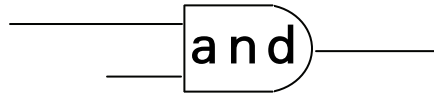


Mathematics for the Digital Age



Programming in Python

>>> Second Edition:
with Python 3

Maria Litvin

Phillips Academy, Andover, Massachusetts

Gary Litvin

Skylight Software, Inc.

Skylight Publishing
Andover, Massachusetts

Skylight Publishing
9 Bartlet Street, Suite 70
Andover, MA 01810

web: <http://www.skylit.com>
e-mail: sales@skylit.com
support@skylit.com

**Copyright © 2010 by Maria Litvin, Gary Litvin, and
Skylight Publishing**

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the authors and Skylight Publishing.

Library of Congress Control Number: 2009913596

ISBN 978-0-9824775-8-8 (soft cover)

ISBN 978-0-9824775-4-0 (hard cover)

The names of commercially available software and products mentioned in this book are used for identification purposes only and may be trademarks or registered trademarks owned by corporations and other commercial entities. Skylight Publishing and the authors have no affiliation with and disclaim any sponsorship or endorsement by any of these products' manufacturers or trademarks' owners.

1 2 3 4 5 6 7 8 9 10 15 14 13 12 11 10

Printed in the United States of America

Appendix A. Getting Started with Python

This appendix is on the Internet at:

www.skylit.com/python

Appendix B. Selected Functions: Built-In, Math, and Random

| | |
|----------------------------------|--|
| <code>help(obj)</code> | displays help for a function or module, as in <pre>>>> import math >>> help(math)</pre> |
| <code>input(s)</code> | displays <code>s</code> as a prompt, then reads a line of text, typed in by the user, and returns it as a string |
| <code>abs(x)</code> | returns the absolute value of <code>x</code> |
| <code>max(a, b)</code> | returns the largest of <code>a</code> , <code>b</code> |
| <code>min(a, b)</code> | returns the smallest of <code>a</code> , <code>b</code> |
| <code>int(s)</code> | converts a string or a float into an integer |
| <code>float(s)</code> | converts a string or an int into a float |
| <code>str(n)</code> | converts <code>n</code> into a string |
| <code>bin(n)</code> | returns a string that represents <code>n</code> in binary |
| <code>hex(n)</code> | returns a string that represents <code>n</code> in hex |
| <code>oct(n)</code> | returns a string that represents <code>n</code> in octal |
| <code>len(s)</code> | returns the length of a string, list, or tuple |
| <code>sum(lst)</code> | returns the sum of the numbers from a list or tuple |
| <code>max(lst)</code> | returns the largest element of a list or tuple |
| <code>min(lst)</code> | returns the smallest element of a list or tuple |
| <code>list(s)</code> | converts a string or tuple into a list |
| <code>tuple(s)</code> | converts a string or list into a tuple |
| <code>range(n)</code> | generates <code>0, . . . , n-1</code> , as in: <pre>for i in range(n):</pre> |
| <code>open(pathname)</code> | opens a file |
| <code>open(pathname, 'w')</code> | creates a file and opens it for writing |

| <code>from math import *</code> | | <code>from random import *</code> | |
|---------------------------------|------------|-----------------------------------|---------------------------|
| <code>pi</code> | 3.14159... | <code>x = random()</code> | $0 \leq x < 1$ |
| <code>e</code> | 2.71828... | <code>r = randint(a, b)</code> | $a \leq r \leq b$ |
| <code>sqrt(x)</code> | \sqrt{x} | <code>a = choice(s)</code> | <code>s[r]</code> |
| <code>pow(x, y)</code> | x^y | <code>shuffle(lst)</code> | shuffles <code>lst</code> |
| <code>exp(x)</code> | e^x | | |

