Duke CS101 Python Reference Sheet

Mathematical Operators

Symbol	Meaning	Example
+	addition	$4 + 5 = 9$
−	subtraction	$9 - 5 = 4$
*	multiplication	3*5 = 15 and 4.0*5 = 20.0
/	division	$6/4 = 1.5$
//	floor division	$6//4 = 1$
%	mod/remainder	$5\%3 = 2$
**	exponentiation	2**3 = 8, 3**2 = 9

String Operators

Symbol	Meaning	Example
+	concatenation	`"ab" + "cd" == "abcd"`
*	repeat	`"xo"*3 == "xoxoxo"`

Comparison Operators

Symbol	Meaning	Example
==	is equal to	3 == 3 is True
!=	is not equal to	3! = 3 is False
>=	is greater than or equal to	4 >= 3 is True
<=	is less than or equal to	4 <= 3 is False
>	is strictly greater than	$4 > 3$ is True
<	is strictly less than	$3 < 3$ is False

Boolean operators

Assume x = 5

not	flips/negates the value of a bool	(not $x == 5$) is False
and	returns True only if both parts of the expression are true	($x > 3$ and $x < 7$) is True
or	returns True if at least one part of the expression is True	($x < 3$ or $x > 7$) is False ($x < 3$ or $x < 7$) is True

Type Conversion Functions

int(x)	turn x into an integer value. int can fail, e.g., int("abc") raises an error	int(`"123"`) == 123
float(x)	turn x into an float value, float can fail, e.g., float("abc") raises an error	float(`"2.46"`) == 2.46
str(x)	turn x into a string value	str(432) == `"432"`
type(x)	the type of x	type(1) == int type(1.2) == float

List Functions

lst.append(...)	append an element to lst, changing lst	[1,2,3].append(8) == [1,2,3,8]
lst.index(elt)	returns the first index of elt if it is in lst, otherwise causes an error	[1,2,3].index(2) == 1 [1,2,3].index(4) # causes an error
lst.count(elt)	return number of occurrences of elt in lst	[1,2,1,2,3].count(1) == 2
lst.sort()	mutate list so it is sorted. can use the optional key=FUNCTION parameter to modify how it sorts	lst = [(10, 9), (2, 7), (1, 8)] lst.sort() lst == [(1, 8), (2, 7), (10, 9)] lst.sort(key=max) lst == [(2, 7), (1, 8), (10, 9)]
sum(lst)	returns sum of elements in list lst	sum([1,2,4]) == 7
max(lst)	returns maximal element in lst, can use the optional key=FUNCTION parameter to modify how it finds maximum values	max([5,3,1,7,2]) == 7 max([51,12,43], key=lambda x: x%10) == 43
min(lst)	returns the minimum element in lst, can use the optional key=FUNCTION parameter to modify how it finds minimum values	max([5,3,1,7,2]) == 1 max([51,12,43], key=lambda x: x%10) == 51

Miscellaneous Functions

len(x)	length of sequence x, e.g., String/List	len(`"duke"`) == 4
range(x)	a sequence of integers starting at 0 and going up to but not including x	range(5) == [0, 1, 2, 3, 4]
range(start, stop)	a sequence of integers starting at start and going up to but not including stop	range(3, 7) == [3, 4, 5, 6]
range(start, stop, inc)	a sequence of integers starting at start and going up to but not including stop with increment inc	range(3, 9, 2) == [3, 5, 7]
list(str)	a list of the characters from string str	list(`"hat"`) == [`'h','a','t'`]
sorted(x)	return list that is sorted version of sequence/iterable x, doesn't change x. Can use optional parameter key==FUNCTION to change how the list is sorted.	sorted(`"cat"`) == [`'a','c','t'`] sorted([(1,4,2),(3,7,2),(8,2,9)], key=max) == [(1, 4, 2), (3, 7, 2), (8, 2, 9)]
min(x, y, z)	minimum value of all arguments	min(3, 1, 2) == 1 min(`"z", "b", "a"`) == `"a"`
max(x, y, z)	maximum value of all arguments	max(3, 1, 2) == 3 max(`"z", "b", "a"`) == `"z"`
abs(x)	absolute value of the int or float x	abs(-33) == 33 abs(-33.5) == 33.5

String Functions

Assume s = "colorful"

