CH-2: Introduction to Python Programming

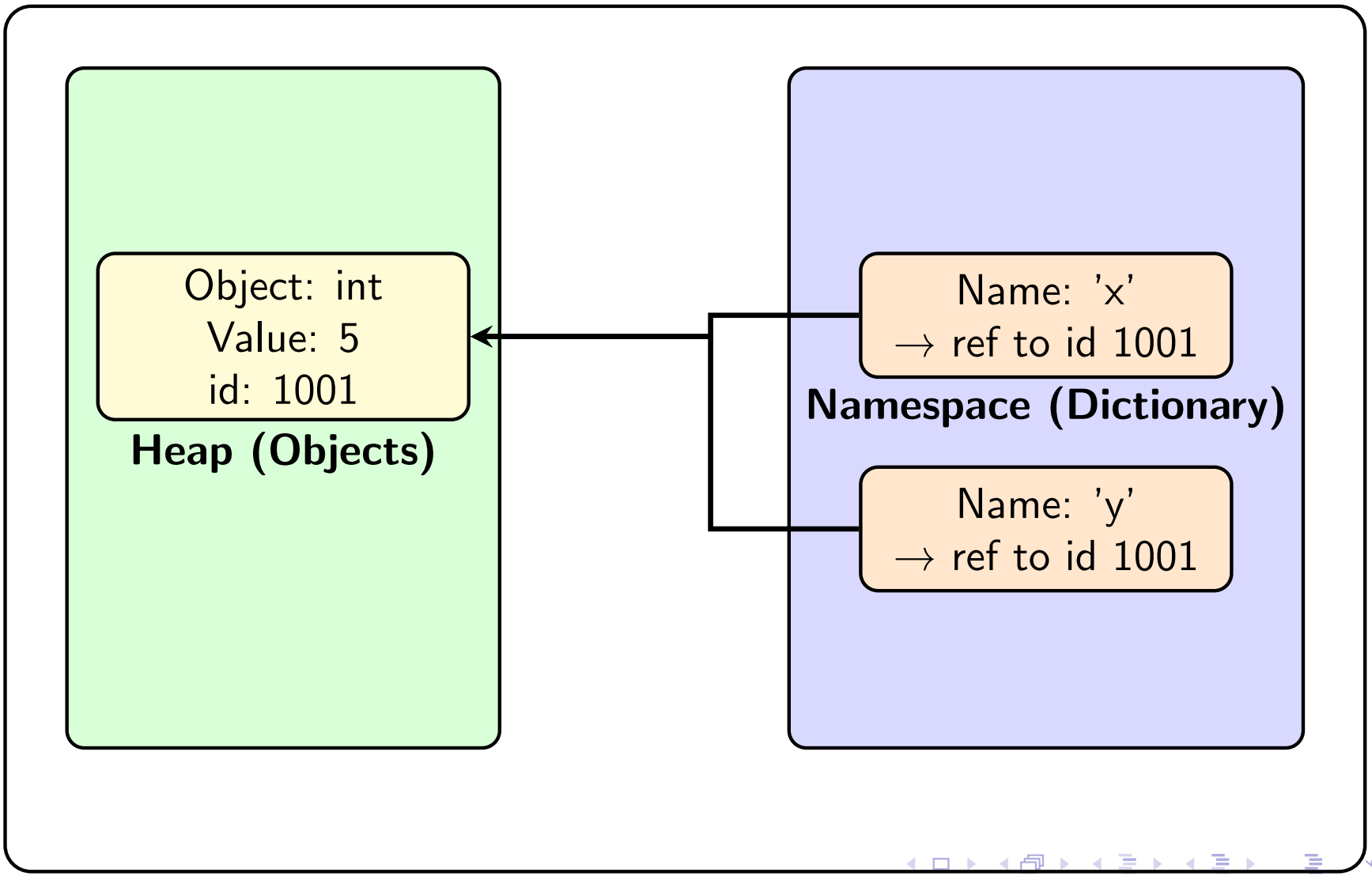
September 2024

Variables and Assignment Statements

- We can use IPython's interactive mode as a calculator for simple expressions.
- Python expressions also may contain variables, which store values for later use in your code.
- You can add variable values and view the result.
- Calculations in Assignment Statements
- Python's Style: The Style Guide for Python Code helps you write code that conforms to Python's coding conventions. The style guide recommends inserting one space on each side of the assignment symbol = and binary operators like + to make programs more readable.
  https://peps.python.org/pep-0008/

- Variable Names: A variable name, such as x, is an identifier. Each identifier may consist of letters, digits and underscores (_) but may not begin with a digit. Python is case sensitive, so number and Number are different identifiers because one begins with a lowercase letter and the other begins with an uppercase letter.
- Type: Each value in Python has a type that indicates the kind of data the value represents.

