Python I

Yale SCHOOL OF PUBLIC HEALTH

Big Data Summer Immersion

Introduction to Python

Please log into YCRC Jupyter notebook server…

If you can’t log in, please try to look at the computer of someone sitting next to you

Programming languages for Data Science

The two most popular languages for Data Science are:

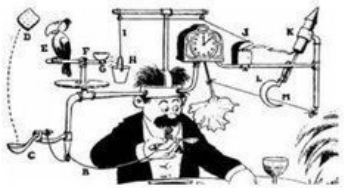
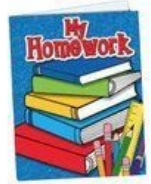
General purpose programming language

- Can do a lot more than data analysis
- Code is easy to read
- Easy to write larger software packages
- Good machine learning package (scikitlearn)
- Deep learning packages

Focused on data analysis

- Better for creating pdf reports
- Easy to create interactive apps
- RStudio created a great IDE and support

AS SEEN BY USERS OF …

	STATA	R	SAS	Python	SPSS
STATA					
R					
SAS					
Python					
SPSS					

Terminology: scripts, modules, and packages

A **script** is a piece of code that is run to accomplish a specific task

- E.g., one could write and run a script to download a set of files

A **module** is a piece of code that reused by different scripts and programs

- Modules are ***imported*** by other programs/scripts to add commonly used functionality.
- E.g., import matplotlib.pyplot as plt

A **package** (library) contains several related modules

Packages are an essential building block in programming. Without packages, you would waste a lot of time writing code that's already been written

Jupyter notebooks

Jupyter notebooks allow one to create an analysis document that contains text, analysis code, and plotted results

Because one can see all the code used to generate results, it allows one to create reproduce analyses

IF you use it right…

Programming in Python

Understanding the language fundamentals is important

Learn through practice, not only by reading or watching but by doing

- Like learning to ride a bike

Today, we will focus on basic/fundamental data types in Python. In particular, we will focus on: Numbers (Ints and Floats), Booleans (True/False), Strings, and Lists.

Expressions

Expressions

Expressions describe how a computer should combine pieces of data

- They are evaluated to by the computer and return a value
- E.g., mathematical expressions
  - Multiplication: `3 * 4`
  - Exponentiation: `3**4`

Operation	Operation	Example	**Value**
Addition	+	2 + 3	5
Subtraction	-	2 – 3	-1
Multiplication	*	2*3	6
Division	/	7/3	2.667
Remainder	%	7 % 3	1
Exponentiation	**	2**.05	1.414

Syntax

The *Syntax* of a language is its set of grammar rules for how expressions can be written

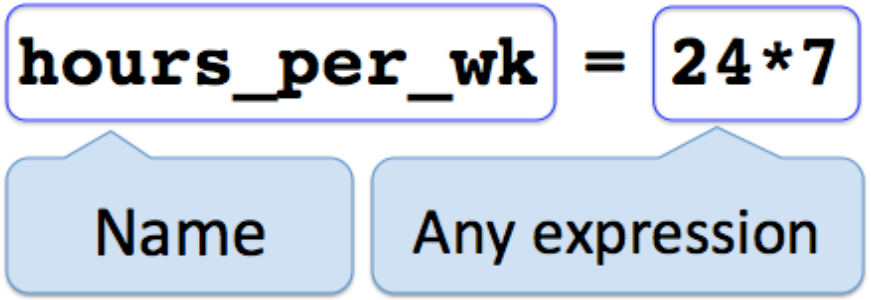
- *SyntaxError* indicates that an expression structure doesn't match any of the rules of the language.
- E.g., failed attempt at exponentiation: `3 * * 4`

```
  File "<ipython-input-2-012ea60b41dd>", line 1
    3 * * 4
        ^
SyntaxError: invalid syntax
```

Let's explore this in Jupyter!

