Appendix A: Integer Operators 8

This documentation was generated from the Python documentation available by typing *help*(*int*) in the Python shell. In this documentation the variables *x*, *y*, and *z* refer to integers (Table 8.1).

Table 8.1 Integer operators

Operator	Returns	Comments
$x + y$	int	Returns the sum of x and y
$x - y$	int	Returns the difference of x and y
x*y	int	Returns the product of x and y
x/y	float	Returns the quotient of x divided by y
x//y	int	Returns the integer quotient of x divided by y
x % y	int	Returns x modulo y. This is the remainder of dividing x by y
$-x$	int	Returns the negation of x
x&y	int	Returns the bit-wise *and* of x and y
$x \mid y$	int	Returns the bit-wise *or* of x and y
x ˆ y	int	Returns the bit-wise *exclusive or* of x and y
$x \ll y$	int	Returns a bit-wise shift left of x by y bits. Shifting left by 1 bit multiplies x by 2
$x \gg y$	int	Returns a bit-wise right shift of x by y bits
˜ x	int	Returns an integer where each bit in the x has been inverted. $x + x = -1$ for all x
abs(x)	int	Returns the absolute value of x
divmod(x, y)	(q,r)	Returns the quotient q and the remainder r as a tuple
float(x)	float	Returns the float representation of x
hex(x)	str	Returns a hexadecimal representation of x as a string
int(x)	int	Returns x

(continued)

Table 8.1 (continued)

Operator	Returns	Comments
oct(x)	str	Return an octal representation of x as a string
pow(x, y[, z])	int	Returns x to the y power modulo z. If z is not specified then it returns x to the y power
repr(x)	str	Returns a string representation of x
str(x)	str	Returns a string representation of x

Appendix B: Float Operators 9

This documentation was generated from the Python documentation available by typing *help*(*float*) in the Python shell. In this documentation at least one of the variables *x* and *y* refer to floats (Table 9.1).

Table 9.1 Float operators

Operator	Returns	Comments
x + y	float	Returns the sum of x and y
x − y	float	Returns the difference of x and y
x*y	float	Returns the product of x and y
x/y	float	Returns the quotient of x divided by y
x//y	float	Returns the quotient of integer division of x divided by y. However, the result is still a float
x % y	float	Returns x modulo y. This is the remainder of dividing x by y
abs(x)	int	Returns the absolute value of x
divmod(x, y)	(q,r)	Returns the quotient q and the remainder r as a tuple. Both q and r are floats, but integer division is performed. The value r is the whole and fractional part of any remainder. The value q is a whole number
float(x)	float	Returns the float representation of x
int(x)	int	Returns the floor of x as an integer
pow(x, y)	float	Returns x to the y power
repr(x)	str	Returns a string representation of x
str(x)	str	Returns a string representation of x

© Springer-Verlag London 2014
K.D. Lee, *Python Programming Fundamentals*,
Undergraduate Topics in Computer Science, DOI 10.1007/978-1-4471-6642-9_9

Appendix C: String Operators and Methods 10

This documentation was generated from the Python documentation available by typing *help*(*str*) in the Python shell. In the documentation found here the variables *s* and *t* are references to strings (Table 10.1).

Table 10.1 String operators and methods

Operator	Returns	Comments
s+t	str	Return a new string which is the concatenation of s and t
s in t	bool	Returns True if s is a substring of t and False otherwise
s==t	bool	Returns True if s and t refer to strings with the same sequence of characters
s>=t	bool	Returns True if s is lexicographically greater than or equal to t
s<=t	bool	Returns True if s is lexicographically less than or equal to t
s>t	bool	Returns True if s is lexicographically greater than t
s<t	bool	Returns True if s is lexicographically less than t
s != t	bool	Returns True if s is lexicographically not equal to t
s[i]	str	Returns the character at index i in the string. If i is negative then it returns the character at index len(s)−i
s[[i]:[j]]	str	Returns the slice of characters starting at index i and extending to index j−1 in the string. If i is omitted then the slice begins at index 0. If j is omitted then the slice extends to the end of the list. If i is negative then it returns the slice starting at index len(s)+i (and likewise for the slice ending at j)
s ∗ i	str	Returns a new string with s repeated i times
i ∗ s	str	Returns a new string with s repeated i times
chr(i)	str	Return the ASCII character equivalent of the integer i
float(s)	float	Returns the float contained in the string s
int(s)	int	Returns the integer contained in the string s
len(s)	int	Returns the number of characters in s
ord(s)	int	Returns the ASCII decimal equivalent of the single character string s

