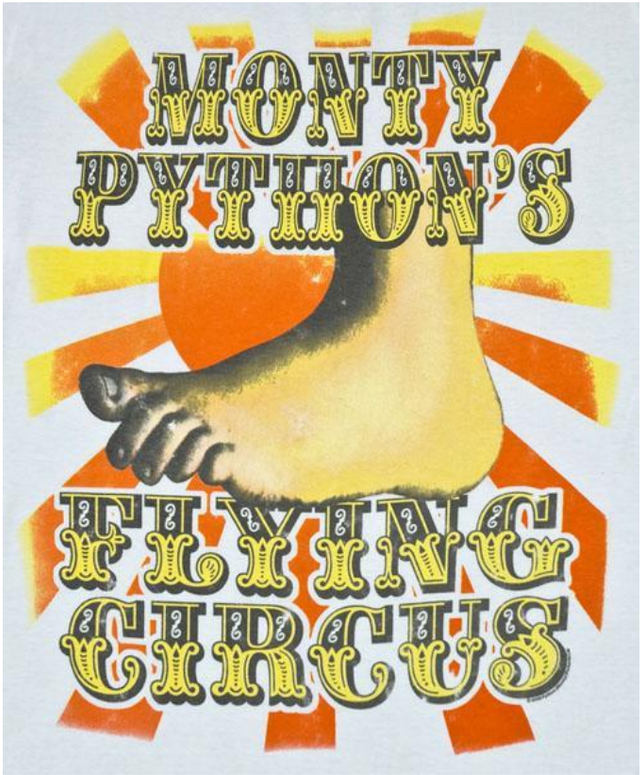
FIT1045: Algorithms and Programming Fundamentals in Python

Lecture 2

Introduction to Python



Correction: Hurdles

To pass this unit a student must obtain:

- ~~45%~~ 40% or more in the unit's examination, and
- ~~45%~~ 40% or more in the unit's total non-examination assessment, and
- an overall unit mark of 50% or more.

If a student does not pass these hurdles then a mark of no greater than **45-NH** will be recorded for the unit.

Announcements / Moodle updates

Assignments:

- **No interviews**
- Instead, a **written documentati**on is required

Week 1 section

- say hi to your tutor and class mates in "Week 1 Discussion Forums“ or on Ed
- Tutorials in week 1 are used to get to know each other, set up python and answer basic questions
- current **workbook** version available (additional resources)

Week 2 section

- The sheets for week 2 will be available by Friday (afternoon)
- the submission for tutorial prep and workshop will open by the end of this week

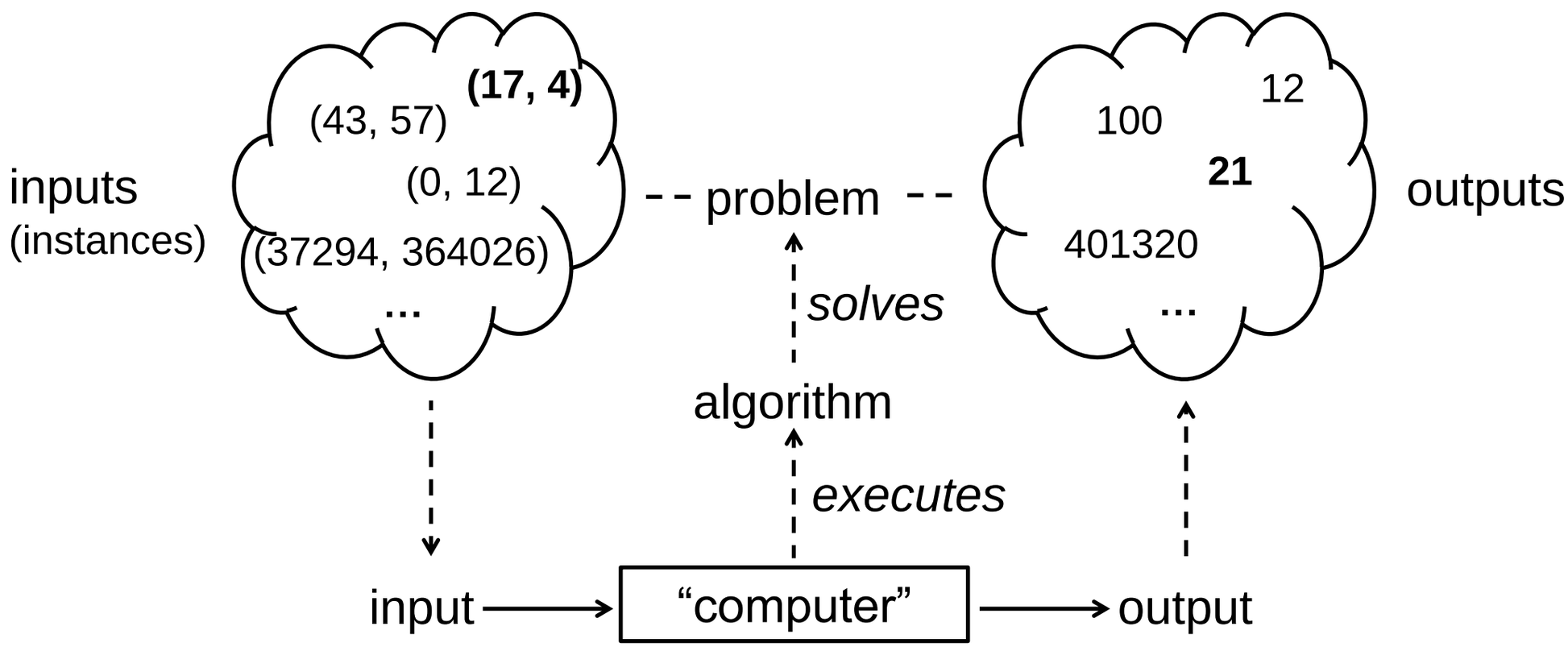
Golden rules for online discussions

- Identify yourself clearly in all communications.
- Review what you wrote and try to interpret it objectively.
- Would you be happy for your future employer to read your post?
- If you wouldn't say it face to face, don't say it online.
- Respect others' privacy.
- Seek help, not an unfair advantage.
- Follow the golden rules.
- **Don't post links to off-topic or inappropriate contents**

Last Lecture

A ***problem*** is a collection of *inputs* (questions), each of which has at least one correct associated *output* (answer).

