

Welcome to the introduction
to computational cognitive
modelling workshop!

Part 1: Introduction to Python

Part 1: Introduction to Python

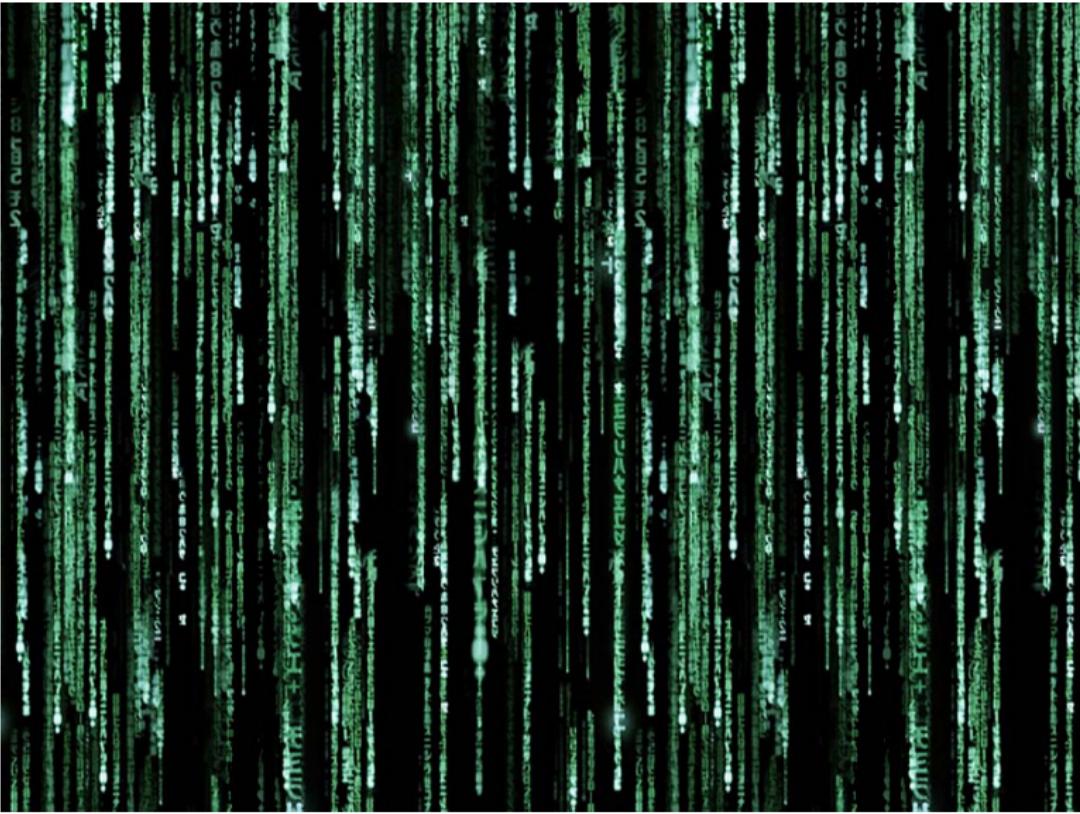
Olivia Guest

Chris Brand

Nick Sexton

Nicole Cruz De Echeverria Loebell

What is programming?



It is like use using any other language!

- ▶ Programming = telling a computer what to do

It is like use using any other language!

- ▶ Programming = telling a computer what to do
- ▶ You write the recipe, the computer follows it

It is like use using any other language!

- ▶ Programming = telling a computer what to do
- ▶ You write the recipe, the computer follows it
- ▶ Programming is an art, a craft, or a type of engineering.

Who is a programmer?



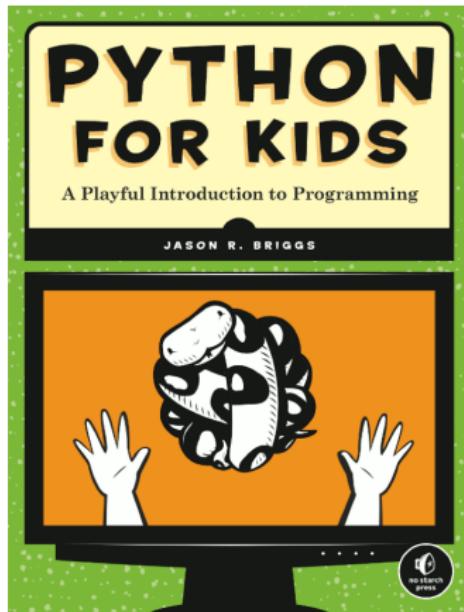
Who is a programmer?