Names

Assignment statements



Names store the values (from an expression)

- i.e., they are like variables in algebra

Names are assigned values using the = symbol

- E.g., my_number = 7

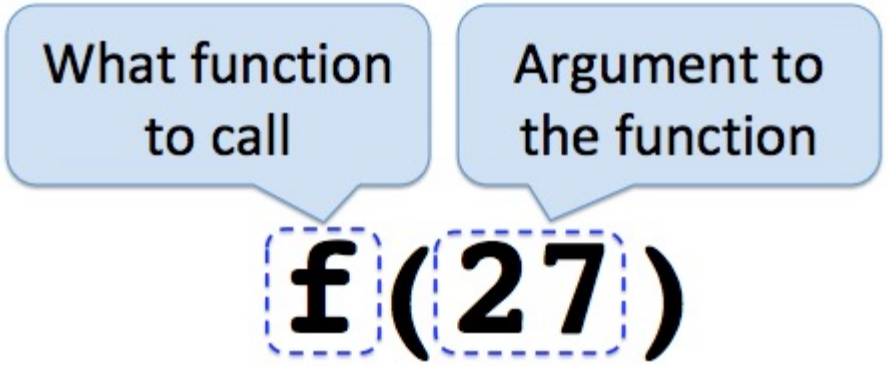
Let's explore this in Jupyter!

Call Expressions

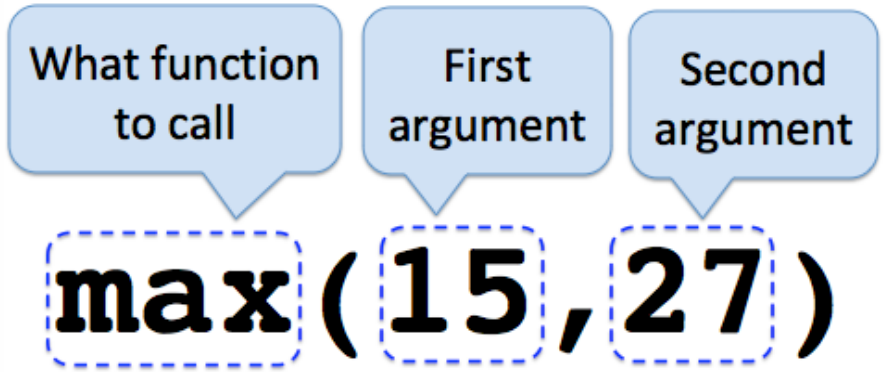
Anatomy of a Call Expression

Call expressions are expressions that call functions

- Functions take in one or more values (arguments) and (usually) return another value



Example: taking the maximum value



Let's explore this in Jupyter!

Numerical data

Arithmetic operations

Operation	Operation	Example	**Value**
Addition	+	2 + 3	5
Subtraction	-	2 – 3	-1
Multiplication	*	2*3	6
Division	/	7/3	2.667
Remainder	%	7 % 3	1
Exponentiation	**	2**.05	1.414

We can store the output of evaluating expression in names

- my_result = 10 * 2

Numbers in Python: Ints and Floats

Python has two basic number types

- **int**: an integer of any size
- **float**: a number with an optional decimal part

An int never has a decimal point - a float always does

- 3 # int of float?
- 2.7 # int of float?

A float might be printed using scientific notation

Notes on Floats

Three limitations of float values:

- They have limited size (but the limit is huge)
- They have limited precision of 15 - 16 decimal places
- After arithmetic, the final few decimal places can be wrong

Let's explore this in Jupyter!

Strings

Text and Strings

A string value is a snippet of text of any length

- 'a'
- 'word'
- "there can be 2 sentences. Here's the second!"

Strings consisting of numbers can be converted to numbers

- int('12')
- float('1.2')

Any value can be converted to a string

- str(5)

Let's explore this in Jupyter!

Types

Every value has a type

We've seen several types so far:

- int: 2
- Built-in function: abs()
- float: 2.2
- str: 'Red fish, blue fish'

The type function can tell you the type of a value

- type(2)
- type('Red fish')

An expression's type is based on its value, not how it looks

- my_text = 2
- type(my_text)

Let's explore this in Jupyter!

Conversions

Strings that contain numbers can be converted to numbers

- int('12')
- float('1.2')
- float('one point two') # Not a good idea!

Any numeric value can be converted to a string

- str(5)

Numbers can be converted to other numeric types

- float(1)
- int(1.2) # DANGER: loses information!

Lists

Lists

Lists are ways to store multiple items

We can create lists using square brackets []

- my_list = [2, 3, 4]

We can also access list items using square brackets []

- my_list[2]

Lists can contain elements of different types

- my_list2 = [5, 6, 'seven']

Let's explore this in Jupyter!