An ***algorithm*** is a sequence of unambiguous *instructions* for solving a *problem*, i.e., for obtaining a required output for any legitimate input in a finite amount of time."



Objectives of this lecture

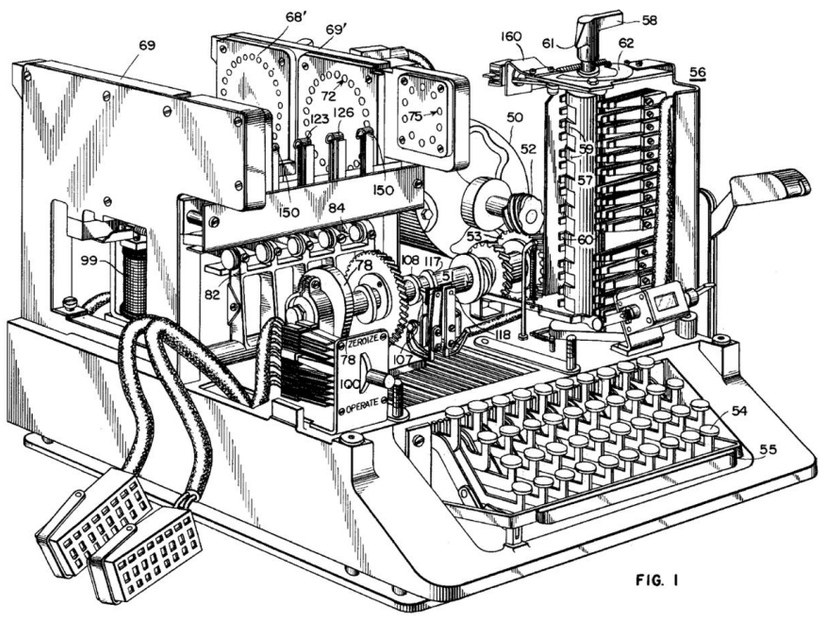
Becoming familiar with basic components of Python expressions:

- basic data types (numbers and text)
- operators, functions, and variables

This matches up with learning outcomes

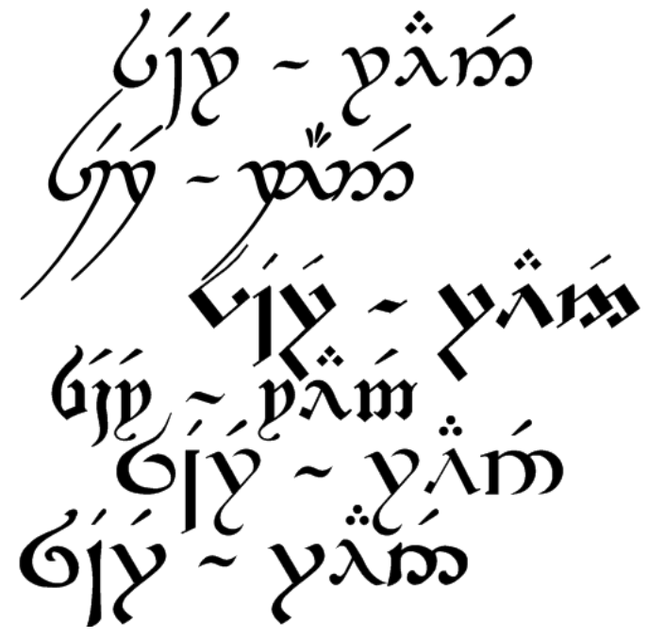
- 2, implement problem solving strategies

What is Python?



https://www.thinglink.com/scene/619900611055321088

A (virtual) Machine –
The Python Interpreter



https://www.pinterest.com.au/pin/269090146459263825/

A Language –
The Python programming language

What is Python… and why are we using it?

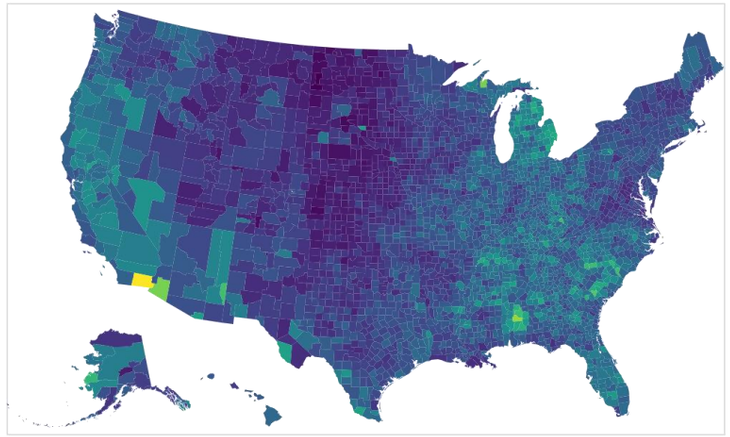
General purpose language with emphasis on readability

(created in 1990 by Guido van Rossum)

Currently in version 3.8.x
(which we use in this unit)

Very popular for a long time
(2nd most active on GitHub)

Lots of libraries available
(particularly for data science)



Altair library: https://altair-viz.github.io/gallery/choropleth.html

Where are we?