(continued)

K.D. Lee, *Python Programming Fundamentals*,
Undergraduate Topics in Computer Science, DOI 10.1007/978-1-4471-6642-9_10

Table 10.1 (continued)

Method	Returns	Comments
repr(s)		Returns a string representation of s. This adds an extra pair of quotes to s
str(s)	str	Returns a string representation of s. In this case you get just the string s
s.capitalize()	str	Returns a copy of the string s with the first character upper case
s.center(width[, fillchar])	str	Returns s centered in a string of length width. Padding is done using the specified fill character (default is a space)
s.count(sub[, start[, end]])	int	Returns the number of non-overlapping occurrences of substring sub in string s[start:end]. Optional arguments start and end are interpreted as in slice notation
s.encode([encoding[, errors]])	bytes	Encodes s using the codec registered for encoding. encoding defaults to the default encoding. errors may be given to set a different error handling scheme. Default is 'strict' meaning that encoding errors raise a UnicodeEncodeError. Other possible values are 'ignore', 'replace' and 'xmlcharrefreplace' as well as any other name registered with codecs.register_error that can handle UnicodeEncodeErrors
s.endswith(suffix[, start[, end]])	bool	Returns True if s ends with the specified suffix, False otherwise. With optional start, test s beginning at that position. With optional end, stop comparing s at that position. suffix can also be a tuple of strings to try
s.expandtabs([tabsize])	str	Returns a copy of s where all tab characters are expanded using spaces. If tabsize is not given, a tab size of 8 characters is assumed
s.find(sub[, start[, end]])	int	Returns the lowest index in s where substring sub is found, such that sub is contained within s[start:end]. Optional arguments start and end are interpreted as in slice notation. Return −1 on failure
s.format(*args, **kwargs)	str	
s.index(sub[, start[, end]])	int	Like s.find() but raise ValueError when the substring is not found
s.isalnum()	bool	Returns True if all characters in s are alphanumeric and there is at least one character in s, False otherwise
s.isalpha()	bool	Returns True if all characters in s are alphabetic and there is at least one character in s, False otherwise
s.isdecimal()	bool	Returns True if there are only decimal characters in s, False otherwise

(continued)

Table 10.1 (continued)

Method	Returns	Comments
s.isdigit()	bool	Returns True if all characters in s are digits and there is at least one character in s, False otherwise
s.isidentifier()	bool	Returns True if s is a valid identifier according to the language definition
s.islower()	bool	Returns True if all cased characters in s are lowercase and there is at least one cased character in s, False otherwise
s.isnumeric()	bool	Returns True if there are only numeric characters in s, False otherwise
s.isprintable()	bool	Returns True if all characters in s are considered printable in repr() or s is empty, False otherwise
s.isspace()	bool	Returns True if all characters in s are whitespace and there is at least one character in s, False otherwise
s.istitle()	bool	Returns True if s is a titlecased string and there is at least one character in s, i.e. upper- and titlecase characters may only follow uncased characters and lowercase characters only cased ones. Return False otherwise
s.isupper()	bool	Returns True if all cased characters in s are uppercase and there is at least one cased character in s, False otherwise
s.join(sequence)	str	Returns a string which is the concatenation of the strings in the sequence. The separator between elements is s
s.ljust(width[, fillchar])	str	Returns s left-justified in a Unicode string of length width. Padding is done using the specified fill character (default is a space)
s.lower()	str	Returns a copy of the string s converted to lowercase
s.lstrip([chars])	str	Returns a copy of the string s with leading whitespace removed. If chars is given and not None, remove characters in chars instead
s.partition(sep)	(h,sep,t)	Searches for the separator sep in s, and returns the part before it, the separator itself, and the part after it. If the separator is not found, returns s and two empty strings
s.replace (old, new[, count])	str	Returns a copy of s with all occurrences of substring old replaced by new. If the optional argument count is given, only the first count occurrences are replaced
s.rfind(sub[, start[, end]])	int	Returns the highest index in s where substring sub is found, such that sub is contained within s[start:end]. Optional arguments start and end are interpreted as in slice notation. Returns −1 on failure
s.rindex(sub[, start[, end]])	int	Like s.rfind() but raise ValueError when the substring is not found

