The Python interpreter

CSE 140

University of Washington

Michael Ernst

Two ways to run Python

- The Python interpreter
  - You type one expression at a time
  - The interpreter evaluates the expression and prints its value
- Running a Python program
  - Python evaluates all the statements in the file, in order
  - Python does not print their values (but does execute `print` statements)
    - Writing an expression outside a statement (assignment, print, etc.) is useless, unless it is a function call that has a side effect

The Python interpreter

The interpreter is a loop that does:

- Read an expression
- Evaluate the expression
- Print the result

  If the result is **`None`**, the interpreter does not print it

  This inconsistency can be confusing!

(Jargon: An interpreter is also called a "read-eval-print loop", or a REPL)

How to launch the Python interpreter

Two ways to launch the interpreter:

- Run IDLE; the interpreter is called the "Python shell"
- Type **`python`** at the operating system command line
  - Type **`exit()`** to return to the operating system command line

These are not the same:

- Operating system command line, or "shell" or "command prompt" (cmd.exe under Windows) or "terminal"
  - Runs programs (Python, others), moves around the file system
  - Does not understand Python code like **`1+2`** or **`x = 22`**
- Python interpreter
  - Executes Python statements and expressions
  - Does not understand program names like **`python`** or **`cd`**

Running a Python program

- Python evaluates each statement one-by-one
- Python does no extra output, beyond **`print`** statements in the program

- Two ways to run a program:
  - While editing a program within IDLE, press F5 (menu item "Run >> Run Module")
    - Must save the program first, if it is modified
  - Type at operating system command line:
    `python myprogram.py`

Python interpreter vs. Python program

- Running a Python file as a program gives different results from pasting it line-by-line into the interpreter
  - The interpreter prints more output than the program would
- In the Python interpreter, evaluating a top-level expression prints its value
  - Evaluating a sub-expression generally does not print any output
  - The interpreter does not print a value for an expression that evaluates to **`None`**
    - This is primarily code that is executed for side effect: assignments, print statements, calls to "non-fruitful" functions
- In a Python program, evaluating an expression generally does not print any output

Side effects vs. results

- Some Python code is executed because it has a useful value

```
(72 - 32) * 5.0 / 9
math.sqrt(3*3 + 4*4)
```

- Some Python code is executed because it has a side effect

```
print "hello"
x = 22
```

- A function (call) can be of either variety
  - *Think Python* calls a function that returns a value a "fruitful function"
  - A function that only prints some text is non-fruitful
  - A function should either return a value, or have a side effect
    - It is bad style for a function to do both
  - Printing a value is *completely different* from returning it
- When the code is executed for side effect, its value is **`None`**