Python Expressions
Announcements/ Logistics

- **Homework 0** due in class today
- Lab 1 for
  - Section 04 will be held today afternoon at 1 pm
  - Section 05 will be held today afternoon at 2.30 pm
  - Section 08 will be held tomorrow afternoon at 1 pm
  - Section 09 will be held tomorrow afternoon at 2.30 pm
- TA schedule is up on the course webpage
  - **Sun-Thurs 7 - 9.30 pm in TCL 217a and TCL 216**
- I have office hours today 2.30-4 pm in TBL 309B
Python and Interfaces

- Interfaces we will use to Python:
  - **IPython**
    - Interactive command-line terminal for Python
    - Created by Fernando Perez
    - Powerful interface to use Python
    - Often called a **REPL (‘Read-Eval-Print-Loop’)**
  - **Jupyter Notebook**
    - Created in 2011, a new web-based interface for Python
    - Teaching aid in class—makes teaching programming more interactive and efficient
    - Also Popular tool for scientific exposition, especially data science (even in languages such as R and Julia)
- In labs you will be writing python programs as a script with extension .py that can be executed from the terminal
Installing Python

• Checking version of Python on machine (Mac, Linux)
  • python --version

• For this class, we need Python 3.6.4 or above

• Installing Python3 on your machine
  • https://www.python.org/downloads/

• **Preinstalled on all lab machines**

• If your personal machine is Windows
  • It is possible to get everything set up
  • Lots of information online

• Initially, recommend doing lab work on machines in the CS labs
Aspects of Languages

- **Primitive constructs**
  - English: words
  - Programming languages: numbers, strings, simple operators
Aspects of Languages

- **Syntax**
  - English: “cat dog boy” (incorrect), “cat hugs boy” (correct)
  - Programming language: “hi”5 (incorrect), 4*5 (correct)
Aspects of Languages

- **Semantics** is the meaning associated with a syntactically correct string of symbols
  - English: can have many meanings (ambiguous), e.g.
    - “Flying planes can be dangerous”

- **Programming languages:**
  - Must be unambiguous
  - Can only have one meaning
  - Actual behavior can sometimes be not what is intended!
Python Program

• A program is a sequence of definitions and commands
  • Definitions are evaluated
  • Commands are executed by the Python interpreter in a shell
  • Commands instruct interpreter to do something
  • Can be typed directly in a shell or stored in a file that is read and evaluated
  • In lectures, we’ll use Jupyter for instant evaluation and output
  • In labs, you’ll write your program as a script and save it with a .py extension, e.g. `helloworld.py`. You can execute the program from the terminal: `python3 helloworld.py`
Python Primitives

- **Values:**
  - E.g. 10 (integer), 3.145 (float), ‘Williams’ (string)

- **Types:**
  - E.g. int, float, str, bool, NoneType
  - Can use `type()` to see the type of an value
  - Knowing the **type** of a value allows us to choose the right **operator** when creating **expressions**

- **Operators:**
  - E.g. `+ - * / % // =`

- **Expressions:**
  - E.g. ‘3+4’, ‘Williams’ * 3, len(‘shikha’)
  - Always produce a value as a result

- **Built-in functions:**
  - int, float, str, print, input, max, min, len
Python: Interactive Ways

“>>” tells you it is an interactive python session in the terminal

>> 1 + 2
3

>> 3* 4
12

“In [] and Out” tells you it is an interactive python session in Jupiter

In [10]: 12/3
Out [10]: 4.0

Out vs Print: “Print” means it is printed onto the console and will actually be shown to the user when you edit/run the script

In [11]: print(25//5)
5
Operator Precedence

• Operator precedence without parenthesis

    **
    *
    /
    + and - (left to right as they appear)

• Parenthesis used to override precedence and tell Python do these operations within parenthesis first
Variable Assignment

• A variable names a value that we want to use later in a program

• **Variables as a box model.**
  An assignment statement `var = exp` stores the value of `exp` in a “box” labeled by the variable name

• Later assignments can change the value in a variable box.
  **Note:** The symbol `=' is pronounced “gets” not “equals”!