(continued)

Table 10.1 (continued)

Method	Returns	Comments
s.rjust(width[, fillchar])	str	Returns s right-justified in a string of length width. Padding is done using the specified fill character (default is a space)
s.rpartition(sep)	(t,sep,h)	Searches for the separator sep in s, starting at the end of s, and returns the part before it, the separator itself, and the part after it. If the separator is not found, returns two empty strings and s
s.rsplit([sep[, maxsplit]])	string list	Returns a list of the words in s, using sep as the delimiter string, starting at the end of the string and working to the front. If maxsplit is given, at most maxsplit splits are done. If sep is not specified, any whitespace string is a separator
s.rstrip([chars])	str	Returns a copy of the string s with trailing whitespace removed. If chars is given and not None, removes characters in chars instead
s.split([sep[, maxsplit]])	string list	Returns a list of the words in s, using sep as the delimiter string. If maxsplit is given, at most maxsplit splits are done. If sep is not specified or is None, any whitespace string is a separator and empty strings are removed from the result
s.splitlines([keepends])	string list	Returns a list of the lines in s, breaking at line boundaries. Line breaks are not included in the resulting list unless keepends is given and true
s.startswith (prefix[, start[, end]])	bool	Returns True if s starts with the specified prefix, False otherwise. With optional start, test s beginning at that position. With optional end, stop comparing s at that position. Prefix can also be a tuple of strings to try
s.strip([chars])	str	Returns a copy of the string s with leading and trailing whitespace removed. If chars is given and not None, removes characters in chars instead.
s.swapcase()	str	Returns a copy of s with uppercase characters converted to lowercase and vice versa
s.title()	str	Returns a titlecased version of s, i.e. words start with title case characters, all remaining cased characters have lower case
s.translate(table)	str	Returns a copy of the string s, where all characters have been mapped through the given translation table, which must be a mapping of Unicode ordinals to Unicode ordinals, strings, or None. Unmapped characters are left untouched. Characters mapped to None are deleted
s.upper()	str	Returns a copy of s converted to uppercase
s.zfill(width)	str	Pad a numeric string s with zeros on the left, to fill a field of the specified width. The string s is never truncated

Appendix D: List Operators and Methods

11

This documentation was generated from the Python documentation available by typing *help*(*list*) in the Python shell. In the documentation found here the variables *x* and *y* are references to lists (Table 11.1).

Table 11.1 List operators and methods

Method	Returns	Comments
list()	list	Returns a new empty list. You can also use `[]` to initialize a new empty list
list(sequence)	list	Returns new list initialized from sequence's items
[item [,item]+]	list	Writing a number of comma-separated items in square brackets constructs a new list of those items
x+y	list	Returns a new list containing the concatenation of the items in x and y
e in x	bool	Returns True if the item e is in x and False otherwise
del x[i]		Deletes the item at index i in x. This is not an expression and does not return a value
x==y	bool	Returns True if x and y contain the same number of items and each of those corresponding items are pairwise equal
x>=y	bool	Returns True if x is greater than or equal to y according to a lexicographical ordering of the elements in x and y. If x and y have different lengths their items are == up to the shortest length, then this returns True if x is longer than y
x<=y	bool	Returns True if x is lexicographically before y or equal to y and False otherwise
x>y	bool	Returns True if x is lexicographically after y and False otherwise
x<y	bool	Returns True if x is lexicographically before y and False otherwise
x != y	bool	Returns True if x and y are of different length or if some item of x is not == to some item of y. Otherwise it returns False

(continued)

K.D. Lee, *Python Programming Fundamentals*,
Undergraduate Topics in Computer Science, DOI 10.1007/978-1-4471-6642-9_11

Table 11.1 (continued)

