PYTHON BASIC QUESTIONS

https://www.datacamp.com/community/tutorials/18-most-common-python-list-questions-learn-python#question1

1. When To Use Lists And When To Use Tuples, Dictionaries Or Sets
2. How To Select An Element From A List
3. How To Transform Lists Into Other Data Structures
4. How To Determine The Size Of Your List
5. What’s The Difference Between The Python append() and extend() Methods?
6. How To Concatenate Lists
7. How To Sort a List
8. How To Clone Or Copy A List
9. How Does List Comprehension Work In Python?
10. How To Count Occurrences Of A List Item In Python
11. How To Split A List Into Evenly Sized Chunks
12. How To Loop over A List
13. How To Create Flat Lists Out Of Lists
14. How To Get An Intersection Of Two Lists
15. How To Remove Duplicates From A List
16. Why NumPy Instead Of Lists?
17. How To Create Empty NumPy Arrays
18. How To Do Math With Lists

**1. When To Use Python Lists And When To Use Tuples, Dictionaries Or Sets**

The introduction seems pretty straightforward when you’re just reading it, but when you’re actually working on a small python script or a whole project, the choice for a list or some other sequence type might not be as clear to you.

However, choosing the right data structure for your data is essential!

Keep on reading to find out more.

**Lists Versus Tuples**

Tuples are used to collect an immutable ordered list of elements. This means that:

* You can’t add elements to a tuple. There’s no append() or extend() method for tuples,
* You can’t remove elements from a tuple. Tuples have no remove() or pop() method,
* You can find elements in a tuple since this doesn’t change the tuple.
* You can also use the in operator to check if an element exists in the tuple.

So, if you’re defining a constant set of values and all you’re going to do with it is iterate through it, use a tuple instead of a list. It will be faster than working with lists and also safer, as the tuples contain “write-protect” data.

**Lists Versus Dictionaries**

* A list stores an ordered collection of items, so it keeps some order. Dictionaries don’t have any order.
* Dictionaries are known to associate each key with a value, while lists just contain values.

Use a dictionary when you have an unordered set of unique keys that map to values.

**Note** that, because you have keys and values that link to each other, the performance will be better than lists in cases where you’re checking membership of an element.

**Lists Versus Sets**

* Just like dictionaries, sets have no order in their collection of items. Not like lists.
* Set requires the items contained in it to be hashable, lists store non-hashable items.
* Sets require your items to be unique and immutable. Duplicates are not allowed in sets, while lists allow for duplicates and are mutable.

## 2. How To Select An Element From A List

If you want who work properly with these lists, you will need to know how to access them.

You typically access lists to change certain values, to update or delete them, or to perform some other kind of operation on them. The way you access lists, and, by extension, all other sequence types, is by using the index operator [ ]. Inside, you put an integer value.

### How To Get The Last Element Of A List In Your List

The answer to this question is an addition to the explanation in the previous section.

Try putting a negative value, such as -1 or -2 to the index operator to retrieve the last element

### What Does The Index Out Of Range Error Mean?

This error is one that you will see quite often, especially when you’re new to programming or new to Python.

The best way to understand what this error means is to try it out for yourself.

Use the interactive learning widget above and try passing excessively small negative or large positive numbers!

As you can see, you might get an “Index Out Of Range” error in cases where you pass an integer value to the index operator that is bigger or way smaller than your list! It means that you are assigning a value or referring to an index that does not exist (yet).

### The Slice Notation In Lists

When you’re new to programming or Python, this can be one particularly confusing topic.

In general, you use the slice notation when you want to select more than one list element at a time. Much like when you select just one element from a list, you use the double brackets. What’s so special now is that instead of just an integer, you also put a : in between the double brackets

### How To Randomly Select An Element In A List

# Import `choice` from the `random` library

from \_\_\_\_ import \_\_\_\_

# Construct your `list` variable with a list of the first 4 letters of the alphabet

list = ['\_','\_','\_','\_']

# Print your random 'list' element

print(choice(list))

## 3. How To Transform Python Lists Into Other Data Structures

Sometimes, a list is not exactly what you need. In these cases, you might want to know how to transform your list to a more appropriate data structure. Below, we list some of the most common transformation questions and their answers.

### How To Convert A List To A String

You convert a list to a string by using ''.join(). This operation allows you to glue together all strings in your list together and return them as a string.

listOfStrings = ['One', 'Two', 'Three']

strOfStrings = ''.join(listOfStrings)

print(strOfStrings)

### How To Convert A List To A Tuple

You can change a list to a tuple in Python by using the tuple() function. Pass your list to this function and you will get a tuple back!

**Remember**: tuples are immutable. You can’t change them afterwards!

### How To Convert Your List To A Set In Python

As you will remember, a set is an unordered collection of unique items. That means not only means that any duplicates that you might have had in your original list will be lost once you convert it to a set, but also the order of the list elements.

You can change a list into a set with the set() function.

# Pass your list to `tuple()`

tuple(\_\_\_)

# Transform your list into a set

set(\_\_\_)

**How To Convert Lists To A Dictionaries**

A dictionary works with keys and values, so the conversion from a list to a dictionary might be less straightforward. Let’s say you have a list like this:

helloworld=[‘hello’,’world’,’1’,’2’]

You will need to make sure that ‘hello’ and ‘world’ and ‘1’ and ‘2’ are interpreted as key-value pairs. The way to do this is to select them with the slice notation and pass them to zip().zip() actually works like expected: it zips elements together. In this case, when you zip the helloWorld elements helloWorld[0::2] and helloWorld[1::2], your output will be: list(zip(helloWorld))

**Note** that you need to use list() to print out the result of the zip() function.

You will pass this to the dict() function, which will interpret hello as a key and world as a value. Similarly, 1 will be interpreted as a key and 2 as a value.