1. Describing "things" in Python
2. *Types* of things: numbers, text, and more
3. Operators and precedence
4. Variables: giving names to things
5. Functions and modules

We can describe "things" in English

a smiling man with a hot cup of tea ← expression

↓ describes

← thing (idea/concept)

Similarly, Python allows us to describe certain types of things

Not all sequences of English words are valid expressions

a with tea smiling man of cup a hot ← not a valid expression

↓ describes

?

Similarly, every programming language has a grammar

We can describe “things” literally…

three ← expression

↓ describes “literally”

3 ← thing (idea/concept)

…or indirectly

the sum of 1.7 and 0.9 rounded up ← expression

↓ describes

3 ← thing (idea/concept)

Let's describe things in Python

Simply running `python` (or `python3`) gives Python *interpreter* in interactive mode (shell)

```
Python 3.7.2 (default, Jan 13 2019, 12:50:01)
[Clang 10.0.0 (clang-1000.11.45.5)] on darwin
Type "help", "copyright", "credits" or "license" for
more information.
>>>
```

prompt

Can type in Python *expressions* to be "interpreted"

Let's start with literal expression

Simply running `python` (or `python3`) gives Python *interpreter* in interactive mode (shell)

```
Python 3.7.2 (default, Jan 13 2019, 12:50:01)
[Clang 10.0.0 (clang-1000.11.45.5)] on darwin
Type "help", "copyright", "credits" or "license" for
more information.
>>>
```

integer literal

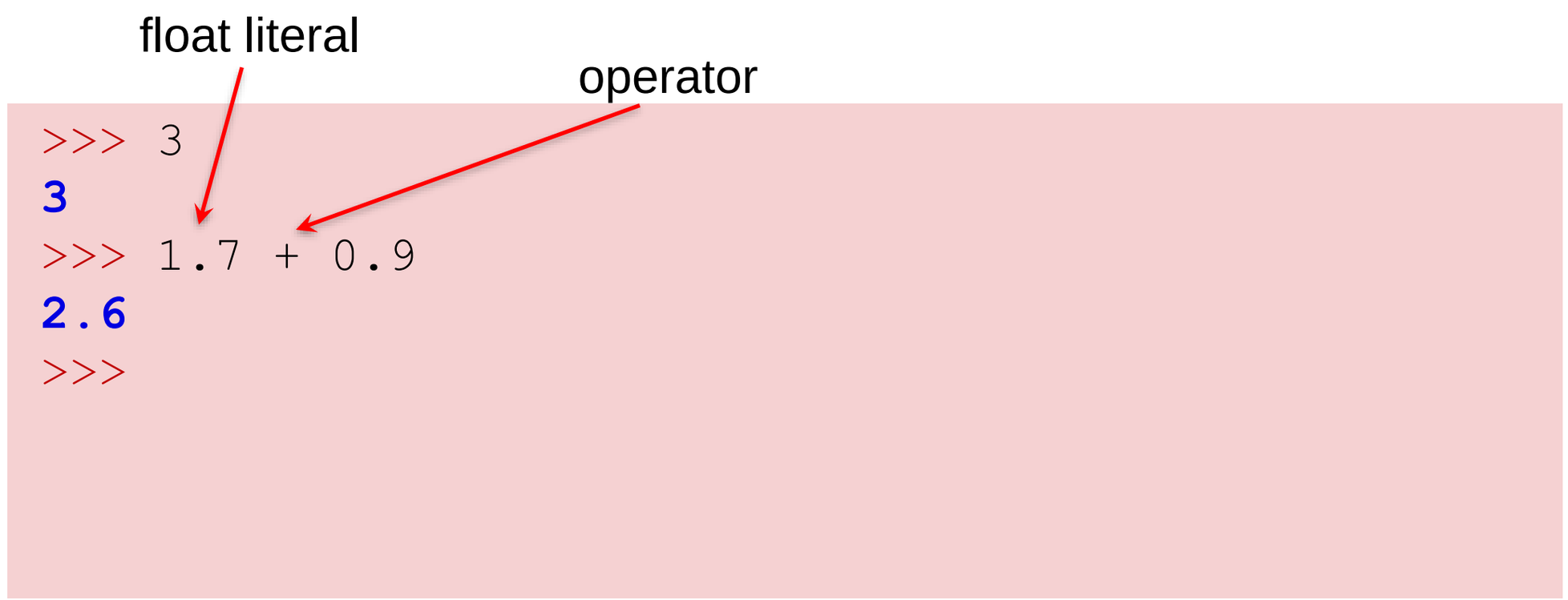
```
>>> 3
3
>>>
```

result of expression

…and gradually add complexity

Simply running `python` (or `python3`) gives Python *interpreter* in interactive mode (shell)

```
Python 3.7.2 (default, Jan 13 2019, 12:50:01)
[Clang 10.0.0 (clang-1000.11.45.5)] on darwin
Type "help", "copyright", "credits" or "license" for
more information.
>>>
```