.find(str)	index of first occurrence	s.find("o") == 1 s.find("e") == -1
.rfind(str)	index of last occurrence	s.rfind("o") == 3 s.rfind("e") == -1
.count(str)	number of occurrences	s.count("o") == 2 s.count("r") == 1 s.count("e") == 0
.strip()	copy with leading/trailing whitespace removed	" big ".strip() == "big"
.split()	list of "words" in s separated by whitespace	"big bad dog".split() == ["big","bad", "dog"]
.split(",")	list of "items " in s that are separated by a comma In general can split on any string, not just a comma, e.g., s.split(":") will split on a colon and s.split("gat") will split on the string "gat".	"this,old,man".split(",") == ["this", "old", "man"]
' '.join(lst)	concatenate elements of lst, a list of strings, separated by ' ' or any string	':'.join(['a','b','c']) == "a:b:c"
.startswith(str)	boolean if starts with string	s.startswith("color") == True s.startswith("cool") == False
.endswith(str)	boolean if ends with string	s.endswith("ful") == True s.endswith("color") == False
.upper()	uppercase of s	s.upper() == "COLORFUL"
.lower()	lowercase of s	"HELLO".lower() == "hello"

Random Functions (import random)

random.randint(start, end)	Returns a random integer between start and end inclusive. Unlike range() and list slicing, the largest value it can return is end, not end-1. So random.randint(0,2) can return 0, 1, or 2.
random.random()	returns a random float between 0 and 1

Math Functions (import math)

math.pi	3.1415926535897931	
math.sqrt(num)	returns square root of num as float	math.sqrt(9) == 3.0

File Functions

open("filename")	opens a file, returns file object	f = open("foo.txt")
f.readlines()	returns the entire file as a list of lines (strings)	s = f.readlines()
f.close()	close the file f	

Set Functions

set(lst)	returns a set of the elements from list lst	set([1,1,5,7,7]) == {1,5,7}
s.add(item)	adds the item into the set, and returns nothing.	s = set() s.add(1) s.add(1) s == {1}
s.update(lst)	adds the elements in the list lst into the set, and returns nothing.	s = set([1,2]) s.update([1,4,5]) s == {1,2,4,5}
s.remove(item)	removes the item from the set, error if item not there.	s = set([1,4,6]) s.remove(1) s = {4,6}
s.union(t)	returns new set representing s UNION t, i.e., all elements in either s OR t, t can be any iterable (e.g., a list)	s = set([3,4,5]) t = set([1,2,3]) s.union(t) == {1,2,3,4,5}
s.intersection(t)	returns new set representing s INTERSECT t, i.e., only elements in both s AND t, t can be any iterable (e.g., a list)	s = set([3,4,5]) t = set([1,2,3]) s.intersection(t) == {3}
s.difference(t)	returns new set representing s difference t, i.e., elements in s that are not in t	s = set([3,4,5]) t = set([1,2,3]) s.difference(t) == {4,5}
s.symmetric_difference(t)	returns new set representing elements in s or t, but not in both	s = set([3,4,5]) t = set([1,2,3]) s.symmetric_difference(t) == {1,2,4,5}
$s\|t$	returns/evaluates to union of s and t, both must be sets.	Equivalent: s.union(t)
s & t	returns/evaluates to intersection of s and t, both must be sets.	Equivalent: s.intersection(t)
s - t	returns/evaluates to set with all elements in s that are not in t	Equivalent: s.difference(t)
$s \wedge t$	returns/evaluates to set with all elements from s and t that are not in both s and t	Equivalent: s.symmetric_difference(t)

Dictionary Functions

Assume d = {'a':1, 'b':2, 'c':3}

d[key]	returns the value associated with key, error if key not in dictionary d	d['a'] == 1 d['hello'] raises Error
d.get(key)	returns value associated with key, returns None if key not in dictionary d	d.get('b') == 2 d.get('hello') == None
d.get(key,default)	returns value associated with key, returns default if key not in d	d.get('d', -1) == -1
d.keys()	returns a list/view of the keys in dictionary	d.keys() == ['a', 'b', 'c']
d.values()	returns a list/view of values in dictionary	d.values() == [1, 2, 3]
d.items()	returns a list/view of tuples, (key,item) pairs from dictionary	d.items() == [('a',1), ('b',2), ('c',3)]