Method	Returns	Comments
x[i]	item	Returns the item at index i of x
x[[i]:[j]]	list	Returns the slice of items starting at index i and extending to index j−1 in the string. If i is omitted then the slice begins at index 0. If j is omitted then the slice extends to the end of the list. If i is negative then it returns the slice starting at index len(x)+i (and likewise for the slice ending at j)
x[i]=e		Assigns the position at index i the value of e in x. The list x must already have an item at index i before this assignment occurs. In other words, assigning an item to a list in this way will not extend the length of the list to accommodate it
x+=y		This mutates the list x to append the items in y
x*=i		This mutates the list x to be i copies of the original x
iter(x)	iterator	Returns an iterator over x
len(x)	int	Returns the number of items in x
x*i	list	Returns a new list with the items of x repeated i times
i*x	list	Returns a new list with the items of x repeated i times
repr(x)	str	Returns a string representation of x
x.append(e)	None	This mutates the value of x to add e as its last element. The function returns None, but the return value is irrelevant since it mutates x
x.count(e)	int	Returns the number of occurrences of e in x by using == equality
x.extend(iter)	None	Mutates x by appending elements from the iterable, iter
x.index(e,[i,[j]])	int	Returns the first index of an element that == e between the start index, i, and the stop index, j−1. It raises ValueError if the value is not present in the specified sequence. If j is omitted then it searches to the end of the list. If i is omitted then it searches from the beginning of the list
x.insert(i, e)	None	Insert e before index i in x, mutating x
x.pop([index])	item	Remove and return the item at index. If index is omitted then the item at len(x)−1 is removed. The pop method returns the item and mutates x. It raises IndexError if list is empty or index is out of range
x.remove(e)	None	remove first occurrence of e in x, mutating x. It raises ValueError if the value is not present
x.reverse()	None	Reverses all the items in x, mutating x
x.sort()	None	Sorts all the items of x according to their natural ordering as determined by the item's __cmp__ method, mutating x. Two keyword parameters are possible: key and reverse. If reverse = True is specified, then the result of sorting will have the list in reverse of the natural ordering. If key = f is specified then f must be a function that takes an item of x and returns the value of that item that should be used as the key when sorting

Appendix E: Dictionary Operators and Methods 12

This documentation was generated from the Python documentation available by typing *help*(*dict*) in the Python shell. In the documentation found here the variable *D* is a reference to a dictionary. A few methods were omitted here for brevity (Table 12.1).

Table 12.1 Dictionary operators and methods

Method	Returns	Comments
dict()	dict	New empty dictionary
dict(mapping)	dict	New dictionary initialized from a mapping object's (key, value) pairs

dict(seq)	dict	New dictionary initialized as if via
		D = { }
		for k, v in seq
		D[k] = v

dict(**kwargs)	dict	New dictionary initialized with the name = value pairs in the keyword arg list. For example: dict(one = 1, two = 2)
k in D	bool	True if D has key k, else False
del D[k]		Deletes key k from dictionary D
D1== 2	bool	Returns True if dictionaries D1 and D2 have same keys mapped to same values
D[k]	value type	Returns value k maps to in D. If k is not mapped, it raises a KeyError exception
iter(D)	iterator	Returns an iterator over D
len(D)	int	Returns the number of keys in D
D1!=D2	bool	Returns True if D1 and D2 have any different keys or keys map to different values
repr(D)	str	Returns a string representation of D
D[k]=e	–	Stores the key,value pair k,e in D

(continued)

© Springer-Verlag London 2014
K.D. Lee, *Python Programming Fundamentals*,
Undergraduate Topics in Computer Science, DOI 10.1007/978-1-4471-6642-9_12

Table 12.1 (continued)

Method	Returns	Comments
D.clear()	None	Remove all items from D
D.copy()	dict	A shallow copy of D
D.get(k[,e])	value type	D[k] if k in D, else e. e defaults to None
D.items()	items	A set-like object providing a view on D's items
D.keys()	keys	A set-like object providing a view on D's keys
D.pop(k[,e])	v	Remove specified key and return the corresponding value. If key is not found, e is returned if given, otherwise KeyError is raised
D.popitem()	(k, v)	Remove and return some (key, value) pair as a 2-tuple; but raise KeyError if D is empty
D.setdefault(k[,e])	D.get(k,e)	Returns D.get(k,e) and also sets d[k] = e if k not in D
D.update(E, **F)	None	Update D from dict/iterable E and F If E has a .keys() method, does: for k in E: D[k] = E[k] If E lacks .keys() method, does: for (k, v) in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]
D.values()	values	An object providing a view on D's values