…and gradually add complexity

Simply running `python` (or `python3`) gives Python *interpreter* in interactive mode (shell)

```
Python 3.7.2 (default, Jan 13 2019, 12:50:01)
[Clang 10.0.0 (clang-1000.11.45.5)] on darwin
Type "help", "copyright", "credits" or "license" for
more information.
>>>
```

```
>>> 3
3
>>> 1.7 + 0.9
2.6
>>> round(1.7 + 0.9)
3
>>>
```

function

…and gradually add complexity

Simply running `python` (or `python3`) gives Python *interpreter* in interactive mode (shell)

```
Python 3.7.2 (default, Jan 13 2019, 12:50:01)
[Clang 10.0.0 (clang-1000.11.45.5)] on darwin
Type "help", "copyright", "credits" or "license" for
more information.
>>>
```

```
>>> 3
3
>>> 1.7 + 0.9
2.6
>>> round(1.7 + 0.9)
3
>>> round(1.7 + 0.9) - 1
2
```

Interpreter indicates mistakes

Simply running `python` (or `python3`) gives Python *interpreter* in interactive mode (shell)

```
Python 3.7.2 (default, Jan 13 2019, 12:50:01)
[Clang 10.0.0 (clang-1000.11.45.5)] on darwin
Type "help", "copyright", "credits" or "license" for
more information.
>>>
```

```
>>> 3
3
>>> 1.7 + 0.9
2.6
>>> 1.7 0.9 +
  File "<stdin>", line 1
    1.7 0.9 +
          ^
SyntaxError: invalid syntax
```

Rule: operator has to be in-between operands (here, 1.7 and 0.9)

Interpreter tells us that we haven't followed the Python grammar (or *syntax*)

Where are we?

1. Describing "things" in Python
2. *Types* of things: numbers, text, and more
3. Operators and precedence
4. Variables: giving names to things
5. Functions and modules

“Things” in Python are *objects* of different *types*

We’ve already seen objects of types

- ‘int’ to represent whole numbers
- ‘float’ to represent decimal numbers

```
>>> type(1)
<class 'int'>
>>> type(1.0)
<class 'float'>
>>>
```

String objects represent text

Strings in Python are *sequences* of characters.
Strings are usually enclosed in *single quotes*.

```
>>> type(1)
<class 'int'>
>>> type(1.0)
<class 'float'>
>>> 'one'
'one'
>>> type('one')
<class 'str'>
>>>
```

String objects represent text

Strings in Python are *sequences* of characters.
Strings are usually enclosed in *single quotes*.
…but can also use double quotes

```
>>> "President Nixon declared 'I am not a crook'."
"President Nixon declared 'I am not a crook'."
>>>
```

single quote inside string

String objects represent text

Strings in Python are *sequences of characters*.
Strings are usually enclosed in *single quotes*.
…but can also use double quotes
…three double quotes allow entering multiline string

```
>>> "President Nixon declared 'I am not a crook'."
"President Nixon declared 'I am not a crook'."
>>> """This is a text
... spanning multiple lines."""
'This is a text\nspanning multiple lines.'
>>>
```

'\n' represents “new line”

Some operators and functions applicable to strings

```
>>> 'hello' + 'world'
'helloworld'
>>> 'hello' * 3
'hellohellohello'
>>> len('hello')
5
```

for strings, plus-operator means *concatenation*

length function

Lists are another *sequence type*

```
>>> 'hello' + 'world'
'helloworld'
>>> 'hello' * 3
'hellohellohello'
>>> len('hello')
5
```

opening and closing brackets start and end list expression

```
>>> [1, 2, 3]
[1, 2, 3]
>>> type([1, 2, 3])
<class 'list'>
>>> len([1,2,3])
3
>>> [1, 2, 3] + [4]
[1, 2, 3, 4]
```

comma delimits list elements

Sometimes we need to convert between object types

```
>>> 'The shortest route takes ' + 15 + ' minutes.'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (not "int") to str
```

Function	Result
int(x)	x interpreted as integer
float(x)	x interpreted as float
str(x)	x interpreted as string

```
>>> int(3.7)
3
>>> float(3)
3.0
>>> str(3.0)
'3.0'
>>> float('3341.10')
3341.1
```