Ada Lovelace, Grace Hopper, Amanda Spann, etc.



Who is a programmer?

Children!



Maddy Petrovich, 14, of Wellesley, Mass. first started learning how to use the programming language Scratch when she was 10 years old.

<http://hereandnow.wbur.org/2012/12/26/computer-programming-kids>

Who is a programmer?

You! Seriously

- ▶ **Anybody can program**

Who is a programmer?

You! Seriously

- ▶ **Anybody can program**
- ▶ But not everybody is Maya Angelou or Jane Austen!

Who is a programmer?

You! Seriously

- ▶ **Anybody can program**
- ▶ But not everybody is Maya Angelou or Jane Austen!
- ▶ But who cares? We can still communicate! We can still code!

Why program?

Why speak or write?

- ▶ Communicate ideas unambiguously

Why program?

Why speak or write?

- ▶ Communicate ideas unambiguously
- ▶ Allow your ideas to “live” on their own

Why program?

Why speak or write?

- ▶ Communicate ideas unambiguously
- ▶ Allow your ideas to “live” on their own
- ▶ It is actually fun and definitely addictive

Why program?

Why speak or write?

- ▶ Communicate ideas unambiguously
- ▶ Allow your ideas to “live” on their own
- ▶ It is actually fun and definitely addictive
- ▶ Save time by automating repetitive tasks

Why program?

Why speak or write?

- ▶ Communicate ideas unambiguously
- ▶ Allow your ideas to “live” on their own
- ▶ It is actually fun and definitely addictive
- ▶ Save time by automating repetitive tasks
- ▶ All the cool kids are doing it

What is Python?

Will it eat my mouse?



What is Python?

Not John Cleese, but closer...



What is Python?

Not a snake



- ▶ Named after Monty Python

What is Python?

Not a snake



- ▶ Named after Monty Python
- ▶ A programming language

What is Python?

Not a snake



- ▶ Named after Monty Python
- ▶ A programming language
- ▶ Not scary

What is Python?

Not a snake



- ▶ Named after Monty Python
- ▶ A programming language
- ▶ Not scary
- ▶ Cool

Who created Python?

A Dutch human

DOCTOR FUN

6 Apr 2000



Copyright © 2000 David Farley, d-farley@metalab.unc.edu
<http://metalab.unc.edu/Dave/drfun.html>

This cartoon is made available on the Internet for personal viewing only. Opinions expressed herein are solely those of the author.

Who created Python?

Guido van Rossum

aka Benevolent Dictator For Life



Why use Python?

It was created for you

- ▶ Easy and intuitive

Why use Python?

It was created for you

- ▶ Easy and intuitive
- ▶ Open source and free

Why use Python?

It was created for you

- ▶ Easy and intuitive
- ▶ Open source and free
- ▶ Close to plain English

Why use Python?

It was created for you

- ▶ Easy and intuitive
- ▶ Open source and free
- ▶ Close to plain English
- ▶ Lots of help online

Why use Python?

It was created for you

- ▶ Easy and intuitive
- ▶ Open source and free
- ▶ Close to plain English
- ▶ Lots of help online
- ▶ Even geeks like it

Time to get serious!

Open up 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!

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!

is easy to explain, it may be a good idea. Namespaces are one honking great idea — let's do more of those!

Assignment

Put a value into a variable

Python shell:

- ▶ Symbol in Python: =
- ▶ Does **not** mean equals!



