CSE 453

Python Basics

Running a Program in Python

- Interactive Mode
  - Command to invoke Python on *timberlake*
    - python
  - Prompt
    - >>>
  - At prompt, enter instruction(s)
  - To Exit
    - Control-D
    - quit()
    - exit()
- Script Mode
  - Command to invoke Python on *timberlake*
    - python *script.py*
      - ✓ Where *script.py* is the name of your script (program)
      - ✓ Python programs should have the *.py* extension

Some Syntax Rules

- Python is case sensitive
- No punctuation at end of line

Comments

- Line Comment
  - #
  - Everything after the # on a line is considered a comment
- Paragraph (Multiline) Comment
  - '''
    - Three single quotes
  - Everything between ''' and ''' is a comment

Data Types

- Types
  - Integer
    - int
  - Float
    - float
  - String
    - str
- Automatic Type Casting

Variables

- Declarations NOT Required
  - Variable must be assigned a value before it is used
- Automatic Type Casting
  - Type determined when a value is assigned to a variable
  - Examples
    - `price = 37.92`
    - `course_number = 508`
    - `model = "dynaflow"`
    - `bore = 24.0`
      - ✓ `Note that despite the '.0', this is still assigned a float`
  - A variable may be reassigned a different data type at any time
    - Example

```
score = 95.2   # Assigned as a float
score = 85     # Reassigned as an integer
```

- Naming Rules
  - No key words
  - Case sensitive
  - No spaces
  - First character must be a letter or an underscore
  - After the first character, numbers may be used
- Determining the data type of a variable or value
  - Built-in Function: type(*expression*)
  - *expression* is the type of data the variable or value that is being tested
  - Example

```
>>> std_dev = 37.5
>>> type(std_dev)
<class 'float '>
```

- Type Conversion
  - Despite the fact that Python has automatic type casting, there may be times when data must be forced to a particular type
    - Especially useful when inputting data
      - ✓ Data input from console is a string
        - May need to be converted to a float or an int if it is numeric data
  - Forced Type Casting
    - Integer
      - ✓ int(item)
    - Floating Point
      - ✓ float(item)
    - String
      - ✓ str(item)
    - ✍ An exception occurs if the item is not valid data for the type you are attempting to convert it to
      - ✓ Example
        - int('Hello') results in an exception
- Garbage Collection
  - When a variable is assigned a new value, the old value is removed from memory

Assignments

- Operator
  - =
- Format
  - *variable = expression*

Mathematical Operators

Operator	Operation
+	Addition
-	Subtraction
/	Floating Point Division
*	Multiplication
//	Integer Division
**	Exponentiation
%	Modulus (Remainder)

Python 2 vs. Python 3

- Integer Division
  - Integer / Integer
    - Python 2
      - ✓ Integer returned
    - Python 3
      - ✓ Float returned
    - Prevention
      - ✓ float(x)/y
        - Causes Python 2 & 3 to act similarly
      - ✓ Use floor function (defined later under math functions) to force the result to be an integer with the same quotient in both versions

Mathematical Operators

- Unary Operators
  - +
  - -
- Order of Precedence
  - Same as standard arithmetic
  - Parenthesis can be used to override precedence
- Rounding & Truncation in Integer Division
  - Positive quotient is truncated
  - Negative quotient is rounded away from zero

Mixed Type Expressions & Data Type Conversions

- Two Integer Operations → Integer
  - An operation on 2 integer values results in an integer
- Two Float Operations → Float
  - An operation on 2 floating point values results in a floating point
- Mixed Type Operations
  - Operations on an integer value and a floating point value
    - Integer temporarily converted to a floating point
    - Result of the operation is a floating point

Overflow/Underflow

- Overflow
  - Python will report if a value overflows
- Underflow
  - Applicable to floating point numbers
  - Python approximates an underflowed value to zero

Augmented Assignment Operator

- Placing the operator in front of the assignment operator (=) creates an augmented assignment operator
- Format
  - *variable* operator = *expression*
  - The above expression is equivalent to
    - *variable* = *variable* operator *expression*
- Example
  - sum += 49
    - Equivalent to
      - ✓ sum = sum + 49

Simultaneous Assignments

- Multiple variables can be assigned a value with a single line of code
- Syntax
  - var1, var2, …, var*n* = exp1, exp2, …, exp*n*
    - where
      - ✓ *var* is a variable
      - ✓ *exp* is an expression
- Example
  - x, y = y, x
    - Swaps the value of *x* and *y*

