



HACKWAGON
• ACADEMY •



DATA SCIENCE 101: PYTHON COLLECTIONS

INTRODUCTION

- Why do we need collections?
- What are collections?
- Commonly used collections:
Lists, Tuples & Dictionary

Create. Conqu



HACKWAGON
• ACADEMY •

WHY DO WE NEED COLLECTIONS?



- There are times where we need to store multiple values in a variable for easier access

```
despacito = 15000  
see_you_again = 5000  
im_yours = 3000
```

despacito

15000

see_you_again

5000

im_yours

3000

Imagine YouTube have
millions of videos

WHAT IS A COLLECTION?



- A data type that allows you to store many values in a single “variable”
- A collection is useful because we can carry many values around in one convenient package
- The following are lists, which is one type of collection:

```
video_views = [15000,5000,3000]  
video_titles = ['despacito', 'see_you_again', 'im_yours']
```

TYPES OF COLLECTIONS



- There are three main type of commonly used Python collections:
 - Lists
 - Tuples
 - Dictionary

LISTS

- Rules of lists
- Slicing lists
- Concatenating lists
- Checking something in a list
- Length of list
- Range function and list



HACKWAGON
• ACADEMY •

LIST CAN HOLD MULTIPLE VALUES



- A list is a compound data type that contains a collection of sequential related data items
- Think of list as a “book” that holds a series of paper (variables)
- Best-Practice: List should contain data of same type.

```
video_views = [15000,5000,3000]  
video_titles = ['despacito', 'see_you_again',  
                'im_yours']
```

INITIATING A LIST



- List constants are surrounded by square brackets and the elements in the list are separated by commas
- The lists will be created with the stipulated order specified in between in the square brackets
- A list element can have mixed data type - even another list
- A list can be empty
- The values of a list can be printed using the print() function

```
>>> print([1,24,76])
[1, 24, 76]

>>> print(['red', 'yellow', 'blue'])
['red', 'yellow', 'blue']

>>> print(['red',23,98.8282828])
['red',23,98.8282828]

>>> print([1, [5,6], 7])
[1, [5,6], 7]

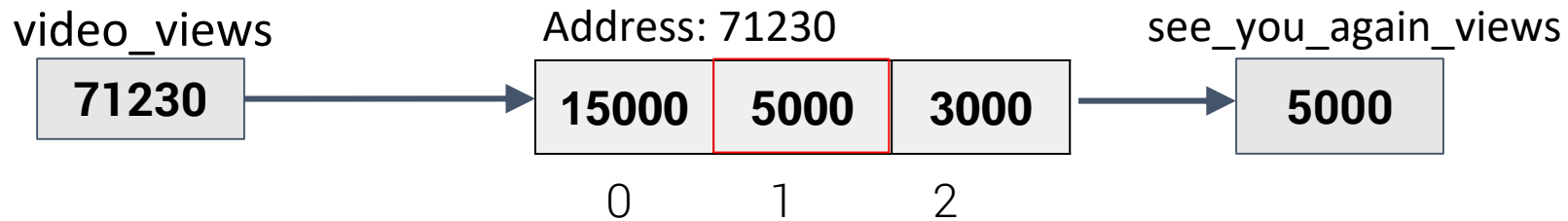
>>> print([])
[]
```


HOW TO ACCESS THE DATA IN LIST



- The items in list are arranged in an index based system. Hence we can just access a data via the index

```
video_views = [15000,5000,3000]  
video_titles = ['despacito', 'see_you_again', 'im_yours']  
see_you_again_views = video_views[1]  
print(see_you_again_views)
```



HOW TO ADD ITEMS TO A LIST



- The following code is how you will add the data to the end of the created list

```
video_views = [15000,5000,3000]
video_titles = ['despacito', 'see_you_again', 'im_yours']

video_views.append(6000)
video_titles.append('slam_dunk')
```

