

- Simultaneous assignment:

```
a, b, c = 10, 20, 30
```

```
a = b
```

```
b = a
```

```
c = c * 2
```

## Some more useful shorthand

- Simultaneous assignment:

```
a, b, c = 10, 20, 30
```

- Swapping variables:

```
a = b
```

```
b = a
```

```
c = c * 2
```

## Some more useful shorthand

- Simultaneous assignment:

```
a, b, c = 10, 20, 30
```

- Swapping variables:

```
a, b = b, a  
c = c * 2
```

## Quick exercise: unit conversion

- Find a partner, and write a program that asks the user to `input()` a number representing a file size in **Kb**
- Store the user input in an appropriate **variable**
- Calculate the equivalent size in **bits, bytes, Mb, and Gb**:
  - 1 byte = 8 bits
  - 1 Kb = 1024 bytes
  - 1 Mb = 1024 Kb
  - 1 Gb = 1024 Mb
- `print()` the **converted sizes** to the screen (ascending)
- Want a **challenge**? See if you can print the **units** beside each of the values (try the `str()` method)

# Discussion

What did you come up with?