Introduction to:
Computers & Programming
Print Statements and Data Types

Adam Meyers
New York University
Outline

• Print Statements in Python
  – The 'Hello World' Program
  – Print is a function that takes a string as an argument

• Data types: strings, integers, floats, etc.

• Introducing Functions, Operators and Variables
Hello World

- Custom when learning programming language
- Write a simple program that causes the string 'Hello World' to be printed to the screen.
- In Python 3
  - print('Hello World')
- Syntax of print
  - All-lowercase “print” followed by
  - Parentheses containing what is to be printed
  - Strings to be printed are surrounded by quote marks
  - Multiple items to print can be separated by commas
Now let's do it wrong

• Suppose we don't use quotes
  – Invalid syntax
  – One can only print legitimate Python objects
    • Such as strings (others will be discussed)

• Suppose we use double quotes
  – In Python, quotes, double quotes or even triple quotes are allowed
  – But they have to match

• Suppose we capitalize the first letter in “Print”
  – Lowercase “print” is the name of a function
  – Capitalized “Print” is not the name of a function
What is a function in math?

• A mapping from each element in one set (the domain) to exactly one element in another set (the range)

• Examples
  – The square of a number
    • Given any number (the domain), there is exactly one square in the range, a subset of the set of numbers
  – The shoe size of a human being
    • For each member of the human race, there is exactly one shoe size (ignoring differences between brands, etc.)

• In Python (and other languages), a function is not exactly the same thing, but sort of
Functions, Procedures, Subroutines, Methods, etc.

- In Programming, all these terms are used to describe essentially a command, defined in terms of a set of statements of that language.
- A Function can provide a mapping from input (domain) to output (range)
  - Like the mathematical definition
  - Input can be a set of zero or more items
  - Output can be a set of zero or more items
- In programming, functions can have side effects
In Python 3.0, “print” is a function

- **Input**: zero or more objects as input
  - Zero strings causes a newline to be printed
- **Output**: None (a special object)
  - Means something like 'no value'
  - Programmers don't normally use the output of 'print'
  - In some languages (C,C++), such functions are called **void functions**

- **Side effect**
  - it prints out the input strings
    - In IDLE, it prints them on a line
  - The side effect is the important feature of this function
Data Types

• There is a function “type” that maps objects to their data type

• Use this function in IDLE as follows:
  • type('Hello')
  • type(“Hello”)
  • type(5)
  • type(5.4)
  • type(print('hello'))
    – Notice the side effect of this statement

• Built in data types:
  • Numeric types: integer, float, complex
  • Sequence types: string, list, tuples
  • Other: Boolean, NoneType (the type of None), ...
Strings in Python

- A string is a sequence of characters surrounded by:
  - Single quotes 'string'
  - Double quotes “string”
  - Triple quotes """string"

- Embedding 1 kind of quote within another:
  - 'the string “Python 3.1” labels this language'
  - “the string 'Python 3.1' labels this language”

- Newlines can be embedded in triple quotes:
  - """These two strings, 'Python 3.1' and “Python 3.1”, are on separate lines"

- Embedded strings are called 'string literals'
Numeric Data Types

- **Integer**
  - Whole numbers (positive and negative)
  - -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5

- **Float (decimal numbers)**
  - -2.305, -0.1334, 0.33333333, 14.789, 5.0

- **Many-digit floats are shortened** (all programming languages do this since floats can be infinitely long).
  - Long decimals are truncated, rounded or written in scientific notation
  - Try: .1111111111111111111111119 and 1111111111111111111111111.9

- **Complex**
  - Sum of floats and multiples of imaginary numbers
    - \( j = \sqrt{-1} \)
    - 4.1+3j, 1+45j, …
Operators are Functions with the Syntax of Arithmetic

