Introduction to: Computers & Programming: Print Statements, Data Types, Functions, Variables

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, asymmetric left/right feet, 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
  – Stuff that happens before output is returned, e.g.,
    • Tangential to output generation, e.g., helpful print-outs, sounds, images
    • Bookkeeping processes that effect other parts of complex program
In Python 3, “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
A Function which both returns output and has a side effect

- def add_two_numbers(number1, number2):
  print('Adding',number1,'and',number2)
  sum = number1 + number2
  print('The result is', sum)
  return(sum)

- The 2 print statements produce side effects
- The return statement ends the function and causes it to return output.
Data Types

• There is a function “type”
  – An object of any type is the input argument
  – The **data type** is the output

• 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""
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: .1111111111111111111119 and 111111111111111111111.9

• 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 X 3 or 5 * 3</td>
<td>5*3</td>
<td>15</td>
</tr>
<tr>
<td>Division</td>
<td>5 ÷ 3 or 5 / 3</td>
<td>5/3</td>
<td>1.6666666</td>
</tr>
<tr>
<td>Exponents</td>
<td>5³</td>
<td>5**3</td>
<td>225</td>
</tr>
<tr>
<td>Integer Division</td>
<td>5 ÷ 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)
  – In such languages, there are no operators, only functions
Some Math from Grade School You May Have Forgotten About

• Integer Division and Modulus in Python
  – \( 5 \div 3 = 1 \text{ r } 2 \)
    • \( 5/3 \rightarrow 1 \) (integer division)
    • \( 5\%3 \rightarrow 2 \) (modulus)

• Order of Operations: PEMDAS
  – Parentheses, Exponents, Multiplication/Division, Addition/Subtraction
  – What does \( 2+5*2 \) equal?
    • 12 or 14?
Ambiguity without PEMDAS

```
2 + (5 * 2)
```

```
(2 + 5) * 2
```
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,
    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
    ## This is a comment – it will do nothing
Conversion to Integers and Floats

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

• The \texttt{float} function converts integers and some strings to floats
  – \texttt{float(5)} ## OK
  – \texttt{float('5.1')} ## OK
  – \texttt{float('5')} ## OK
  – \texttt{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
Writing a function in Python

• `def function_name (arg1, arg2, ...):`
  stuff
  more stuff
  return(output)

• Example:

  `def add_2_numbers(num1, num2):
    print('num1:',num1,'num2:',num2) ## side effect
    sum = num1 + num2
    print('The sum is:', sum) ## side effect
    return(sum) ## what the function returns`
Some Rules for Names and Variables

- Variables and Functions have Names
- Rules guiding Names
  - They can consist of letters, numbers and underscore __
  - They cannot begin with a number
  - They cannot be a special word reserved by Python:
    - and del from not while as elif global or with assert else if pass
      yield break except import print class exec in raise continue
      finally is return def for lambda try
- Be careful not to use the same name twice
- Scope
  - Parameters and variable defined inside a function, only work in that function – these are both instances of local variables
  - Variables defined outside functions are called global variables
  - Warning: Do not use the same name for a global variable and a local variable: we will discuss this more later in the term
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.
Zip Archive Files: a Technical Detail

• This has nothing to do with Python
• When people need to send each other multiple files, it is standard to use file archiving software:
• The zip format is probably the most common format for archiving
• If you do not know how to collect a group of files into an archive, please look it up and/or seek help during the lab class or from one of the tutors.
Homework 2: Due Before 5th Class

• Work on homework exercises in lab session (next class) & at home.
• Open an Idle shell and do exercises 1—5
  – 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 contents of IDLE window (see File menu).
  – It is possible that you will end up creating multiple output files of this type.
  – Name these files from idle:
    • Homework2a-your-name1.txt, Homework2a-your-name2.txt, ...
• When instructed create a .py file and name it: Homework2b-your-name.py or some other name that includes your name and homework 2 in it. This file will contain the answers to exercises 6—8.
• You should combine all your homework files into one zip archive and Submit it through NYUClasses.
Homework 2 Exercises 1

1 Use the print function to print the following strings, including the embedded quotes and blank lines:

A The newscaster said, “And Now for Something Completely Different!”

B One quote: ', Two quotes “, Red Quotes, Blue Quotes

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

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

4 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: \(1 \times 1^{1+1} \) and \((1 \times (1^{1+1})) + 1\) both evaluate as 2

A \(5 \times 5/5 - 5\)
B \(5 - 5^{2} \times 5\)
C \(60 - 40 \times 1.5 + 5^{2} - 25\)

5 Evaluate each of 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 \(X \% 3 == 2\)

- Since these are infinite sets, list the first 8 members of each set and then use 3 dots to indicate that the set is not completed, e.g., the set of all positive integers can be represented as \(\{1, 2, 3, 4, 5, 6, 7, 8 ...\}\)
Homework 2 Exercises 3

- In Idle, go to the file menu and open a new window. Write a program according to the following set of instructions in that window and save as “Homework2b-your-name.py”

- Create the 3 functions described on the next page and include code that executes the functions when you run the program. For example, if you define a function called “print_that_4_times”, that takes one argument, you should include a line:

  ```python
  print_that_4_times('chicken')
  ```

  in your file after you define the function.

- All function should include comments. Function names and variable names should make the nature of the function as clear as possible.
6  Function 1:
   – The function should take exactly 3 arguments and computes an average of the 3 arguments.
   – It should return the average as output
   – Include a comment stating what a better average function might do.

7  Function 2:
   – The function takes one argument and prints the following, each on a separate line:
     • “You entered:” (without the quotes)
     • The argument itself
     • The argument three times on the same line.

8  Function 3 – Bonus Question (Extra points if you get this one to work)
   – Write a function that adds two averages together.
   – It takes 6 arguments: number1, number2, number3, number4, number5, number6
   – It calls function 1 twice and adds the results together. It should return this result
     • It calls function 1 with number1, number2, number3 as arguments.
     • It calls function 1 with number4, number5 and number6 as arguments.
     • Then it adds the results together and return the result as output.
Homework 2 – Due Before 5th Class

• Read Chapter 3 in Gaddis Book
• Do online Module 3 on website
• Do Quiz 3 in NYUCourses