**Note** that an iterable object can give you an iterator. The iterator, which has a .\_\_next\_\_ method, holds information on where exactly you are in your iteration: it knows what is the next element in the iteration.

## 4. How To Determine The Size Of Your List in Python

You can pass your list to the len() function to get the length of your list back.

len(\_\_\_\_)

**Note** that the len() function is not specifically to retrieve the lenght of your lists, but can also be used with other sequences or collections, such as dictionaries, sets, strings, etc.

an iterable is a value that represents a sequence of one more values. lists are sequences and all instances of Python’s sequence types are iterables.

extend(), on the one hand, takes an iterable (that’s right, it takes a list, set, tuple or string!), and adds each element of the iterable to the list one at a time.

append(), on the other hand, adds its argument to the end of the list as a single item, which means that when the append() function takes an iterable as its argument, it will treat it as a single object.

## 6. How To Concatenate Lists in Python

To concatenate lists, you use the + operator.

## 7. How To Sort a List in Python

There are two very simple ways to get the values in your lists sorted in ascending or descending order:

* You use the sort() method
* Or you use the sorted() function and pass a list to it

The sorted() function can be applied to any Iterable object, which means that it also accepts strings, sets, dictionaries when they are passed to it!

## 8. How To Clone Or Copy A List in Python

There are a lot of ways of cloning or copying a list:

* You can slice your original list and store it into a new variable: newList = oldList[:]
* You can use the built-in list() function: newList = list(oldList)
* You can use the copy library:
  + With the copy() method: newList = copy.copy(oldList)
  + If your list contains objects and you want to copy those as well, you can use copy.deepcopy(): copy.deepcopy(oldList)

## 9. How Does List Comprehension Work In Python?

List comprehension is, basically speaking, a way of elegantly constructing your lists. The best thing about this for those who love math is the fact that they look a lot like mathematical lists.

[x\*\*2 for x in range(10)]

Out[1]: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

Lambda It’s an anonymous function /one line function

## 10. How To Count Occurrences Of A List Item In Python

# Count the occurrences of the letter "a"

list = ["d", "a", "t", "a", "c", "a", "m", "p"]

list.count("a")

### Counting all list items with Counter()

Alternatively, there’s the faster Counter() method from the [collections library](https://docs.python.org/3/library/collections.html):

# Import `Counter` from the `collections` library

from collections import Counter

# This is your list

list = ["a","b","b"]

# Pass `list` to `Counter()`

Counter(list)

Out[1]: Counter({'a': 1, 'b': 2})

## 11. How To Split A Python List Into Evenly Sized Chunks

To split your list up into parts of the same size, you can resort to the zip() function in combination with iter():

# Your list `x`

x = [1,2,3,4,5,6,7,8,9]

# Split `x` up in chunks of 3

y = zip(\*[iter(x)]\*3)

# Use `list()` to print the result of `zip()`

list(y)

Out[1]: [(1, 2, 3), (4, 5, 6), (7, 8, 9)]

## 12. How To Loop over A List in Python

mylist = [[1,2,3],[4,5,6,7],[8,9,10]]

# Loop over your list and print all elements that are of size 3

for x in mylist:

if len(x)==3:

print(x)

[1, 2, 3]

[8, 9, 10]

## 13. How To Create Flat Lists Out Of Lists

To make a simple list out of a list of lists, you can use the sum() function.

# Your initial list of lists

list = [[1,2],[3,4],[5,6]]

# Flatten out your original list of lists with `sum()`

sum(list, [])

Out[1]: [1, 2, 3, 4, 5, 6]

## 14. How To Get An Intersection Of Two Python Lists

### List Intersection Using List Comprehension

If you want to obtain the intersection of two lists, you can use the filter() function.

intersection = [list(filter(lambda x: x in list1, sublist)) for sublist in list2]

# Print the result of the intersection

print(intersection)

[[13, 32], [7, 13, 28], [1, 6]]

### Intersecting Lists With set()

print(list(set(list1) & set(list2)))

# Use `intersection()`

print(list(set(list1).intersection(list2)))

[32, 1, 6, 7, 13, 28]

[32, 1, 6, 7, 13, 28]

## 15. How To Remove Duplicates From A List in Python

# Your list with duplicate values

duplicates = [1, 2, 3, 1, 2, 5, 6, 7, 8]

# Print the unique `duplicates` list

print(list(set(duplicates)))

# A list with small numbers

smallNumbers = [1, 2, 3]

# Print the unique `duplicates` list without the small numbers

list(set(duplicates) - set(smallNumbers))

[1, 2, 3, 5, 6, 7, 8]

Out[1]: [8, 5, 6, 7]

## 16. Why NumPy Instead Of Python Lists?

In general, there seem to be four reasons why Python programmers prefer NumPy arrays over lists in Python:

* because NumPy arrays are more compact than lists.
* because access in reading and writing items is faster with NumPy.
* because NumPy can be more convenient to work with, thanks to the fact that you get a lot of vector and matrix operations for free
* because NumPy can be more efficient to work with because they are implemented more efficiently.

## 17. How To Create Empty NumPy Arrays

import numpy

numpy.array([])

Out[1]: array([], dtype=float64)