LIST ARE MUTABLE: CHANGING DATA IN A LIST



- Strings are "immutable" - we cannot change the contents of a string - we must make a new string to make any change
- Lists are "mutable" - we can change an element of a list using the index operator

```
>>> fruit = 'Banana'
>>> fruit[0] = 'b'
TypeError: 'str' object does not support item assignment

>>> x = fruit.lower()
>>> print(x)
banana

>>> toto = [2,3,23,26,43,45]
>>> print(toto)
[2,3,23,26,43,45]

>>> toto[0] = 42
>>> print(toto)
[42,3,23,26,43,45]
```

LISTS CAN BE CONCATENATED



- We can create a new list by adding two existing lists together

```
>>> a = [1,2,3]
>>> b = [4,5,6]
>>> c = a + b
>>> print(c)
[1,2,3,4,5,6]

>>> print(a)
[1,2,3]
```

LISTS CAN BE SLICED



- Just like in strings, the second number is "up to but not including"

```
>>> x = [2, 4, 6, 8, 10, 12]
>>> x[1:3]
[4, 6]

>>> x[:4]
[2, 4, 6, 8]

>>> x[3:]
[8, 10, 12]

>>> x[:]
[2, 4, 6, 8, 10, 12]
```

CHECKING WHETHER SOMETHING IS IN THE LIST



- Python provides two operators that let you check if an item is in a list
- These are logical operators that return True or False
- They do not modify the list

```
>>> num = [2, 4, 6, 8, 10]
>>> 2 in num
True

>>> 99 in num
False

>>> 7 not in num
True
```


HOW LONG IS THE LIST?



- The `len()` function takes a list as a parameter and returns the number of elements in the list
- Actually `len()` tells us the number of elements of any set or sequence (i.e. such as a string...)

```
>>> self_praise = 'I am handsome'
>>> print(len(self_praise))
13

>>> x = [7, 8, 'john']
>>> print(len(x))
3
```

RANGE FUNCTION AND LIST



- The range function returns a list of numbers that range from zero to one less than the parameter

```
>>> print(range(4))
>>> [0,1,2,3]

>>> class = ['john', 'roy']
print(len(class))
2

>>> print(range(len(class)))
[0,1]
```

OTHER LIST METHODS



- You can make use of the method “dir()” to find out the methods inherent to Python’s list

```
>>> x = list()
>>> type(x)
<type 'list'>

>>> dir(x)
['append', 'count', 'extend', 'index', 'insert', 'pop', 'remove',
'reverse', 'sort']
```

IN-CLASS PRACTICE: LIST METHODS*



- Interesting lists operations to play with

```
test_list = [1,['a','b','c'],3,4]
#add an new item to the back of the list
test_list.append(5)
#merge two list or insert multiple items to the list
test_list.extend([5,6])
#retrieve the 3rd item in the list
test_list[2]
#retrieve the 2nd item in the nested list
test_list[1][1]
#remove the item with the value 1
test_list.remove(1)
#remove the 3rd item in the list
del test_list[2]
#remove and return the 2nd item in the list
test_list.pop(1)
#retrieve the number of items in the list
len(test_list)
```

IN-CLASS QUESTIONS: UN-NESTING THE NESTED



- Given the following nested list, make use of the list index to output all the names within the list

```
youtube_videos = [ [1, 'despacito', 15000], [2, 'see_you_again', 5000], [3, 'im_yours', 3000] ]
```

IN-CLASS QUESTIONS: VIDEO VIEW LOOKUP APP



- Suppose you have 2 list that contain titles and views of 1 million videos respectively, how can we retrieve the video views for 'im_yours'?

```
video_views = [15000,5000,3000]  
video_titles = ['despacito', 'see_you_again', 'im_yours']
```


LIST SUMMARY



- We learnt the following about lists:
- List is a reference variables that can hold multiple values
- Rules of lists (Initiating, accessing, adding, changing)
- Lists can be sliced
- Checking if something is in a list
- Length of list
- Range and list

TUPLES

- What are tuples?
- Things you cannot do with tuples
- Methods inherent to tuples
- When to use tuples over lists?