Sometimes we need to convert between object types

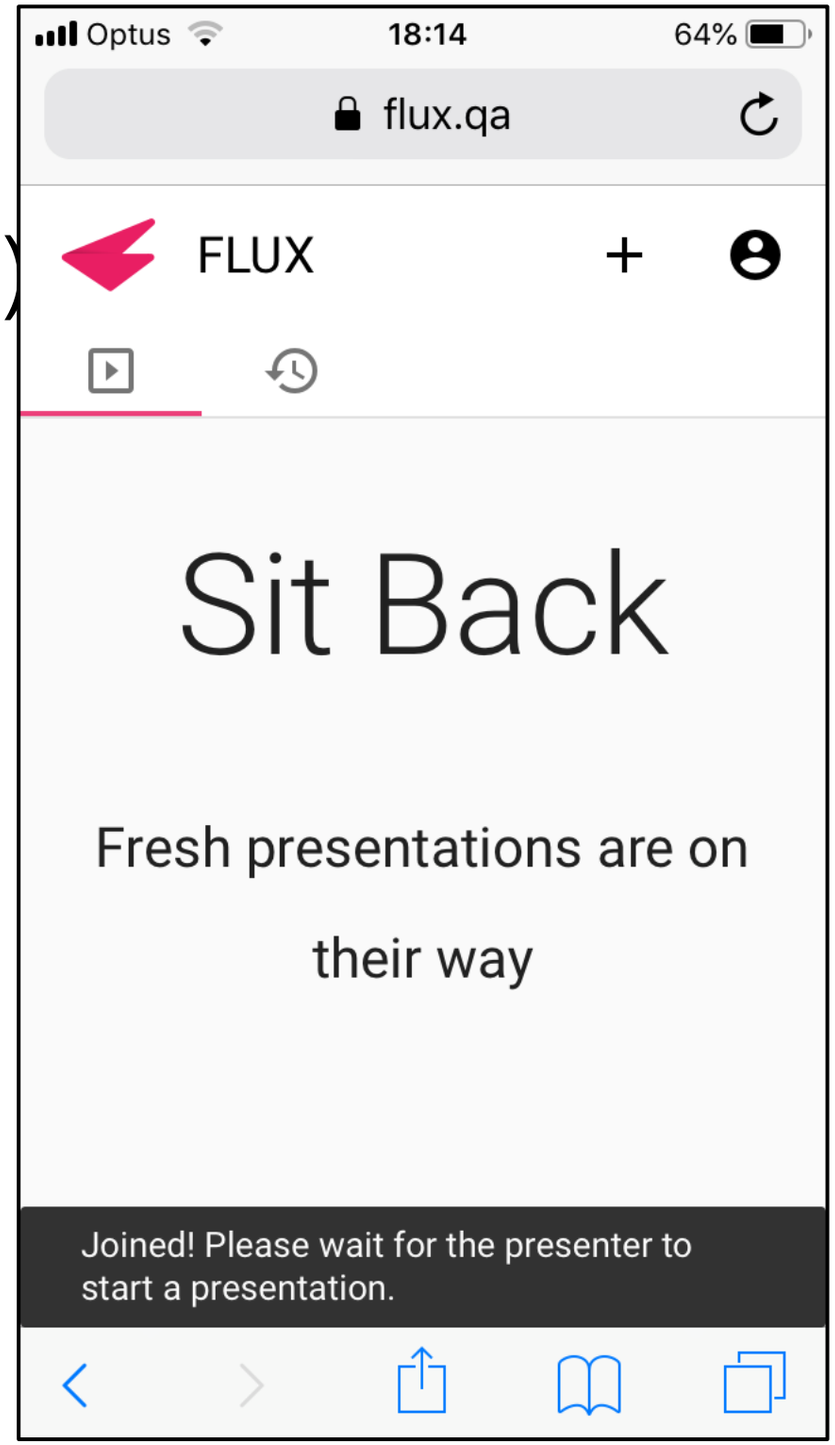
```
>>> 'The shortest route takes ' + str(15) + ' minutes.'
'The shortest route takes 15 minutes.'
```

Function	Result
int(x)	x interpreted as integer
float(x)	x interpreted as float
str(x)	x interpreted as string

```
>>> int(3.7)
3
>>> float(3)
3.0
>>> str(3.0)
'3.0'
>>> float('3341.10')
3341.1
```

Quiz time

1. Visit https://flux.qa
2. Log in (your Authcate details)
3. Touch the + symbol
4. Enter your audience code
   - Clayton: `AXXULH`
   - Malaysia: `LWERDE`
5. Answer questions



Where are we?

1. Describing "things" in Python
2. *Types* of things: numbers, text, and more
3. Operators and precedence
4. Variables: giving names to things
5. Functions and modules

Operators for numeric expressions

Operation	Description
x + y	Sum
x - y	Difference
x * y	Product
x / y	Division
x // y	Integer division
x % y	Remainder
-x	Negative
x ** y	Power

```
>>> 7 // 2
3
>>> 7 % 2
1
>>> -3.25
-3.25
>>> 2**3
8
```

Expressions are evaluated using *precedence* rules

Operation	Precedence
+, -	Lowest
*, /, //, %	Medium
**	Highest

Relative precedence of some operators

10//2 + 3**2

10//2 + 9

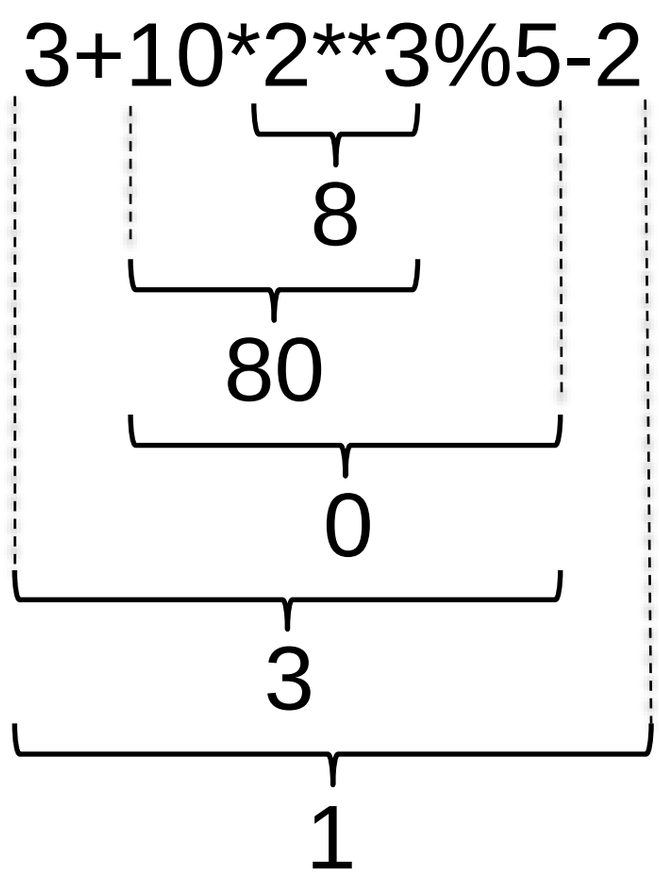
5 + 9

14

A more complicated example...