<table>
<thead>
<tr>
<th>Operation</th>
<th>Standard Arithmetic</th>
<th>Python</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td>$5 + 3$</td>
<td>5+3</td>
<td>8</td>
</tr>
<tr>
<td>Subtraction</td>
<td>$5 - 3$</td>
<td>5-3</td>
<td>2</td>
</tr>
<tr>
<td>Multiplication</td>
<td>$5 \times 3$ or $5 \times 3$</td>
<td>5*3</td>
<td>15</td>
</tr>
<tr>
<td>Division</td>
<td>$5 \div 3$ or $5 / 3$</td>
<td>5/3</td>
<td>1.6666666</td>
</tr>
<tr>
<td>Exponents</td>
<td>$5^3$</td>
<td>5**3</td>
<td>225</td>
</tr>
<tr>
<td>Integer Division</td>
<td>$5 \div 3$ or $5 / 3$</td>
<td>5//3</td>
<td>1</td>
</tr>
<tr>
<td>Modulus</td>
<td>$5 \mod 3$</td>
<td>5%3</td>
<td>2</td>
</tr>
</tbody>
</table>
Operator Syntax vs. Function Syntax

- Each operator takes 2 arguments
  - Syntax of Function:   F(arg1, arg2)
  - Syntax of Operator:    arg1 Op arg2

- Python (and other languages) adopt this syntax because it is intuitive due to our education

- Other languages like LISP adopt Polish Notation
  - (+ 5 3), (-5 3), (* 5 3), (expt 5 3), (mod 5 3)
Some Math from Grade School You May Have Forgotten About

• Order of Operations: PEMDAS
  – Parentheses, Exponents, Multiplication/Division, Addition/Subtraction
  – What does 2+5*2 equal?
    • 12 or 14?

• Integer Division and Modulus in Python
  – 5 | 3 == 1 r 2
    • 5//3 → 1 (integer division)
    • 5%3 → 2 (modulus)
Ambiguity without PEMDAS

\[2 \times (5 + 2) = 14\]

\[(2 + 5) \times 2 = 14\]
Ambiguity gets worse as expressions get longer
Use Parentheses instead of PEMDAS

- Parentheses eliminate ambiguity from arithmetic
  - \((2*5)+2 == 12\)
  - \(2 * (5+2) == 14\)
  - \((2**(5+2)) * 7 == 896\)
  - \(2**((5+2) * 7) == 562949953421312\)

- The PEMDAS defaults also eliminate ambiguity
  - But for computers, not for most people
The Modulus Operator

• Modulus gets the remainder from division
  – 100 % 12 == 4
  – 52 % 12 == 4
  – 28 % 12 == 4

• Example Application
  – Musical Instrument Digital Interface (MIDI)
    • MIDI notes begin with 0 (4 octaves below middle C)
    • MIDI notes end with 127 (G about 5 ½ octaves above middle C)
    • The statement 'MIDI-NUMBER % 12' identifies note classes
      • 0, 12, 24, 36, 48, 60, 72, 84, 96, 108, 120 are all C
      • 1, 13, 25, 37, 49, 61, 73, 85, 97, 109, 121 are all C#
      • 2, 14, 26, 38, 50, 62, 74, 86, 98, 110, 122 are all D
      • Etcetera ...
Operations on Strings (and 1 function)

- You can add (concatenate) strings
  - 'hello' + 'hello' == 'hellohello'
  - 'hello' + ' ' + 'hello' == 'hello hello'

- You can multiply (repeat) strings by an integer
  - 'hello' * 3 == 'hellohellohello'
  - 'hello ' * 3 + 'hello' == 'hello hello hello hello hello'
  - ('hello ' * 3) + 'hello' == 'hello hello hello hello hello'
  - What about 'hello ' * (3 + 'hello') ?