HACKWAGON
• ACADEMY •

WHAT ARE TUPLES?




- Tuple functions the same as the list, except that the contents are immutable

List

```
video_views = [15000,5000,3000]  
view_views[0] = 16000
```

Tuple

```
video_views = (15000,5000,3000)  
view_views[0] = 16000
```



```
-----  
TypeError                                Traceback (most recent call last)  
<ipython-input-32-31c9f406f66f> in <module>()  
      1 student_weights = (70,65,45)  
----> 2 student_weights[0] = 100  
      3  
  
TypeError: 'tuple' object does not support item assignment
```

THINGS YOU CANNOT DO WITH TUPLE



- Tuple functions the same as the list, except that the contents are immutable

```
>>> x = (3, 2, 1)
```

```
>>> x.sort()
```

```
Traceback:AttributeError: 'tuple' object has no attribute 'sort'
```

```
>>> x.append(5)
```

```
Traceback:AttributeError: 'tuple' object has no attribute 'append'
```

```
>>> x.reverse()
```

```
Traceback:AttributeError: 'tuple' object has no attribute 'reverse'
```

METHODS INHERENT TO TUPLE



- Tuple functions the same as the list, except that the contents are immutable

```
>>> x = list()
>>> dir(x)
['append', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']

>>> t = tuple()
>>> dir(t)
['count', 'index']
```

WHEN TO USE TUPLES OVER LISTS?



- Since Python does not have to build tuple structures to be modifiable, they are simpler and more efficient in terms of memory use and performance than lists
- So in our program when we are making "temporary variables" we prefer tuples over lists.

IN-CLASS PRACTICE: LIST OF TUPLES



- Given the list of YouTube videos,
 - Print all titles and views
 - Print the views of 'im_yours'
 - Print all titles only
 - Add a new video "gangnam_style" which view is 8400 into the list

```
youtube_videos = [ ('despacito', 15000), ('see_you_again', 5000), ('im_yours', 3000) ]
```

SUMMARY OF TUPLES



- Tuples are an immutable version of lists
- Tuples are preferred over lists when we are creating temporary variables

DICTIONARY

- Dictionary vs list
- What is a dictionary?
- Properties of dictionary
- Functions of dictionary