Operation	Precedence
+, -	Lowest
*, /, //, %	Medium
**	Highest

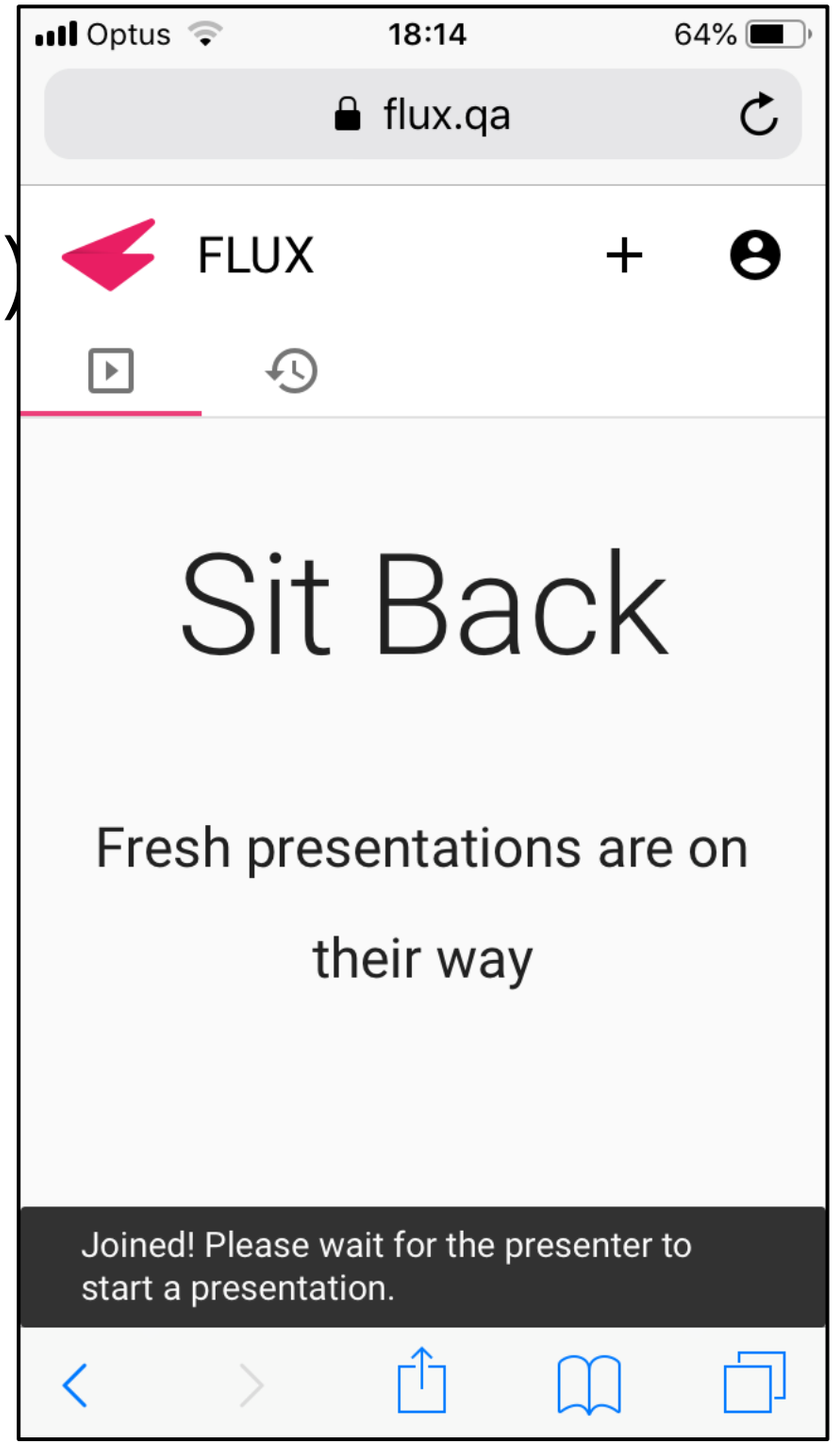
Resulting evaluation order of complex expression with intermediate result values



This expression is a bit too convoluted. In your own programs you should always try to be as clear as possible.

Quiz time

1. Visit https://flux.qa
2. Log in (your Authcate details)
3. Touch the + symbol
4. Enter your audience code
   - Clayton: `AXXULH`
   - Malaysia: `LWERDE`
5. Answer questions



Where are we?

1. Describing "things" in Python
2. *Types* of things: numbers, text, and more
3. Operators and precedence
4. Variables: giving names to things
5. Functions and modules

Variables are names for objects

variable

assignment operator

```
>>> x = 17*2
>>>
```

assigned value

no value is printed to Python console

instead variable is associated with result of right-hand-side expression

Variables are names for objects

variable

assignment operator

```
>>> x = 17*2
>>> x
34
>>> x/2
17.0
```

assigned value

after assignment:
variable can be used interchangeably with associated object

Variables are names for objects

```
>>> x = 17*2
>>> x
34
>>> x/2
17.0
```

Allows to decompose computation into *meaningful* steps.

```
>>> (10-4)*80*5/2**2
600.0
>>> x0 = 4
>>> x1 = 10
>>> weight = 80
>>> acceleration = 5/2**2
>>> work = (x1-x0)*weight*acceleration
600.0
```