- You cannot use / or ** with strings

- len('spam') – function indicating length of string
Conversion to Type String

• The `str` function converts objects to strings
  – `str(5)`
  – `str(5.1)`
  – `str(print('hi'))`
  – Any Python object can be converted to a string
    • This depends on the definition of that type of object

• The print function converts objects to strings before printing them
  – This also depends on the object's definition
Printing Non-strings

• For numbers, this is pretty obvious (try it)
  – `print(5, 6, 7) ## printing integers`
  – `print(5.5, 100.1) ## printing floats`

• For other objects, this is not so obvious
  – `Print(1+ 5j) ## it adds ( ) around complex numbers`
  – `print(print(5)) ## prints the 5 as a side effect and then
    ## prints the output of print as 'None'`

• Definitions of types of objects can include printing
  instructions, how they are represented as a string,
  and many other non-obvious properties
Comments

• A comment is a statement that is not evaluated
• In Python (and many languages)
  – Everything following “#” is a interpreted as a comment
  – For example, 
    ```python
    print('Hello World')  # This will print Hello World
    ```
  – Comments are used by programmers to help make their code readable (by themselves, by others, etc.)
  – Comments can occur after code or on lines by themselves
    ```python
    ## This is a comment – it will do nothing
    ```
Conversion to Integers and Floats

• The `int` function converts floats and some strings to integers
  – `int(5.1)` ## OK
  – `int('5')` ## OK
  – `int('five')` ## will cause an error

• The `float` function converts integers and some strings to floats
  – `float(5)` ## OK
  – `float('5.1')` ## OK
  – `float('5')` ## OK
  – `float('five')` ## will cause error
Other Python Functions/Operators
Convert Integers to Floats

• “Normal” division
  – 4/2 == 2.0
  – 5/2 == 2.5
  – 1/3 == 0.3333333333333333

• But Not Integer Division
  – 4//2 == 2
  – 5//2 == 2

• Some of the functions in the math module
  – math.ceil, math.floor, math.trunc
Summary 1

• Functions map input to output and sometimes have side effects
  – This is similar to the mathematical concept of a function
  – Operators are like functions, but they have different syntax

• The Python 3 function 'print' displays strings as a side effect, the output of the function is unimportant

• Using functions and operators correctly requires using the proper syntax (spelling, punctuation, indentation, …)
Summary 2

- The print function prints the string version of an object as a side effect.

- Types are categories of objects:
  - An operator/function may handle different types differently:
    - 'Hello ' + 'World' vs. 5 + 2
    - Some operators/functions are only defined for certain types:
      - ** and / are defined for integers and floats, not strings.

- Programming languages (and math) employ rules of precedence to make sure that statements are unambiguous:
  - Such rules, e.g., PEMDAS, sometimes make it difficult for human beings.
  - However, parentheses can make programs readable for both humans and computers.
Homework Due Soon

• Read chapter 2 in Donaldson book

• Open an Idle shell and do exercises on the next slides
  – If an exercise involves writing code, write code that solves that problem and run it.
  – If a question requires an explanation, write that explanation as a comment (preceded by #)
  – When you are done, save the contents of the IDLE window (see File menu).

• Name this file: Homework2-your-name.txt

• Submit the resulting file using Blackboard.
Homework Exercises 1

• Use the print function to print the following strings, including the embedded quotes and blank lines:
  – *The newscaster said, “And Now for Something Completely Different!”*
  – *One quote: ', Two quotes “, Red Quotes, Blue Quotes*

• Evaluate the following expressions and explain (as comments) why they yield different answers:
  – '5' + '4'
  – 5 + 4

• Incorporate type conversion into a statement so that you combine '5' and '4' to get '9'
Homework Exercises 2

- Evaluate each of the following expressions. Then make a new version of each expression incorporating parentheses that make the order of operations explicit. Evaluate this second expression (you should get the same result).
  - Example: 1*1**1+1 and (1*(1**1))+1 both evaluate as 2
  - 5*5/5–5
  - 5-5**5*5
  - 60-40*1.5+5**2-25

- Evaluate the following expressions:
  - 3%3, 4%3, 7%3, 25%3, 98%3, 137%3, 1997%3, 1313%3 and 10011%3
  - Make a list (as a comment) of 3 sets: the set of numbers such that X%3 = 0, the set where X%3 = 1 and the set where X%3 = 2
Important Note

• This is the First and Last time that we will be handing in Idle Output (saved as a .txt file)
• In the future, we will start writing program files (scripts) and saving them as .py files.