Memory Model: `x = 5, y = x`



Self Check

1. (**True/False**) The following are valid variable names: 3g, 87 and score_4.
2. (**True/False**) Python treats y and Y as the same identifier.
3. (**IPython Session**) Calculate the sum of 10.8, 12.2 and 0.2, store it in the variable total, then display total's value.

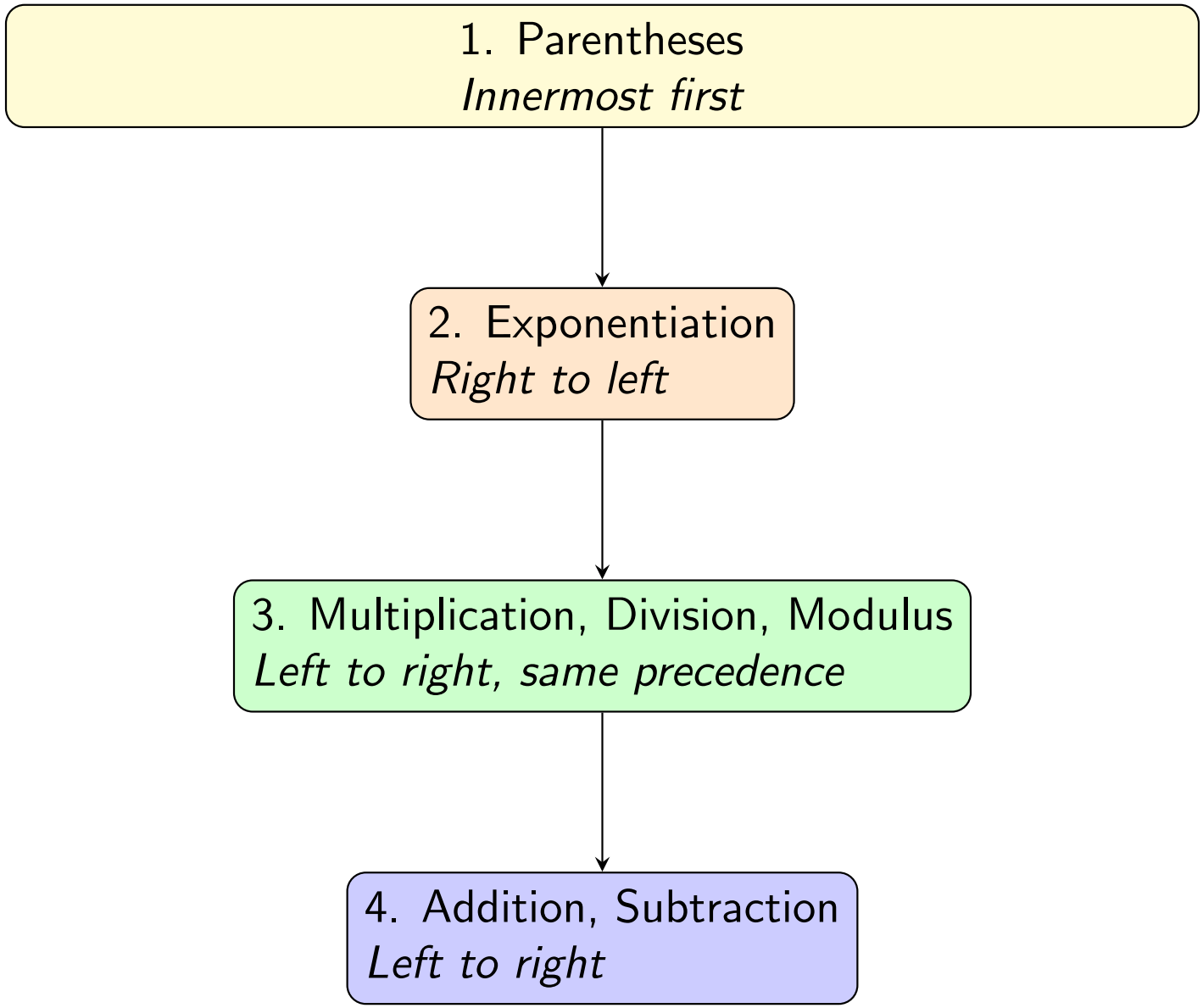
Arithmetic

Python operation	Arithmetic operator	Algebraic expression	Python expression
Addition	+	$f + 7$	`f + 7`
Subtraction	-	$p - c$	`p - c`
Multiplication	*	$b \cdot m$	`b * m`
Exponentiation	**	x^y	`x ** y`
True division	/	x/y or $\frac{x}{y}$ or $x \div y$	`x / y`
Floor division	//	$\lfloor x/y \rfloor$ or $\left\lfloor \frac{x}{y} \right\rfloor$ or $\lfloor x \div y \rfloor$	`x // y`
Remainder (modulo)	%	$r \bmod s$	`r % s`

Operands	Operator	Result	Example
`int`, `int`	+, -, *, //, %	`int`	$5 + 2 = 7$
`int`, `int`	/	float	$5 / 2 = 2.5$
`float`, `float`	+, -, *, /, //, %	`float`	5.0 * 2.0 = 10.0
`int`, `float` / `float`, `int`	any	`float`	$5 + 2.0 = 7.0$

Note: Mixed-type expressions (`int` + `float`) always produce `float`.
True division '/' always yields a `float` even for integers.

- Exceptions and Tracebacks: Dividing by zero with / or // is not allowed and results in an exception.
- Grouping Expressions with Parentheses: Parentheses group Python expressions, as they do in algebraic expressions.
- Operator Precedence Rules:
    1. Expressions in parentheses evaluate first. In expressions with nested parentheses, the expression in the innermost parentheses evaluates first.
    2. Exponentiation operations evaluate next. If an expression contains several exponentiation operations, Python applies them from **right to left**.
    3. Multiplication, division and modulus operations evaluate next. If an expression contains several multiplication, true-division, floor-division and modulus operations, Python applies them from **left to right**. Multiplication, division and modulus are "on the same level of precedence."