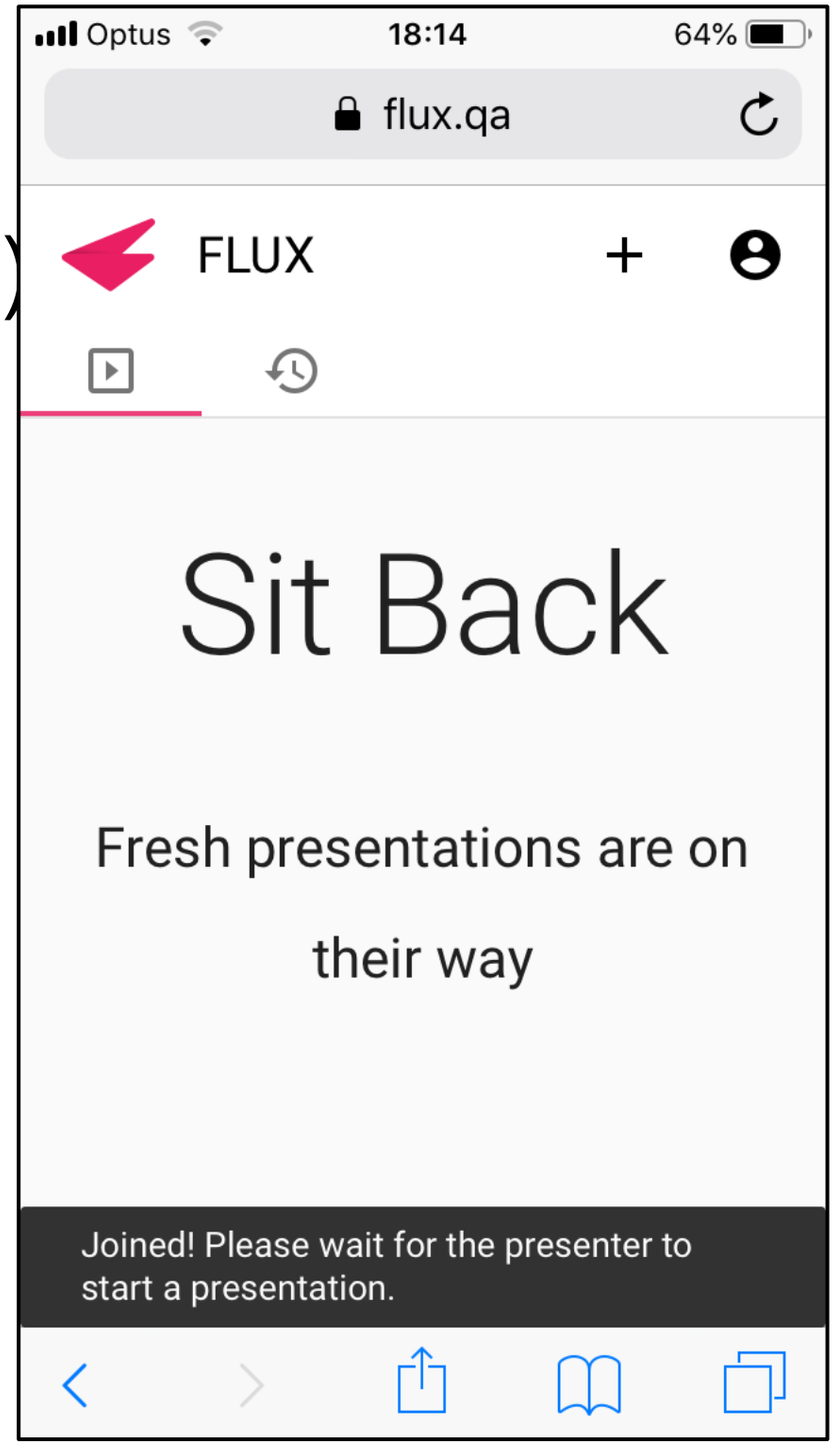
Some notes on variables

- Meaningful names are necessary to write readable programs
- Can't start names with digits
- Names are case sensitive (e.g., `price` and `Price` are different variables)
- Certain keywords are reserved by the Python and cannot be used as names

False	**break**	**else**	**if**	**not**	**try**
None	**class**	**except**	**import**	**or**	**while**
True	**continue**	**finally**	**in**	**pass**	**with**
and	**def**	**for**	**is**	**raise**	**yield**
as	**del**	**from**	**lambda**	**return**	
assert	**elif**	**global**	**nonlocal**		

Quiz time

1. Visit https://flux.qa
2. Log in (your Authcate details)
3. Touch the + symbol
4. Enter your audience code
   - Clayton: `AXXULH`
   - Malaysia: `LWERDE`
5. Answer questions



Where are we?

1. Describing "things" in Python
2. *Types* of things: numbers, text, and more
3. Operators and precedence
4. Variables: giving names to things
5. Functions and modules

Functions

function name

argument

```
>>> round(2.718)
3
>>>
```

Functions

```
>>> round(2.718)
3
>>> abs(-100)
100
>>>
```

Functions

Multiple arguments
(separated by commas)

```
>>> round(2.718)
3
>>> abs(-100)
100
>>> max(17, 20, 100)
100
>>>
```

Functions

```
>>> round(2.718)
3
>>> abs(-100)
100
>>> max(17, 20, 100)
100
>>> round(2.718, 2)
2.72
```

optional argument
(number of decimal digits to round to)

More functions available in *modules*

`math` module provides common mathematical functions

Function	Description
sqrt(x)	square root
ceil(x)	smallest integer >= x
floor(x)	largest integer <= x
factorial(x)	x! (=x(x-1)(x-2)…1)
cos(x)	cosine
sin(x)	sine
exp(x)	exponential
log(x)	logarithm

