The Python interpreter

CSE 190p
University of Washington
Michael Ernst
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
Side effects vs. results

• Some Python code is executed because it has a useful value
  \[(72 - 32) * 5.0 / 9\]
  ```python
  math.sqrt(3*3 + 4*4)
  ```

• Some Python code is executed because it has a side effect
  ```python
  print "hello"
  x = 22
  ```

• A function (call) can be of either variety
  – *Think Python* calls a function that returns a function 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*
Python interpreter vs. Python program

- Running a Python file as a program gives different results from pasting it line-by-line into the interpreter.
- In a Python program, evaluating an expression generally does not print any output.
  - In the Python interpreter, evaluating a sub-expression generally does not print any output.
- The interpreter prints more output than the program would.
- The interpreter does not print a value for code that is executed for side effect: assignments, print statements, calls to “non-fruitful” functions.