Assignment

Put a value into a variable

Python shell:

- ▶ Symbol in Python: =
- ▶ Does **not** mean equals!
- ▶ Make the variable have the value
- ▶ Read as “assign”, “make”, or “equals”

```
>>> i = 5  
>>> print i
```

Assignment

Put a value into a variable

Python shell:

- ▶ Symbol in Python: =
- ▶ Does **not** mean equals!
- ▶ Make the variable have the value
- ▶ Read as “assign”, “make”, or “equals”

```
>>> i = 5  
>>> print i  
5
```

Assignment

Put a value into a variable

Python shell:

- ▶ Symbol in Python: =
- ▶ Does **not** mean equals!
- ▶ Make the variable have the value
- ▶ Read as “assign”, “make”, or “equals”

```
>>> i = 5  
>>> print i  
5  
>>> i + 2
```

Assignment

Put a value into a variable

Python shell:

- ▶ Symbol in Python: =
- ▶ Does **not** mean equals!
- ▶ Make the variable have the value
- ▶ Read as “assign”, “make”, or “equals”

```
>>> i = 5
>>> print i
5
>>> i + 2
7
```

Assignment

Put a value into a variable

Python shell:

- ▶ Symbol in Python: =
- ▶ Does **not** mean equals!
- ▶ Make the variable have the value
- ▶ Read as “assign”, “make”, or “equals”

```
>>> i = 5
>>> print i
5
>>> i + 2
7
>>> i = i + 1
>>> print i
```

Assignment

Put a value into a variable

Python shell:

- ▶ Symbol in Python: =
- ▶ Does **not** mean equals!
- ▶ Make the variable have the value
- ▶ Read as “assign”, “make”, or “equals”

```
>>> i = 5
>>> print i
5
>>> i + 2
7
>>> i = i + 1
>>> print i
6
```

Assignment

Put a value into a variable

Python shell:

- ▶ Symbol in Python: =
- ▶ Does **not** mean equals!
- ▶ Make the variable have the value
- ▶ Read as “assign”, “make”, or “equals”

```
>>> i = 5
>>> print i
5
>>> i + 2
7
>>> i = i + 1
>>> print i
6
>>> i += 0.8
```

Assignment

Put a value into a variable

Python shell:

- ▶ Symbol in Python: =
- ▶ Does **not** mean equals!
- ▶ Make the variable have the value
- ▶ Read as “assign”, “make”, or “equals”

```
>>> i = 5
>>> print i
5
>>> i + 2
7
>>> i = i + 1
>>> print i
6
>>> i += 0.8
7.8
```

Basic maths

Arithmetic is fun when you don't have to do it manually!

Python shell:

- ▶ Use Python as a calculator
- ▶ Assign results into variables
- ▶ All your favourites like plus, minus, times, divided by, etc.

```
>>> (41 + 45.87) / 2
```

Basic maths

Arithmetic is fun when you don't have to do it manually!

Python shell:

- ▶ Use Python as a calculator
- ▶ Assign results into variables
- ▶ All your favourites like plus, minus, times, divided by, etc.

```
>>> (41 + 45.87) / 2  
43.435
```

Basic maths

Arithmetic is fun when you don't have to do it manually!

Python shell:

- ▶ Use Python as a calculator
- ▶ Assign results into variables
- ▶ All your favourites like plus, minus, times, divided by, etc.

```
>>> (41 + 45.87) / 2  
43.435  
>>> 5**2
```

Basic maths

Arithmetic is fun when you don't have to do it manually!

Python shell:

- ▶ Use Python as a calculator
- ▶ Assign results into variables
- ▶ All your favourites like plus, minus, times, divided by, etc.

```
>>> (41 + 45.87) / 2  
43.435  
>>> 5**2  
25
```

Basic maths

Arithmetic is fun when you don't have to do it manually!

Python shell:

- ▶ Use Python as a calculator
- ▶ Assign results into variables
- ▶ All your favourites like plus, minus, times, divided by, etc.

```
>>> (41 + 45.87) / 2  
43.435  
>>> 5**2  
25  
>>> a = 8  
>>> b = a / 2  
>>> c = b**2
```

Basic maths

Arithmetic is fun when you don't have to do it manually!

Python shell:

- ▶ Use Python as a calculator
- ▶ Assign results into variables
- ▶ All your favourites like plus, minus, times, divided by, etc.

```
>>> (41 + 45.87) / 2  
43.435  
>>> 5**2  
25  
>>> a = 8  
>>> b = a / 2  
>>> c = b**2  
>>> print a, b, c
```

Basic maths

Arithmetic is fun when you don't have to do it manually!

