Announcements

- **Tutorial 1.1**
  - Should be submitted today (Tuesday 05/04)

- **Tutorial 1.2**
  - Should be submitted tomorrow (Wednesday 05/05)

- **Tutorial 1.3**
  - Should be submitted Friday (05/07)

- **Homework 01:**
  - Due Monday 05/10

- **Reading Week 1**
  - Due Sunday 05/09
## Updated Rubric

<table>
<thead>
<tr>
<th>Component</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participation</td>
<td>5%</td>
</tr>
<tr>
<td>4 Homeworks</td>
<td>30%</td>
</tr>
<tr>
<td>Reading reflections</td>
<td>15%</td>
</tr>
<tr>
<td>Daily Tutorials</td>
<td>20%</td>
</tr>
<tr>
<td>Final Project</td>
<td>35%</td>
</tr>
</tbody>
</table>
- Popular for data science & software development
- Mature data science and computational text analysis tools
- Learn through practice and doing
- Follow along in the demos
Names & Variables
Assignment Statements

- Statements perform an action
  - don’t have a value
- Assignment statement changes the meaning of the name to the left of the $=$ symbol
- The name is bound to a value (not an equation)

```
hours_per_wk = 24*7
```
Two real number types in Python

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

An int never has a decimal point; a float does.

A float might be printed using scientific notation.
Limitations on float values

- Floats have limited size (the limit is huge)

- Floats have limited precision of 15-16 decimal places

- After arithmetic, the final few decimal places can be wrong
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) becomes “5”
Assume you have run the following statements:

\[
\begin{align*}
&x = 3 \\
&y = '4' \\
&z = '5.6'
\end{align*}
\]

What is the source of the error in each example?

A. \( x + y \)
B. \( x + \text{int}(y + z) \)
C. \( \text{str}(x) + \text{int}(y) \)
D. \( y + \text{float}(z) \)
Types – Every value has a type

We’ve seen 5 types so far:

- int: 2
- float: 2.2
- str: ‘Red fish, blue fish’
The *type* function tells you the type of a value

- `type(2)`
- `type(2+2)`

An expression’s “type” is based on its value

- `x = 2, y="hello"`
- `type(x), type(y) = ???`
Strings that contain numbers can be converted to numbers

- `int("12")`
- `float("1.2")`
- `float("one point two")`  # Not a good idea
Any value can be converted to a string
- `str(6)`

Numbers can be converted to other numeric types
- `float(1)`
- `int(2.3)`. # DANGER: why is this a bad idea
Collections
Collections considered

Ordered:
- Lists
- Tuples

Unordered:
- Sets
- Dictionaries
 Lists

- store multiple items in a single variable
  - fruit = ["bananas", "apples", "oranges"]

- Order is preserved

- Access items with brackets
  - first_fruit = fruit[0]
  - second_fruit = fruit[1]
  - last_fruit = fruit[-1]

- What are the values assigned to these three names?
fruit = ["bananas", "apples", "oranges"]

- Access multiple items:
  - sub_fruit1 = [0:2]
  - sub_fruit1 = ???
  - sub_fruit2 = [:2]
  - sub_fruit2 = ???
  - sub_fruit3 = [0:]
  - sub_fruit3 = ???
fruit = ["bananas", "apples", "oranges"]

- **Adding at the end:**
  - fruit.append("grapefruit")

- **Modifying at a specific location:**
  - fruit[1] = "strawberry"
  - fruit ???
Tuples

- Immutable lists

- `play = ("Shakespeare", "A Midsummer Night's Dream", 1595)`

- Used to group together related data
Unordered and unindexed collection

authors = set(['Shakespeare', 'Austin', 'Morrison', 'Woolf'])

No duplicates
Dictionary

- Store data values in *key:value* pairs.
- Ordered, changeable, no duplicates
- `{"a": 1, "b": 2, "c": 3}`
Anatomy of a Call Expression

What function to call
Argument to the function

"Call f on 27."
Anatomy of a Call Expression

What function to call

First argument

Second argument

\[ \text{max} \ (15, 27) \]
## Python Built-in Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Function</th>
<th>Function</th>
<th>Function</th>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs()</td>
<td>delattr()</td>
<td>hash()</td>
<td>memoryview()</td>
<td>set()</td>
<td></td>
</tr>
<tr>
<td>all()</td>
<td>dict()</td>
<td>help()</td>
<td>min()</td>
<td>setattr()</td>
<td></td>
</tr>
<tr>
<td>any()</td>
<td>dir()</td>
<td>hex()</td>
<td>next()</td>
<td>slice()</td>
<td></td>
</tr>
<tr>
<td>ascii()</td>
<td>divmod()</td>
<td>id()</td>
<td>object()</td>
<td>sorted()</td>
<td></td>
</tr>
<tr>
<td>bin()</td>
<td>enumerate()</td>
<td>input()</td>
<td>oct()</td>
<td>staticmethod()</td>
<td></td>
</tr>
<tr>
<td>bool()</td>
<td>eval()</td>
<td>int()</td>
<td>open()</td>
<td>str()</td>
<td></td>
</tr>
<tr>
<td>breakpoint()</td>
<td>exec()</td>
<td>isinstance()</td>
<td>ord()</td>
<td>sum()</td>
<td></td>
</tr>
<tr>
<td>bytearray()</td>
<td>filter()</td>
<td>issubclass()</td>
<td>pow()</td>
<td>super()</td>
<td></td>
</tr>
<tr>
<td>bytes()</td>
<td>float()</td>
<td>iter()</td>
<td>print()</td>
<td>tuple()</td>
<td></td>
</tr>
<tr>
<td>callable()</td>
<td>format()</td>
<td>len()</td>
<td>property()</td>
<td>type()</td>
<td></td>
</tr>
<tr>
<td>chr()</td>
<td>frozenset()</td>
<td>list()</td>
<td>range()</td>
<td>vars()</td>
<td></td>
</tr>
<tr>
<td>classmethod()</td>
<td>getattr()</td>
<td>locals()</td>
<td>repr()</td>
<td>zip()</td>
<td></td>
</tr>
<tr>
<td>compile()</td>
<td>globals()</td>
<td>map()</td>
<td>reversed()</td>
<td><strong>import</strong>()</td>
<td></td>
</tr>
<tr>
<td>complex()</td>
<td>hasattr()</td>
<td>max()</td>
<td>round()</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Control Statements
Say we have a list of author names, how can we find the length of each name?
Only apply computation under certain scenario
Don’t reinvent the wheel
Installing and Accessing Python Libraries

- Install via command line:
  - pip install <Library Name>

- Access library in python:
  - import <Library Name>
Libraries we will use

- BeautifulSoup – webscraping
- NLTK – Processing text
- Spacy - Processing text
- little_mallet – Topic Modeling
- Pandas - Tables
- Matplotlib - Visualization
- Numpy – Vectors
- Sklearn – Machine Learning