School Nova

Computer Science

First Semester overview
Sets and Dictionaries

12/15/2019

By Oleg Smirnov

First semester core topics and commands, p1

- Using Python editor, saving your code in a file, running your code
- Arithmetic operators
- Comparison operators
- Logical operators: and, or, not
- String, integer, float, Boolean: definitions and basic operations
- Str(), int(), float(), bool(), type(), id()
- Print() and f-strings
- Input()
- Try-except-else structure, purpose, application
- Iteration: while, continue, break
- Iteration: for loop, range(), len()

First semester core topics and commands, p2

- Conditional: if else, if elif else
- Lists: format, indexing, slicing, nested list
- List operations: append, extend, remove, insert, pop, del, in, not in
- List copy() and deepcopy(). Difference between '=' and copy()
- Differences between lists, tuples, sets, and dictionaries
- Type conversion: list(), tuple(), set(), dict(). Empty data structures.
- Set methods: union(), update(), intersection(), difference(), and so on*
- Dictionary: using/updating keys, accessing/adding elements
- Dictionary methods: get(), items(), keys(), pop(), update(), values() *

** you don't need to memorize all the methods but you should know of their existence and be able to use with a reference such as https://docs.python.org/3.8/library/stdtypes.html#set*

Core data structures recap

List: [2, 4, 6]	**ordered** (index) and ***mutable***; can contain **any object** can be nested
Tuple: (2, 4, 6)	**ordered** (index) and ***immutable***; can contain **any object** can be nested
Set: {2, 4, 6}	**unordered** (no index) and ***mutable***; contains **unique** and **immutable** objects can **NOT** be nested
Dictionary: {"two": 2, "four": 4}	**unordered** and ***mutable***; **BUT** can by indexed a “key”; “keys” must be **immutable** can contain **any object** can be nested

Core set methods (part 1)

x = {1, 2, 3}	=>	{1, 2, 3}
x.add(5)	=>	{1, 2, 3, 5}
x.discard(2)	=>	{1, 3, 5}
x.remove(1)	=>	{3, 5}

Note: discard is better than remove since no error if the element is absent

x.update([8])	=>	{3, 5, 8}

Note: for update, the argument must be an iterable object, [8], {8}, or (8,)

x.copy()	=>	{3, 5, 8} # returns a shallow copy
y = x	=>	{3, 5, 8}

Note: '=' creates another reference to the same object. Changing x you also change y and vice versa.

x.pop()	=>	{3, 5} or {3, 8} or {5, 8}

Note: for sets, the pop method removes a random element only; does not accept an argument

x.clear()	=>	{ } # empty set

Core set methods (part 2)

x.isdisjoint(y)	True if sets x and y do NOT have an intersection
	True if: x = {1, 2}, y = {3, 4}
x.issubset(y)	True if x is a subset of y
	True if: x = {1, 2}, y = {1, 2, 3, 4}
x.issuperset(y)	True if x contains y
	True if: x = {1, 2, 3, 4}, y = {1, 2}
x.intersection(y)	Returns a set that is the intersection of x and y
	if x = {1, 2, 3}, y = {2, 3, 4} => {2, 3}
x.union(y)	Return a set that is the union of x and y
	if x = {1, 2, 3}, y = {2, 3, 4} => {1, 2, 3, 4}
x.difference(y)	Returns a set that is the difference between x and y
	(that is, removes the intersection of x and y from x)
	x = {1, 2, 3}, y = {2, 3, 4} => {1}
x.symmetric_difference(y)	Returns a set which is a union of x.difference(y) and y.difference(x)
	x = {1, 2, 3}, y = {2, 3, 4} => {1, 4}

Core dictionary methods

D = {"name": "Canada", "capital": "Ottawa", "population": 37.6, "area": 3.86}

D[“area”] or *D.get(“area”)* => returns 3.86

D[“area”] = 3.87 => changes the value

for i in D: print(i)	*for i in D: print(D[i])*
Name	Canada
Capital	Ottawa
Population	37.6
area	3.86

D[“largest city”] = “Toronto” # adds a new entry to the dictionary, as long as # the “largest city” is a **new key**

del D[“population”] # deletes a key and its value

Many standard list and set methods work with dictionaries, for example:
in, not in, pop(), clear(), copy(), update(), len(), and so on.