module name

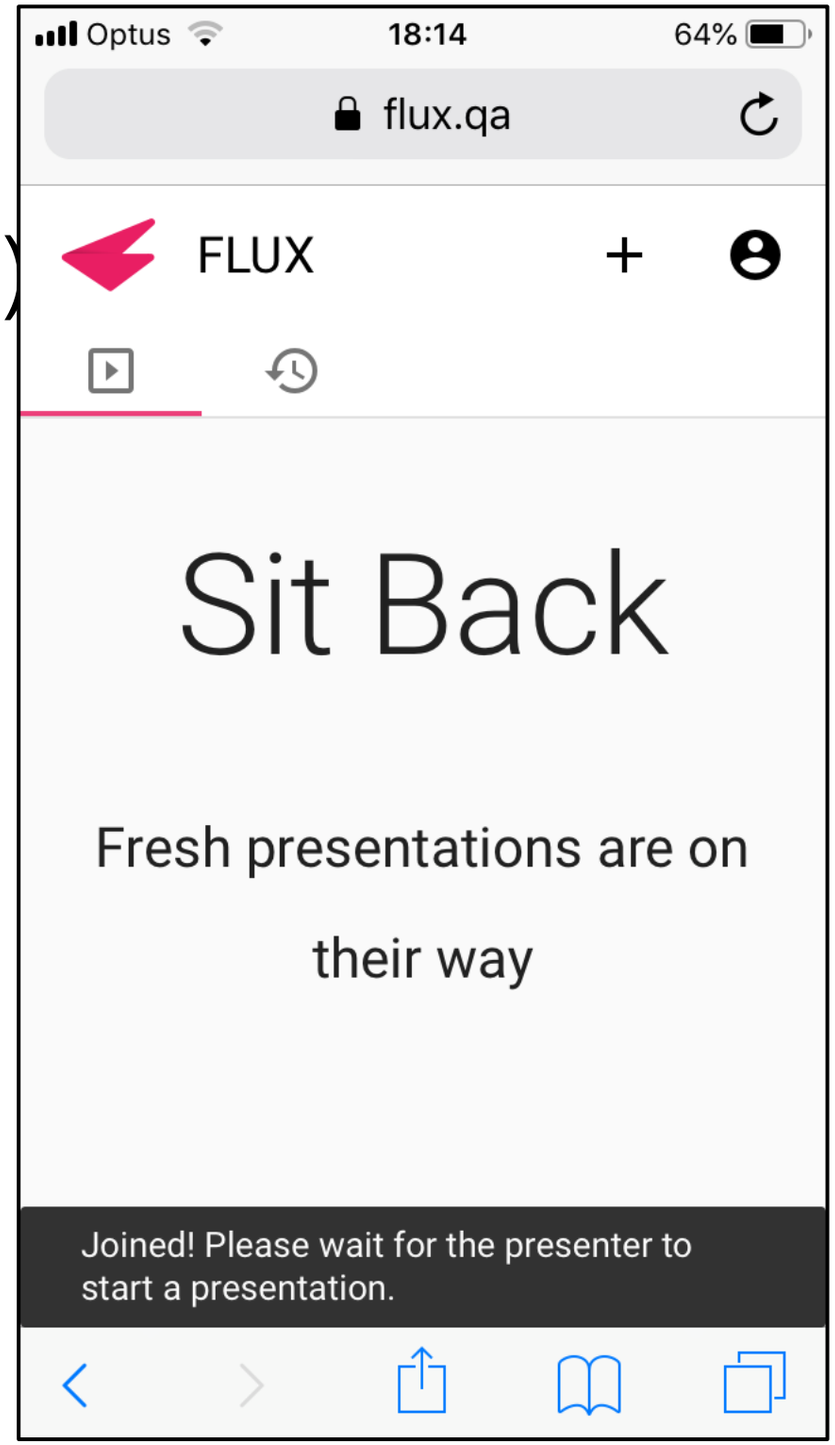
```
>>> from math import sqrt
>>> sqrt(2)
1.4142135623730951
>>> from math import floor
>>> floor(19.7)
19
```

function name

need to *import* function before it can be used

Quiz time

1. Visit https://flux.qa
2. Log in (your Authcate details)
3. Touch the + symbol
4. Enter your audience code
   - Clayton: `AXXULH`
   - Malaysia: `LWERDE`
5. Answer questions



Getting help in the shell

```
>>> help()
Welcome to Python 3.6's help utility!

If this is your first time using Python, you should
definitely check out the tutorial on the Internet at
https://docs.python.org/3.6/tutorial/.

Enter the name of any module, keyword, or topic to get
help on writing Python programs and using Python
modules.  To quit this help utility and return to the
interpreter, just type "quit".

To get a list of available modules, keywords, symbols,
or topics, type "modules", "keywords", "symbols", or
"topics".  Each module also comes with a one-line
summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type
"modules spam".

help>
```

Getting help in the shell

```
>>> help('math')
Help on built-in module math:

NAME
    math

DESCRIPTION
    This module is always available.  It provides access
to the
    mathematical functions defined by the C standard.

FUNCTIONS
    acos(...)
        acos(x)

         Return the arc cosine (measured in radians) of
x.
```

Getting help in the shell

```
>>> help(pow)
Help on built-in function pow in module builtins:


pow(x, y, z=None, /)
    Equivalent to x**y (with two arguments) or x**y % z
    (with three arguments)

    Some types, such as ints, are able to use a more
    efficient algorithm when invoked using the three
    argument form.
```

By the end of this week you should be able to…

- Explain what is a computational problem and what is an algorithm
- Write algorithms for elementary school arithmetic in "pseudocode"
- Write a Python expression for computing the area of a circular sector of angle *a* and radius *r*

More help

Some sites:

- http://www.python.org
- http://www.pythontutor.com/
- https://www.codecademy.com/learn/python

Books

- "Introduction to Computing using Python: An Application Development Focus", by L. Perkovic
- https://wiki.python.org/moin/BeginnersGuide/NonProgrammers

Recommended Reading

- "*Introduction to Computing using Python: An Application Development Focus*", by L. Perkovic, **Chapter 5**
- FIT1045/53 Workbook, **Chapter 1** (§1.1)