Built-in Functions

Function	Description
print(msg)	Displays msg to the console, where msg may be a list consisting of strings and/or variables
input(prompt)	Gets user data from the console, prompting the user with the string argument prompt
eval(string)	Evaluates the expression that is stored in a string and returns the evaluated expression as an integer. Note: An error is returned if a numeric literal has a leading zero(s)
ord(c)	Returns the ASCII code for the character c
chr(code)	Returns the character represented by the code
max(x1, x2, …)	Returns the largest value among x1, x2, …
min(x1, x2, …)	Returns the smallest value among x1, x2, …
pow(a,b)	Returns a^b. Same as a**b.
round(x)	Returns the integer closest to x. If x is equally close to two integers, the even integer is returned.
round(x,n)	Returns the float value rounded to n digits after the decimal point

Derived from Table 3.1, page 64, of Y. Daniel Liang, *Introduction to Programming Using Python*, Pearson Education, Inc. (Prentice Hall), 2013

Python 2 vs. Python 3

- print
  - Python 2
    - Print is a statement
      - ✓ Parenthesis NOT Required
        - ✓ They are allowed, though
          - Care must be taken if using parenthesis so that the objects being printed are not treated as a tuple
  - Python 3
    - Print is a function
      - ✓ Parenthesis Required
- round()
  - Python 2
    - round() function rounds away from zero when a number is at the halfway point
    - Float returned
  - Python 3
    - round() function rounds towards the nearest even when a number is at the halfway point
    - May return integer or float
      - ✓ Use of the second argument can ensure the desired type is returned
      - ✓ Zero will cause a float to be returned

- Python 2
  - Example
    - ✓ Code:

```
x = 4.5
y = round(x)
print y
```

    - ✓ Output:

```
5.0
```

- Python 3
  - Example
    - ✓ Code:

```
x=4.5
y=round(x)
print (y)
y=round(x,2)
print (y)
```

    - ✓ Output:

```
4
4.5
```

Example

- Python 3
  - Code

```
x = 3
y = 7.5
print(eval('x + y'))

maximum = max(x, y, 6)
print(' Maximum is ',maximum)
```

  - Output

```
10.5
Maximum is 7.5
```

- Python 2
  - Code

```
x = 3
y = 7.5
print eval('x + y')

maximum = max(x, y, 6)
print 'Maximum is ',maximum
```

  - Output

```
10.5
Maximum is 7.5
```

Getting User Input

- Built-in Function: input(*prompt*)
- *prompt* is a string that prompts the user for input
- Input is a string
- Numeric data needs to be converted to appropriate type

Outputting to the Console

- Built-in Function: print(*argument list*)
- *argument list* can be a combination of string literals, variables, and special arguments
  - Arguments are separated by commas
- String Literals
  - Enclosed by
    - ′
      - ✓ Single quote
    - ″
      - ✓ Double quote
      - ✓ Useful if the string contains an apostrophe
    - ‴
      Three single quotes
      - ✓ This is the most versatile
      - ✓ Allows double quotes and single quotes to be in the string
- The newline character is appended to the end of what is displayed by default

Python 2 vs. Python 3

- Special Arguments
  - end
  - sep
- Python 3
  - Special Arguments
    - end=″
      - ✓ By default the print function outputs a newline character at the end of the output
      - ✓ Using this special argument replaces the newline character by what is inside the quotes
        - If nothing is inside the quotes, the newline is simply not appended to the output
    - sep=″
      - ✓ Separates each argument by value in quotes when output
  - Examples
    - print (′Data Recorded Successfully′)
    - print (width, ′ * ′, height, ′ = ′, area, end=″)
    - print (sensor1, sensor2, sensor3, sep= ′,′, end=″)

- Python 2
  - ☞ The special arguments sep & end have no meaning in Python 2
    - ↳ Suppressing the endline character at the end of a Python2 print statement can be accomplished by placing a comma at the end of a line
      - ✓ Example
        - – Code

```
print ″First Line″
print ″Second Line″
print ″Both on the″,
print ″same line″
```

        - – Output

```
First Line
Second Line
Both on the same line
```

Outputting to the Console

- Formatting
  - ☞ Built-in Function: format(*item, ' format_specifier'*)
  - ☞ *item* pertains to the item to be formatted
  - ☞ Used where the *item* would normally be placed in the print function argument list
  - ☞ *format_specifier* pertains to the item to be formatted
    - ↳ Syntax: *w.pc*
      - ✓ *w* ≡ Minimum field width
        - – If omitted the minimum width required to display the number is used
        - – Includes sign, decimal point, commas, digits, etc.
        - – Spaces used to pad to desired width
        - – When width specified is too small to display the number, the field is automatically enlarged to the appropriate size
      - ✓ *p* ≡ Precision
        - – Digits after decimal point
      - ✓ *c* ≡ Conversion code
        - – f ≡ Floating Point
        - – e ≡ Scientific Notation, using e in the display
        - – E ≡ Scientific Notation, using E in the display
    - ↳ Comma Separators
      - ✓ Inserting a comma to the left of the precision indicator (.), commas are used when displaying the integer portion of the number if necessary.
    - ↳ Floating Point Numbers as a Percentage
      - ✓ Use % instead of *f* for the conversion code
    - ↳ Formatting Integers
      - ✓ Use *d* for conversion code
      - ✓ Can NOT specify precision

