The Python Programming Language (Part 1)

CST 205

Today's Lightening Intro to Python

- Check your installation of Python
- Check your installation of a text editor
- Introduce you to
  - Basic python data types
  - Introduce variables
  - Conditionals & Comparisons
  - Lists

What is Python?

- Python is a very high level (VHLL) dynamically typed programming language.

- Used **everywhere**

- Python 2 vs. Python 3

    - We will be using **Python 3.6** or higher

Let's Install Python 3.7.2 Together

- Go to https://www.python.org/downloads/
- Click on "Download Python 3.7.2"
- For Mac, you are better installing Python via home brew: 'brew install python' (instructions here)
- For Windows, scroll down and choose "Windows x86-64 executable installer" (here)
- **Important:** Click "Add Python 3.6 to PATH"

Install an Editor

ATOM

https://atom.io/

Sublime Text

https://www.sublimetext.com/3

Can I install my own favorite editor or IDE?

How To Run Python

1. Python interpreter, a.k.a. the interactive shell
    - Type **python** at the command line prompt.
        - Depending on your setup, you may need to type **py** or **python3** or **python3.6**
2. Use the **python** command with the name of the file (including path) as argument
    - Depending on your setup, you may need to type **py** or **python3** or **python3.6**

Our First Program

- Starting with K&R, it has been a ritual to start learning a new language by printing the phrase, “hello, world” to the command line.

- From the interpreter:
  `>>> print('hello, world')`
  - **print()** is a *function* in Python 3 (more on functions later)

- Using Atom, create a file called **hello.py** and enter the line:
  `print('hello, world')`
  - Navigate to the directory where your file lives using the command line, and execute **python hello.py**

Data Types

- Initially, we will be working with **numbers**, **strings**, and **booleans**.
    - A number can be an integer (int) or floating-point number (float)
    - A **string** (str) is a sequence of characters used to represent text.
    - Boolean values are either True or False
        - Capitalized in Python
        - Not strings!

Operations

- Typical math operations on numbers
- Concatenate strings with **+**
- Repeat strings with ***

Try these out in the interpreter

- 1 + 1
- '1' + '1'
- 1 + '1'
- 5 * 5
- 'hello ' * 5
- 'goodbye' - 'hello'
- True + ‘False'

Comments

- Comments are non-executable text in a program
- Used for:
  - Describing the purpose of a program
  - Describing a section of code
  - Temporarily disabling parts of code
- Single-line comments start with #

Variables

- A variable is like a box that holds a value.

  `x = 7`

  - Python is dynamically typed, so Python automatically assumes the type of the variable.

    - Can check with `type(x)`

- The equals sign, `=`, is known as the assignment operator.

  - Not the same functionality as the equals sign in algebra

  - Assignment operator puts the value in a “box”.

User Input

- input() prompts the user for text input.
  - This was raw_input() in Python 2
  - Value returned is a string.
- Can cast to other data type, for example:

```
your_number = int(input("Favorite number? "))
```

Comparison operator and logic

- To compare values, we use **==** for equality and **!=** for inequality
  - As expected, we also have **>**, **<**, **>=**, **<=**
- Logical **and** is **True** if and only if both sides are **True**

```
(3 != 7) and ('gr' + 'een' == 'green')
```

- Logical **or** is **True** if at least one side is **True**

```
('up' == 'down') or (1j**2 == -1)
```

Conditional Syntax

```
if condition:
    # run this block of code
elif other_condtion:
    # run this block of code
else:
    # run this block of code
```

Lists

- Store data where order is important
- Can mix data types (but probably shouldn't)
- Square brackets [] denote a list.
- Indexed at 0.
- *Mutable* (i.e., can change in-place)

List example

```
empty_list = []

big_cities = ['Shanghai', 'Beijing', 'Delhi', 'Lagos']

# access list using list[index]
# indexed at 0
big_cities[2]

# slice lists using list[start:end]
big_cities[1:3]
```

Modifying Lists

- Add elements to end of list using append()

```
big_cities.append('Tianjin')
```

- Modify existing element in-place using index

```
big_cities[1] = '北京'
```

- Obtain length of list using len()

```
len(big_cities)
```

Other List Operations

```
list.extend(other_list)

del list[index]

list.remove(element)

list.sort()

max(list)
```

Searching Lists

- Can check if an element is in a list
    - Provides a True or False value

```
if 'Monterey' in big_cities:
    print('Who knew?')
else:
    print('Monterey isn\'t one of the biggest cities.')
```

Questions?