Appendix C. String Operations and Methods

Contents type

| | |
|--|---|
| The following methods return <code>True</code> if all the characters in <code>s</code> belong to the corresponding category; otherwise return <code>False</code> : | |
| <code>s.isalpha()</code> | Checks whether all the characters in <code>s</code> are letters |
| <code>s.isdigit()</code> | Checks whether all the characters in <code>s</code> are digits |
| <code>s.isalnum()</code> | Checks whether each character in <code>s</code> is either a letter or a digit |
| <code>s.isupper()</code> | Checks whether all the letters in <code>s</code> are upper case |
| <code>s.islower()</code> | Checks whether all the letters in <code>s</code> are lower case |
| <code>s.isspace()</code> | Checks whether all characters in <code>s</code> are “white space” (spaces, tabs, newline, etc.) |

Examples:

```
>>> 'ab7'.isalpha()
False
>>> 'ab7'.isdigit()
False
>>> 'ab7'.isalnum()
True
>>> 'ab7'.isupper()
False
>>> 'ab7'.islower()
True
>>> ' * '.isspace()
False
>>> '\n'.isspace()
True
```

Length and substrings

| | |
|--------------------------|---|
| <code>len(s)</code> | Returns the number of characters in <code>s</code> |
| <code>ch = s[i]</code> | Sets <code>ch</code> to the i -th character in <code>s</code> |
| <code>s2 = s[i:j]</code> | Sets <code>s2</code> to the substring of <code>s</code> from i to $j-1$ |

Examples:

```
>>> len('abcd')
4
>>> 'abcd'[1]
'b'
```

```
>>> 'abcd'[1:3]
'bc'
>>> 'abcd'[:3]
'abc'
```

Search

| | |
|--|--|
| The following methods return an int: | |
| <code>s.find(sub)</code> | Returns the index of the first occurrence of <code>sub</code> in <code>s</code> ; if <code>sub</code> is not found, returns <code>-1</code> |
| <code>s.rfind(sub)</code> | Returns the index of the last occurrence of <code>sub</code> in <code>s</code> |
| <code>s.count(sub)</code> | Returns the number of times <code>sub</code> occurs in <code>s</code> |
| <code>s.find(sub, start, end)</code> <code>s.rfind(sub, start, end)</code> <code>s.count(sub, start, end)</code> | The versions with optional arguments <code>start</code> , <code>end</code> , look for <code>sub</code> only within the substring of <code>s</code> between <code>start</code> and <code>end-1</code> |

Examples:

```
>>> 'never'.find('e')
1
>>> 'never'.find('x')
-1
>>> 'never'.rfind('e')
3
>>> 'never'.count('e')
2
```

```
>>> 'never'.find('ver')
2
>>> 'never'.find('e',2,4)
3
>>> 'never'.rfind('e',1,3)
1
>>> 'never'.find('ver',2,4)
-1
```

Case conversions

| | |
|--|---|
| The following methods return a new string: | |
| <code>s.upper()</code> | All the letters are converted to upper case |
| <code>s.lower()</code> | All the letters are converted to lower case |
| <code>s.capitalize()</code> | The first letter is converted to upper case |

Examples:

```
>>> 'ab7'.upper()
'AB7'
>>> 'Ab7'.lower()
'ab7'
>>> 'ab7'.capitalize()
'Ab7'
>>> '7ab'.capitalize()
'7ab'
```

Editing

| | |
|--|---|
| The following methods return a new string: | |
| <code>s.replace(old, new)</code> | Replaces every occurrence of <code>old</code> in <code>s</code> with <code>new</code> |
| <code>s.strip()</code> | Removes white space at the beginning and at the end of <code>s</code> |

Examples:

```
>>> '1*2*3'.replace('*', '--')
'1--2--3'
>>> ' ab \n'.strip()
'ab'
```

Parsing

| | |
|--------------------------------------|---|
| The following methods return a list: | |
| <code>s.split(delim)</code> | returns a list of substrings separated by occurrences of <code>delim</code> in <code>s</code> |
| <code>s.splitlines()</code> | returns a list of lines in <code>s</code> — the same as <code>s.split('\n')</code> . |