Example

- Code (Python 2)

```
id_num = 1929
score = 0.81
cost = 7419253.291
print format(cost,'20.4f')
print format(cost,'14.4f')
print format(cost,'6,.4f')
print format(id_num,',d')
print format(id_num,' d')
print format(score,'8,.3%')
```

- Output

```
          7419253.291
   7419253.291
7,419,253.291
1,929
 1929
 81.000%
```

- Code (Python 3)

```
id_num = 1929
score = 0.81
cost = 7419253.291
print(format(cost,'20.4f'))
print(format(cost,'14.4f'))
print(format(cost,'6,.4f'))
print(format(id_num,',d'))
print(format(id_num,' d'))
print(format(score,'8,.3%'))
```

- Output

```
          7419253.291
   7419253.291
7,419,253.291
1,929
 1929
 81.000%
```

Named Constants

- *aka*, Constants
- A name representing a permanent constant
- Python has built-in constants for *pi* and *e* as part of the math module
  - ☞ math.pi
  - ☞ math.e
  - ✍ Note to use these, the math module must be imported
    - ↳ import math

- There is no construct to define a new constant
  - Use a variable
    - Use a naming notation to reflect that it is treated as a constant, such as all uppercase
    - Example
      - ✓ MAX_USERS = 10
- Benefits
  - Value does not have to be repeatedly typed
  - Modification of constant only requires changing the value in one place
  - Descriptive name increases readability

Strings & Characters

- No character data type in Python
  - Strings are used for characters
- Python supports ASCII & Unicode
  - A Unicode character is represented by \u*xxxx* where *xxxx* represents four hexadecimal digits
- String Functions
  - ord(c)
    - Returns the ASCII code for the character c
  - chr(code)
    - Returns the character represented by the code
- Escape Sequences for Special Characters
  - Backslash (\) followed by a letter or combination of digits
  - Examples

Name	Escape Sequence	Description
Newline	\n	Line break or end-of-line (EOL)
Page Feed	\f	New page (print) or clear screen (display)
Carriage Return	\r	Moves to the first position on the same line
Backspace	\b	Moves back one character, overwriting the previous character
Tab	\t	Tab stop
Backslash	\\	\
Single Quote	\’	‘
Double Quote	\”	“

- String Concatenation Operator
  - +
  - Augmented assignment operator may be used
    - +=

Mathematical Functions

- Included in math module
- To use, import the module using
  - ☞ import math
- Mathematical Constants
  - ☞ *pi* (math.pi) & *e* (math.e) are included constants in the math module

Function	Description
fabs(x)	Returns the absolute value for x as a float
ceil(x)	Rounds x up to its nearest integer and returns that integer
floor(x)	Rounds x down to its nearest integer and returns that integer
exp(x)	Returns the exponential function of x (e^x)
log(x)	Returns the natural logarithm of x
log(x,base)	Returns the logarithm of x for the specified base
sqrt(x)	Returns the square root of x
sin(x)	Returns the sine of x. The angle, x, must be in radians.
cos(x)	Returns the cosine of x. The angle, x, must be in radians.
acos(x)	Returns the angle in radians for the inverse of cosine.
tan(x)	Returns the tangent of x. The angle, x, must be in radians.
degrees(x)	Converts angle x from radians to degrees
radians(x)	Converts angle x from degrees to radians

Derived from Table 3.2, page 65, of Y. Daniel Liang, *Introduction to Programming Using Python*, Pearson Education, Inc. (Prentice Hall), 2013

Multiline Statements

- A single statement can be written using multiple lines
- Line Continuation Character
  - ☞ \
- Placing \ at the end of a line means that the next line is considered as a continuation of that line
- Advantage
  - ☞ Increases readability
- Example
  - ☞ The following

```
factorial = 6 * 5* 4 * \
3 * 2* 1
```

  - ☞ Is the same as

```
factorial = 6 * 5 * 4 * 3 * 2 * 1
```

References

- Y. Daniel Liang, *Introduction to Programming Using Python*, Pearson Education, Inc. (Prentice Hall), 2013
- Tony Gaddis, *Starting Out With Python*, Second Edition, Pearson Education, Inc. (Addison Wesley), 2012
- Mark J. Guzdial and Barbara Ericson, *Introduction to Computing and Programming in Python – A Multimedia Approach*, Third Edition, Pearson Education, Inc. (Prentice Hall), 2013
- http://www.python.org
- https://wiki.cse.buffalo.edu/services/content/python
- http://sebastianraschka.com/Articles/2014_python_2_3_key_diff.html
- http://python3porting.com/differences.html