    4. Addition and subtraction operations evaluate last. If an expression contains several addition and subtraction operations, Python applies them from **left to right**.


1. Parentheses
Innermost first
2. Exponentiation
Right to left
3. Multiplication, Division, Modulus
Left to right, same precedence
4. Addition, Subtraction
Left to right

Self Check

1. (Multiple Choice) Given that $y = ax^3 + 7$, which of the following is not a correct statement for this equation?
   a. $y = a * x * x * x + 7$
   b. $y = a * x * *3 + 7$
   c. $y = a * (x * x * x) + 7$
   d. $y = a * x * (x * x + 7)$
2. (True/False) In nested parentheses, the expression in the innermost pair evaluates last.
3. (IPython Session) Evaluate the expression 3 * (4 - 5) with and without parentheses. Are the parentheses redundant?
4. (IPython Session) Evaluate the expressions $4 * *3 * *2$, $(4 * *3) * *2$ and $4 * *(3 * *2)$. Are any of the parentheses redundant?

Function print and an Intro to Single- and Double Quoted Strings

- The built-in print function displays its argument(s) as a line of text.
- Unlike when you evaluate expressions in interactive mode, the text that print displays here is not preceded by Out[1].
- Also, print does not display a string's quotes.
- When print completes its task, it positions the screen cursor at the beginning of the next line.
- The print function can receive a comma-separated list of arguments.

- Printing Many Lines of Text with One Statement:
  - When a backslash (\) appears in a string, it's known as the escape character.
  - The backslash and the character immediately following it form an escape sequence.
  - For example, \n represents the newline character escape sequence, which tells print to move the output cursor to the next line.
  - Placing two newline characters back-to-back displays a blank line.