Examples:

```
>>> 'Line1\nLine 2'.splitlines() | >>> '1, 2, 3'.split(',')
['Line1', 'Line 2']           | ['1', '2', '3']
```

Formatting

| | |
|--|---|
| The following methods return a new string: | |
| <code>s.format(value, ...)</code> | Formats <code>value</code> (or several values according to the format fields in <code>s</code>) |
| <code>s.ljust(w, fill)</code> | Left-justifies <code>s</code> within a string of length <code>w</code> and pads it on the right with the <code>fill</code> character (<code>fill</code> is optional: if not given, <code>ljust</code> , <code>rjust</code> and <code>center</code> , use the space character as the default) |
| <code>s.rjust(w, fill)</code> | Right-justifies <code>s</code> and pads it on left with <code>fill</code> |
| <code>s.center(w, fill)</code> | Positions <code>s</code> in the middle of a string of length <code>w</code> and pads it on both sides with <code>fill</code> |
| <code>s.zfill(w)</code> | Right-justifies the string and pads it with 0s on the left — the same as <code>s.rjust(w, '0')</code> |

Examples:

```
>>> '{0:>4s}{1:7.2f}'.format('$', 2.5) | >>> 'ab'.rjust(6)
' $ 2.50'                               | ' ab'
>>> 'ab'.ljust(6, '*')                   | >>> 'ab'.center(6)
'ab*****'                               | ' ab '
                                           | >>> '12'.zfill(4)
                                           | '0012'
```

Appendix D. List, Set, and Dictionary Operations and Methods

Lists

| Method/Operation | Description |
|-------------------------------|---|
| <code>len(lst)</code> | Returns the number of elements in <code>lst</code> |
| <code>x = lst[i]</code> | Sets <code>x</code> to the i -th element of <code>lst</code> |
| <code>lst[i] = x</code> | Sets the i -th element of <code>lst</code> to <code>x</code> |
| <code>del lst[i]</code> | Deletes the i -th element and decrements the indices of the subsequent elements by one |
| <code>del lst[i:j]</code> | Deletes the slice from i to j and adjusts the indices of the subsequent elements |
| <code>lst2 = lst[i:j]</code> | Creates a copy of the specified slice from <code>lst</code> and assigns it to <code>lst2</code> |
| <code>lst2 = lst[:]</code> | Creates a copy of <code>lst</code> and assigns it to <code>lst2</code> |
| <code>lst.insert(i, x)</code> | Inserts <code>x</code> at index i , shifting the subsequent elements to the right by 1 |
| <code>lst.append(x)</code> | Appends <code>x</code> at the end of <code>lst</code> |
| <code>lst.pop(i)</code> | Returns the i -th element and removes it from <code>lst</code> |
| <code>lst.pop()</code> | Returns the last element and removes it from <code>lst</code> |
| <code>lst.remove(x)</code> | Removes the first occurrence of <code>x</code> from <code>lst</code> ; raises an exception if none found |
| <code>lst.index(x)</code> | Returns the index of the first occurrence of <code>x</code> in <code>lst</code> ; raises an exception if none found |
| <code>lst.count(x)</code> | Returns the number of times <code>x</code> occurs in <code>lst</code> |
| <code>lst.reverse()</code> | Reverses the order of elements in <code>lst</code> ; returns <code>None</code> |
| <code>lst.sort()</code> | Arranges the elements of <code>lst</code> in ascending order; returns <code>None</code> |

Examples:

```
>>> lst=['A', 'C']
>>> lst
['A', 'C']
>>> lst.insert(1, 'B')
>>> lst
['A', 'B', 'C']
>>> lst.append('A')
>>> lst
['A', 'B', 'C', 'A']
>>> lst.insert(2, 'A')
>>> lst
['A', 'B', 'A', 'C', 'A']
>>> lst.count('A')
3
>>> del lst[2]
>>> lst
['A', 'B', 'C', 'A']
>>> lst.reverse()
>>> lst
['A', 'C', 'B', 'A']
>>> lst.index('A')
0
>>> lst.remove('A')
>>> lst
['C', 'B', 'A']
>>> lst.sort()
>>> lst
['A', 'B', 'C']
>>> lst.pop(1)
'B'
>>> lst
['A', 'C']
>>> lst.pop()
'C'
>>> lst
['A']
```