Python shell:

- ▶ Use Python as a calculator
- ▶ Assign results into variables
- ▶ All your favourites like plus, minus, times, divided by, etc.

```
>>> (41 + 45.87) / 2  
43.435  
>>> 5**2  
25  
>>> a = 8  
>>> b = a / 2  
>>> c = b**2  
>>> print a, b, c  
8 4 6
```

Truth values

Perfectly logical

Python shell:

- ▶ == equality
- ▶ != inequality
- ▶ >=, >, <=, <
- ▶ and, or, not

```
>>> True and False  
False
```

Truth values

Perfectly logical

Python shell:

- ▶ == equality
- ▶ != inequality
- ▶ >=, >, <=, <
- ▶ and, or, not

```
>>> True and False  
False  
>>> True or False
```

Truth values

Perfectly logical

Python shell:

- ▶ == equality
- ▶ != inequality
- ▶ >=, >, <=, <
- ▶ and, or, not

```
>>> True and False  
False  
>>> True or False  
True
```

Truth values

Perfectly logical

Python shell:

- ▶ == equality
- ▶ != inequality
- ▶ >=, >, <=, <
- ▶ and, or, not

```
>>> True and False  
False  
>>> True or False  
True  
>>> not False
```

Truth values

Perfectly logical

- ▶ == equality
- ▶ != inequality
- ▶ >=, >, <=, <
- ▶ and, or, not

Python shell:

```
>>> True and False  
False  
>>> True or False  
True  
>>> not False  
True
```

Truth values

Perfectly logical

Python shell:

- ▶ == equality
- ▶ != inequality
- ▶ >=, >, <=, <
- ▶ and, or, not

```
>>> True and False  
False  
>>> True or False  
True  
>>> not False  
True  
>>> 100 == True
```

Truth values

Perfectly logical

Python shell:

- ▶ == equality
- ▶ != inequality
- ▶ >=, >, <=, <
- ▶ and, or, not

```
>>> True and False  
False  
>>> True or False  
True  
>>> not False  
True  
>>> 100 == True  
False
```

Truth values

Perfectly logical

Python shell:

- ▶ == equality
- ▶ != inequality
- ▶ >=, >, <=, <
- ▶ and, or, not

```
>>> True and False  
False  
>>> True or False  
True  
>>> not False  
True  
>>> 100 == True  
False  
>>> a != b
```

Truth values

Perfectly logical

Python shell:

- ▶ == equality
- ▶ != inequality
- ▶ >=, >, <=, <
- ▶ and, or, not

```
>>> True and False
False
>>> True or False
True
>>> not False
True
>>> 100 == True
False
>>> a != b
True
```

Time to get SUPER serious!

Open a text editor

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!

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!

is easy to explain, it may be a good idea. Namespaces are one honking great idea — let's do more of those!

Syntax

Tabs and spaces

- ▶ Syntax is important
- ▶ Very important

```
a = 2
b = 8
if a == b:

    print "equal"
elif a > b:

    print "a is bigger"
else:

    print "b is bigger"
...
```

Syntax

Tabs and spaces

- ▶ Syntax is important
- ▶ Very important
- ▶ Comments also

```
a = 2
b = 8
if a == b:
    # This is a comment!
    print "equal"
elif a > b:
    # Python ignores me!
    print "a is bigger"
else:
    # But you see me!
    print "b is bigger"
...
```

Conditional statements

If, then, else

- ▶ If X is the case, then do Y

```
a = 2  
b = 8  
if a == b:  
    #a and b the same  
    print "equal"
```

Conditional statements

If, then, else

- ▶ If X is the case, then do Y

```
a = 2
b = 8
if a == b:
    #a and b the same
    print "equal"

else:
    #a smaller than b
    print "b is bigger"
```

- ▶ otherwise do Z

Conditional statements

If, then, else

- ▶ If X is the case, then do Y
- ▶ but if A is the case, do B
- ▶ but if C is the case, do D
- ▶ ...
- ▶ otherwise do Z

```
a = 2
b = 8
if a == b:
    #a and b the same
    print "equal"
elif a > b:
    #b smaller than a
    print "a is bigger"
else:
    #a smaller than b
    print "b is bigger"
```

For loops

What sorcery is this?