- Other Escape Sequences

Escape sequence	Description
\n	Insert a newline character in a string. When the string is displayed, for each newline, move the screen cursor to the beginning of the next line.
\t	Insert a horizontal tab. When the string is displayed, for each tab, move the screen cursor to the next tab stop.
\\	Insert a backslash character in a string.
\"	Insert a double quote character in a string.
\'	Insert a single quote character in a string.

- You may also split a long string (or a long statement) over several lines by using the \continuation character as the last character on a line to ignore the line break.
- Calculations can be performed in print statements.

Self Check

Self Check

1. (Fill-In) The _____ function instructs the computer to display information on the screen.
2. (Fill-In) Values _____ of the data type contain a sequence of characters.
3. (IPython Session) Write an expression that displays the type of 'word'.
4. (IPython Session) What does the following print statement display?

```
print('int(5.2)', 'truncates 5.2 to', int(5.2))
```

Triple-Quoted Strings

- Triple-quoted strings begin and end with three double quotes (""") or three single quotes (''').
- *The Style Guide for Python Code* recommends three double quotes ("""). Use these to create:
  - multiline strings
  - strings containing single or double quotes and
  - docstrings, which are the recommended way to document the purposes of certain program components.

Self Check

1. (Fill-In) Multiline strings are enclosed either in _____ or in _____.
2. (IPython Session) What displays when you execute the following statement?

```
print("""This is a lengthy
    multiline string containing
a few lines \
of text""")
```

Getting Input from the User

- The built-in input function requests and obtains user input.
- Input displays its string argument—called a prompt—to tell the user what to type and waits for the user to respond.
- Function input then returns (that is, gives back) those characters as a string that the program can use.
- Function input Always Returns a String.
- If you need an integer, convert the string to an integer using the built-in int function.

Getting Input from the User

Self Check

1. (Fill-In) The built-in _____ function converts a floating-point value to an integer value or converts a string representation of an integer to an integer value.
2. (True/False) Built-in function get_input requests and obtains input from the user.
3. (IPython Session) Use float to convert '6.2' (a string) to a floating-point value. Multiply that value by 3.3 and show the result.

Decision Making: The if Statement and Comparison Operators

- A condition is a Boolean expression with the value True or False.
- True and False are keywords—words that Python reserves for its language features.
- Using a keyword as an identifier causes a **SyntaxError**.
- True and False are each capitalized.
- You'll often create conditions using the comparison operators in the table:

Algebraic operator	Python operator	Sample condition	Meaning
$>$	>	x > y	x is greater than y
$<$	<	x < y	x is less than y
$\geq$	>=	x >= y	x is greater than or equal to y
$\leq$	<=	x <= y	x is less than or equal to y
$=$	==	x == y	x is equal to y
$\neq$	!=	x != y	x is not equal to y

- Making Decisions with the if Statement: Introducing Scripts
- Comments
- Docstrings
- Blank Lines
  - Together, blank lines, space characters and tab characters are known as white space.
- Reading Integer Values from the User
- Suite Indentation
- Confusing == and = : read == as "is equal to" and = as "is assigned."
- Chaining Comparisons

Precedence of the Operators We've Presented So Far

Operators	Grouping	Type
()	left to right	parentheses
**	right to left	exponentiation
* / // %	left to right	multiplication, true division, floor division, remainder
+ -	left to right	addition, subtraction
> <= < >=	left to right	less than, less than or equal, greater than, greater than or equal
== !=	left to right	equal, not equal

Self Check

1. (Fill-In) You use _____ to document code and improve its readability.
2. The comparison operators evaluate left to right and all have the same level of precedence.
3. (IPython Session) For any of the operators $!=, >=$ *or* $<=$, show that a syntax error occurs if you reverse the symbols in a condition.
4. (IPython Session) Use all six comparison operators to compare the values 5 and 9. Display the values on one line using print.

Objects and Dynamic Typing

- Values such as 7 (an integer), 4.1 (a floating-point number) and 'dog' are all objects.
- Every object has a type and a value.
- An object's value is the data stored in the object.
- Objects of Python built-in types:
  - int (for integers)
  - float (for floating-point numbers)
  - str (for strings).
- Variables Refer to Objects: Assigning an object to a variable binds (associates) that variable's name to the object.
- Dynamic Typing:
  - Python uses dynamic typing—it determines the type of the object a variable refers to while executing your code.
  - We can show this by rebinding the variable x to different objects and checking their types.