Sets

| Method/Operation | Description |
|------------------------------|---|
| <code>len(s)</code> | Returns the number of elements in <code>s</code> |
| <code>s.copy()</code> | Returns a copy of <code>s</code> |
| <code>s.add(x)</code> | Adds <code>x</code> to <code>s</code> |
| <code>s.remove(x)</code> | Removes <code>x</code> from <code>s</code> ; raises an exception if <code>x</code> is not in <code>s</code> |
| <code>s.discard(x)</code> | Removes <code>x</code> from <code>s</code> ; has no effect if <code>x</code> is not in <code>s</code> |
| <code>s.pop()</code> | Removes and returns an arbitrary element from <code>s</code> |
| <code>s1.issubset(s2)</code> | Returns <code>True</code> if <code>s1</code> is a subset of <code>s2</code> |
| <code>s.update(s2)</code> | Adds all the elements from a list, tuple, or set <code>s2</code> to <code>s</code> |

Examples:

```
>>> s = {'A', 'C'}
>>> s
{'A', 'C'}
>>> s.add('B')
>>> s
{'A', 'C', 'B'}
>>> s.remove('A')
>>> s
{'C', 'B'}
>>> s.discard('X')
>>> s
{'C', 'B'}
>>> s.pop()
'C'
>>> s
{'B'}
>>> s2 = set('ABCD')
>>> s2
{'A', 'C', 'B', 'D'}
```

```
>>> s.issubset(s2)
True
>>> s.add('X')
>>> s
{'X', 'B'}
>>> s.issubset(s2)
False
>>> s.update(s2)
>>> s
{'A', 'C', 'B', 'D', 'X'}
```

Dictionaries

| Method/Operation | Description |
|-----------------------------|---|
| <code>len(d)</code> | Returns the number of key-value pairs in <code>d</code> |
| <code>x = d[k]</code> | Sets <code>x</code> to the value associated with the key <code>k</code> in <code>d</code> |
| <code>d[k] = x</code> | If the key <code>k</code> is in <code>d</code> , changes the value associated with <code>k</code> to <code>x</code> ; if <code>k</code> is not in <code>d</code> , adds the <code>k:x</code> pair to <code>d</code> |
| <code>del d[k]</code> | Deletes the key <code>k</code> and the associated value from <code>d</code> |
| <code>d2 = d.copy()</code> | Creates a copy of <code>d</code> and assigns it to <code>d2</code> |
| <code>k in d</code> | Returns <code>True</code> if the key <code>k</code> is in <code>d</code> ; otherwise returns <code>False</code> |
| <code>d.keys()</code> | Returns a <code>dict_keys</code> object that contains all the keys in <code>d</code> |
| <code>d.items()</code> | Returns a set (a <code>dict_items</code> object) of all (key, value) pairs in <code>d</code> |
| <code>d.update(d2)</code> | Adds all the key-value pairs from <code>d2</code> to <code>d</code> |
| <code>d.get(k)</code> | The same as <code>d[k]</code> |
| <code>d.get(k, dflt)</code> | If <code>k</code> is in <code>d</code> , returns <code>d[k]</code> ; otherwise returns <code>dflt</code> |

Examples:

```
>>> d = {'K1':'V1', 'K2':'V2'}
>>> d.keys()
dict_keys(['K2', 'K1'])
>>> d.items()
dict_items([('K2', 'V2'), ('K1', 'V1')])
>>> 'K2' in d
True
>>> del d['K2']
>>> d
{'K1': 'V1'}
>>> 'K2' in d
False
>>> d['K2'] = 'V2'
>>> d
{'K2': 'V2', 'K1': 'V1'}
```

```
>>> d.get('K2')
'V2'
>>> d.get('X', 'oops')
'oops'
```