- ▶ Do something

Text editor:

```
print 1
```

Python output:

For loops

What sorcery is this?

- ▶ Do something
- ▶ do something

Text editor:

```
print 1  
print 2
```

Python output:

For loops

What sorcery is this?

- ▶ Do something
- ▶ do something
- ▶ do something

Text editor:

```
print 1  
print 2  
print 3
```

Python output:

For loops

What sorcery is this?

- ▶ Do something
- ▶ do something
- ▶ do something
- ▶ do something

Text editor:

```
print 1
print 2
print 3
print 4
```

Python output:

For loops

What sorcery is this?

- ▶ Do something

Text editor:

```
print 1
print 2
print 3
print 4
print 5
```

Python output:

For loops

What sorcery is this?

- ▶ Do something

Text editor:

```
print 1
print 2
print 3
print 4
print 5
```

Python output:

```
1
2
3
4
5
```

For loops

What sorcery is this?

- ▶ Use a single command!
- ▶ Do something five times

Text editor:

```
for i in      (1,6):  
    print i
```

Python output:

For loops

What sorcery is this?

- ▶ Use a single command!
- ▶ Do something five times

Text editor:

```
for i in      (1,6):  
    print i
```

Python output:

```
1  
2  
3  
4  
5
```

While loops

More magic!

- ▶ For loop: for each of the 5 slices of pizza, eat them

While loops

More magic!

- ▶ For loop: for each of the 5 slices of pizza, eat them
- ▶ What if you don't know how many times you want to do something?

While loops

More magic!

- ▶ For loop: for each of the 5 slices of pizza, eat them
- ▶ What if you don't know how many times you want to do something?
- ▶ Do something *until* something else happens

While loops

More magic!

- ▶ For loop: for each of the 5 slices of pizza, eat them
- ▶ What if you don't know how many times you want to do something?
- ▶ Do something *until* something else happens
- ▶ While loop: while it's raining, use umbrella

While loops

More magic!

- ▶ For loop: for each of the 5 slices of pizza, eat them
- ▶ What if you don't know how many times you want to do something?
- ▶ Do something *until* something else happens
- ▶ While loop: while it's raining, use umbrella
- ▶ While there are slices of pizza, eat them

While loops

More magic!

Text editor:

```
while True:
```

Python output:

While loops

More magic!

Text editor:

```
while True:  
    n = input("Please enter 'hello':")
```

Python output:

While loops

More magic!

Text editor:

```
while True:  
    n = input("Please enter 'hello':")  
    if n == 'hello':
```

Python output:

While loops

More magic!

Text editor:

```
while True:  
    n = input("Please enter 'hello':")  
    if n == 'hello':  
        break
```

Python output:

While loops

More magic!

Text editor:

```
while True:  
    n = input("Please enter 'hello':")  
    if n == 'hello':  
        break
```

Python output:

```
Please enter 'hello':no
```

While loops

More magic!

Text editor:

```
while True:  
    n = input("Please enter 'hello':")  
    if n == 'hello':  
        break
```

Python output:

```
Please enter 'hello':no  
Please enter 'hello':never
```

While loops

More magic!

Text editor:

```
while True:  
    n = ("Please enter 'hello':")  
    if n == 'hello':  
        break
```

Python output:

```
Please enter 'hello':no  
Please enter 'hello':never  
Please enter 'hello':hi?
```

While loops

More magic!

Text editor:

```
while True:  
    n = ("Please enter 'hello':")  
    if n == 'hello':  
        break
```

Python output:

```
Please enter 'hello':no  
Please enter 'hello':never  
Please enter 'hello':hi?  
Please enter 'hello':hello
```

While loops

More magic!

Text editor:

```
while True:  
    n = ("Please enter 'hello':")  
    if n == 'hello':  
        break
```

Python output:

```
Please enter 'hello':no  
Please enter 'hello':never  
Please enter 'hello':hi?  
Please enter 'hello':hello  
>>>
```

Exercises

Do them!



Put your hand up if you need help!