In [1] num = 17
In [2] num
Out [2] 17
In [3] num = num - 5
In [4] num
Abstracting Expressions

• Why give names to values of expressions?
• To reuse names instead of values
• Easier to change code later

In [1] pi = 3.14159
In [2] radius = 2.2
In [3] area = pi * (radius**2)
In [4] area
Out [4] 15.205295600000001
In [5] round(area, 2)
Out [5] 15.21
Programming vs Math

• In programming, “we don’t solve for x”

\[
\begin{align*}
\text{pi} &= 3.14159 \\
\text{radius} &= 2.2 \\
\text{area} &= \pi \times (\text{radius}^2) \\
\text{radius} &= \text{radius} + 1 \quad \# \text{can be shortened to } \text{radius} += 1
\end{align*}
\]

An assignment: expression on the right evaluated first and the value is stored in the variable name on the left.
Built-in functions: input()

- input displays its single argument as a prompt on the screen and waits for the user to input text, followed by Enter/Return. It returns the entered value as a string.

```
In [1] input('Enter your name: ')  
Enter your name: Harry Potter 
Out [1] 'Harry Potter'

In [2] age = input('Enter your age: ') 
Enter your age: 17 
In [3] age 
Out [3] '17'
```

Prompts in Maroon. User input in blue. Inputted values are by default a string.
Built-in functions: print()

- `print` displays a character-based representation of its argument(s) on the screen and returns a special `None` value (not displayed).

In[1] name = 'Harry Potter'
In [2] print('Your name is', name)
Your name is Harry Potter
In [3] age = input('Enter your age : ')
Enter your age: 17
In [4] print('The age of ' + name + ' is ' + age)
The age of Harry Potter is 17

- Can also add spaces through string concatenation.
Built-in functions: \texttt{int()}

- When given a string that’s a sequence of digits, optionally preceded by +/-, \texttt{int} returns the corresponding integer. On any other string it raises a \texttt{ValueError} (correct type, but wrong value of that type).
- When given a float, \texttt{int} return the integer the results by truncating it toward zero.
- When given an integer, \texttt{int} returns that integer.

\begin{verbatim}
In [1] int('42')
Out [1] 42
In [2] int('-5')
In [3] int('3.141')
ValueError
\end{verbatim}
Built-in functions: `float()`

- When given a string that’s a sequence of digits, optionally preceded by +/-, and optionally including one decimal point, `float` returns the corresponding floating point number. On any other string it raises a `ValueError`.
- When given an integer, `float` converts it to floating point number.
- When given a floating point number, `float` returns that number.

In [1] float('3.141')
Out [1] 3.141

In [2] float('-273.15')
Out [2] -273.15

In [3] float('3.1.4')
ValueError
Expressions vs Statement

Expressions

• They always produce a value

  \[ 10 + 12 - 3 \]
  \[ \text{num} + 4 \]
  \[ "CS" + "134" \]

• Expressions can be composed of any combination of values, variables, and function calls

  \[ \text{max}(10, 20) \]

Statements

• They perform an action (that can be visible, invisible or both)

  \[ \text{age} = 12 \]
  \[ \text{print('Hello World')} \]

• Statements may contain expressions, which are evaluated before the action is performed

  \[ \text{print('She is ' + str(age) + ' years old')} \]

• Some statements return a None value which is not normally displayed
Error Messages

• **Type Errors**
  
  ‘134’ + 5
  
  len(134)

• **Value Errors**
  
  int(‘3.142’)
  
  float(‘pi’)

• **Name Errors**
  
  int(‘3.142’)
  
  float(‘pi’)

• **Syntax Errors**
  
  2ndValue = 25
  
  1 + (ans = 42)
Submitting Labs: Git

- Git is a version control system that lets you manage and keep track of your source code history

- **GitHub** is a cloud-based git repository management & hosting service
  - **Collaboration**: Lets you share your code with others, giving them power to make revisions or edits

- **GitLabs** is similar to GitHub but we maintain it internally at Williams and will use it to handle submissions and grading
Acknowledgments

- These slides have been adapted from:
  - http://cs111.wellesley.edu/spring19 and