Objects and Dynamic Typing

- Garbage Collection:
  - Python creates objects in memory and removes them from memory as necessary.
  - Garbage collection in Python is an automatic memory management process that reclaims memory by identifying and deleting objects that are no longer in use, freeing up resources and preventing memory leaks.

Self Check

- (Fill-In) Assigning an object to a variable_____ the variable's name to the object.
- (True/False) A variable always references the same object.
- (IPython Session) What is the type of the expression $7.5 * 3$?

Intro to Data Science: Basic Descriptive Statistics

- In data science, you'll often use statistics to describe and summarize your data.
- Several such descriptive statistics:
  - minimum - the smallest value in a collection of values
  - maximum - the largest value in a collection of values
  - range - the range of values from the minimum to the maximum
  - count - the number of values in a collection
  - sum - the total of the values in a collection.
- Determining the Minimum of Three Values
- Determining the Minimum and Maximum with Built-In Functions *min* and *max*
- Determining the Range of a Collection of Values

- Functional-Style Programming: Reduction.
  - You will be introduced various *functional-style programming* capabilities.
  - These enable you to write code that can be more concise, clearer and easier to debug—that is, find and correct errors.
  - The *min* and *max* functions are examples of a functional style programming concept called reduction.
  - They reduce a collection of values to a single value.
  - Other reductions you'll see include the sum, average, variance and standard deviation of a collection of values.

Self Check

- (Fill-In) The range of a collection of values is a measure of _____.
- (IPython Session) For the values 47, 95, 88, 73, 88 and 84 calculate the minimum, maximum and range.

Exercises

Q.1 (Table of Squares and Cubes) Write a script that calculates the squares and cubes of the numbers from 0 to 5. Print the resulting values in table format, as shown below. Use the tab escape sequence to achieve the three-column output.

number	square	cube
0	0	0
1	1	1
2	4	8
3	9	27
4	16	64
5	25	125

Q.2 (Arithmetic, Smallest and Largest) Write a script that inputs three integers from the user. Display the sum, average, product, smallest and largest of the numbers. Note that each of these is a reduction in functional-style programming.

Exercises

Q.3 (Separating the Digits in an Integer) Write a script that inputs a five-digit integer from the user. Separate the number into its individual digits. Print them separated by three spaces each. For example, if the user types in the number 42339, the script should print 4 2 3 3 9.
Assume that the user enters the correct number of digits. Use both the floor division and remainder operations to "pick off" each digit.

Q.4 (What's wrong with this code?) The following code should read an integer into the variable rating:
rating = input('Enter an integer rating between 1 and 10')

Exercises

Q.5 (What does this code do?) Create the variables x = 2 and y = 3, then determine what each of the following statements displays:

a. print('x =', x)
b. print('Value of', x, '+', x, 'is', (x + x))
c. print('x =')
d. print((x + y), '=', (y + x)).

Q.6 (Sort in Ascending Order) Write a script that inputs three different floating-point numbers from the user. Display the numbers in increasing order. Recall that an if statement's suite can contain more than one statement. Prove that your script works by running it on all six possible orderings of the numbers. Does your script work with duplicate numbers? [This is challenging. In later chapters you'll do this more conveniently and with many more numbers.]

Exercises

Q.6 (7% Investment Return) Some investment advisors say that it's reasonable to expect a 7% return over the long term in the stock market. Assuming that you begin with $1000 and leave your money invested, calculate and display how much money you'll have after 10, 20 and 30 years. Use the following formula for determining these amounts:

$$a = p(1 + r)^n$$

where
p is the original amount invested (i.e., the principal of $1000),
r is the annual rate of return (7%), n is the number of years (10, 20 or 30) and
a is the amount on deposit at the end of the nth year.

Exercises

Q.7 (Integer Value of a Character) Here's a peek ahead. In this chapter, you learned about strings. Each of a string's characters has an integer representation. The set of characters a computer uses together with the characters' integer representations is called that computer's character set. You can indicate a character value in a program by enclosing that character in quotes, as in 'A'. To determine a character's integer value, call the built-in function ord:
Display the integer equivalents of B C D b c d 0 1 2 $ + and the space character.