Python Basics!

CS101 Lecture #4
Administrivia
Homework #2 is due Wed Oct. 19.
Data Types—A Few Points
Represent numbers with an imaginary component.

Use j for i:

\[ z = 1.0 + 1j \]
Complex numbers, $\mathbb{C}$

- Represent numbers with an imaginary component.
- Use $j$ for $i$:
  
  $$z = 1.0 + 1j$$

- $z$.real + $z$.imag * 1j
Strings

- As a literal: text surrounded by quotes.
  - "DEEP"
- Each symbol is a character.
- Unlike numeric types, strings vary in length.
String operations

- **Concatenation**: combine two strings
  - Uses the + symbol
  - 'RACE' + 'CAR'

- **Repetition**: repeat a string
  - Uses the *
  - 'HELLO ' *10

- **Formatting**: used to encode other data as string
  - Uses % symbol
Formatting operator

- Creates string with value inserted
  - Formats nicely
  - Requires indicator of type inside of string
    - "%i" int
    - "%f" float
    - "%e" float (scientific notation)
    - "%s" str
Example

```
print( "An integer: %i" % 7 )
print( "A float:    %f" % 7.0 )
print( "A float:    %e" % 7.0 )
print( "A string:   %s" % 'seven' )
```
Extracts single character
```python
a = "FIRE"
a[0]
```
The integer is the index.
Indexing operator \([\]\)

- Extracts single character
  
  ```
  a = "FIRE"
  a[0]
  ```

- The integer is the index.

- We count from zero!
Indexing operator []

- Extracts single character
  
```python
  a = "FIRE"
  a[0]
```
- The integer is the index.
- **We count from zero!**
- If negative, counts down from end.
Indexing operator 

- Extracts single character
  ```
  a = "FIRE"
  a[0]
  ```
- The integer is the index.
- **We count from zero!**
- If negative, counts down from end.
- Does this work on other data types like `int`?
Slicing operator:

- Extracts range of characters (substring)

Example:
```
a = "FIREHOUSE"
a[0:4]  # Includes character at first index, excludes last
```
Slicing operator:

- Extracts range of characters (substring)
- Range specified inside of indexing operator
Slicing operator:

- Extracts range of characters (substring)
- Range specified inside of indexing operator

```python
a = "FIREHOUSE"
a[0:4]
```
Slicing operator:

- Extracts range of characters (*substring*)
- Range specified inside of indexing operator
  ```python
  a = "FIREHOUSE"
a[0:4]
  ```
- Can be a bit tricky at first:
  - Includes character at first index
  - Excludes character at last index
alpha = "ABCDE"
x = alpha[1:3]

What is the value of x?
A 'AB'
B 'ABC'
C 'BC'
D 'BCD'
E 'CD'
Functions
A function is a small program (block of code) we can run within Python.
A function is a small program (block of code) we can run within Python.
- Saves us from rewriting code
- Don’t reinvent the wheel!
A function is a small program (block of code) we can run within Python.

- Saves us from rewriting code
- Don’t reinvent the wheel!

Analogy: Functions are more verbs.
A function is a small program (block of code) we can run within Python.
- Saves us from rewriting code
- Don’t reinvent the wheel!
- Analogy: Functions are more verbs.
- Also called subroutine or procedure.
When we want to execute a function, we call or invoke it.
When we want to execute a function, we call or invoke it.

Use name of the function with parentheses.
When we want to execute a function, we call or invoke it.

Use name of the function with parentheses.

- print()
When we want to execute a function, we call or invoke it.

Use name of the function with parentheses.

- `print()`

Many functions come built-in to Python or in the standard library.
When we want to execute a function, we call or invoke it.

Use name of the function with parentheses.

- print()

Many functions come built-in to Python or in the standard library.

Others we will compose at need.
Arguments

- Functions can act on data.
Functions can act on data.

*Arguments* are the input to functions.
Functions can act on data.

*Arguments* are the input to functions.

Functions can return a value. (fruitful)
Functions can act on data.

*Arguments* are the input to functions.

Functions can return a value. (fruitful)

Return values are the output of a function.
Arguments

- Functions can act on data.
- *Arguments* are the input to functions.
- Functions can return a value. (fruitful)
- Return values are the output of a function.
  - `print('10')`
- `len('Rex Kwon Do')`
- `abs(-123)`
Arguments

- Functions can act on data.
- **Arguments** are the input to functions.
- Functions can return a value. (fruitful)
- Return values are the output of a function.
  - `print('10')`
  - `len('Rex Kwon Do')`
Arguments

- Functions can act on data.
- *Arguments* are the input to functions.
- Functions can return a value. (fruitful)
- Return values are the output of a function.
  - `print('10')`
  - `len('Rex Kwon Do')`
  - `abs(-123)`
Functions can act on data.

*Arguments* are the input to functions.

Functions can return a value. (fruitful)

Return values are the output of a function.

- `print('10')`
- `len('Rex Kwon Do')`
- `abs(-123)`
Arguments are values passed to a function.
Arguments are values passed to a function.
A function can accept zero to many arguments.
Arguments are values passed to a function. A function can accept zero to many arguments. Multiple arguments are separated by commas:

- \texttt{min( 1,4,5 )}
- \texttt{max( 1,4,5 )}
A set of built-in functions to convert data from one type to another.
A set of built-in functions to convert data from one type to another.

- \texttt{float("0.3")}
- \texttt{str(3 + 5j)}
Type conversion.

- A set of built-in functions to convert data from one type to another.
  - `float( "0.3" )`
  - `str( 3 + 5j )`
- Be careful of nonsense:
  - `int( "Rex" )`
  - `int( 3 + 5j )`
- Also called subroutine or procedure.
- `input` is a built-in function.
- `input` is a built-in function.
- Argument: string prompting user

User input
User input

- `input` is a built-in function.
- Argument: string prompting user
- Return value: input from user (as `str`)

Functions
A program should achieve a goal.
A program should achieve a goal.
Next time we will write our first nontrivial program.
Reminders

- Homework #2 is due Wed Oct. 19.