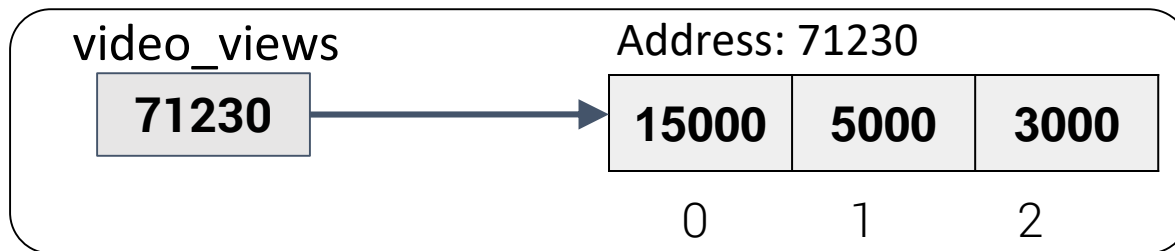
HACKWAGON
• ACADEMY •

DICTIONARY VS LIST

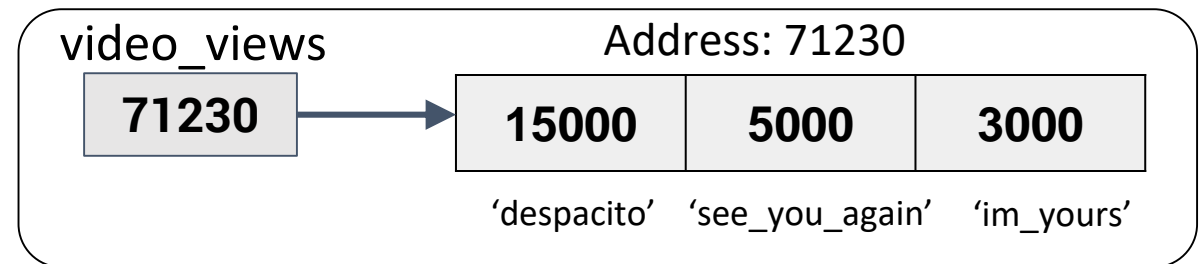


- Just like lists and tuples, dictionary is part of the collection family too
- However, there is one key difference in the way data is stored in dictionary versus lists
- Dictionaries are organized in key-value pairs
- Retrieval through lookups just like real dictionaries!

List



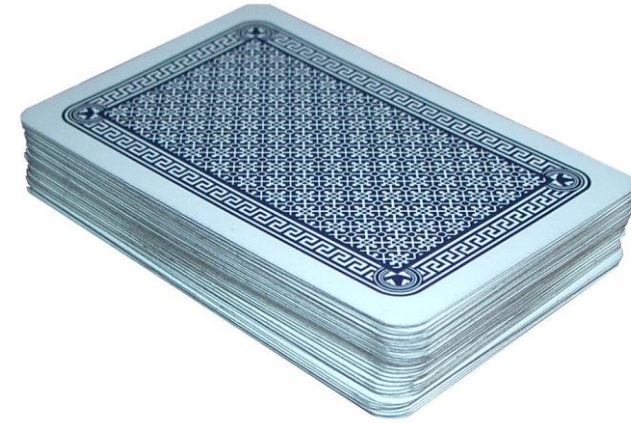
Dictionary



DICTIONARY VS LIST



- Think of lists as:
 - A linear collection of values that needs to stay in order



- Think of dictionaries as:
 - A “bag” of values, each with its own label

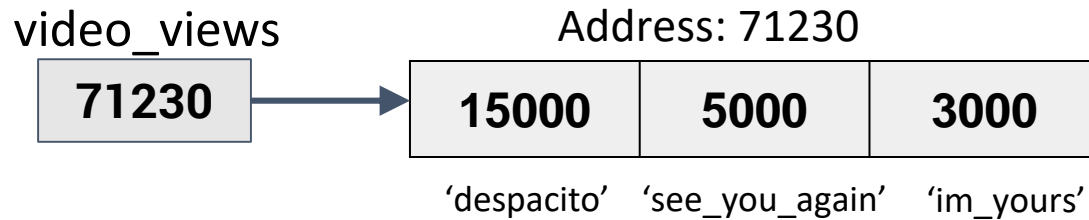


WHAT IS A DICTIONARY?



- Important note: Keys must be unique!

```
video_views = {'despacito':15000, 'see_you_again':5000,  
'im_yours':3000}  
print(video_views['despacito'])
```



PROPERTIES OF DICTIONARY



- Dictionaries are Python's most powerful data collection
- Dictionaries allow us to do fast database-like operations in Python
- Dictionaries have different names in different languages
 - Associative Arrays - Perl / Php
 - Properties or Map or HashMap - Java
 - Property Bag - C# / .Net

FUNCTIONS OF DICTIONARY*



- Explore the following options:

```
stock_prices = {'Apple':94.34, 'Facebook':39.28, 'Google':154.67}
#retrieve the value of an item base on its key
stock_prices['Apple']
#return a list of keys
stock_prices.keys()
#check if a key exist in the dictionary
'Alibaba' in stock_prices
#add a new item to the dictionary
stock_prices['Alibaba'] = 101.22
#update an item in the dictionary
stock_prices['Facebook'] = 42.5
```

IN-CLASS PRACTICE: EXTRACTING LIST FROM DICTIONARY*



- Given the following list of historical three days video views data of two videos, find the average views per month of each videos:

```
hist_video_views = {'despacito':[15000,9800,10100], 'im_yours':[2500,1400,14100]}
```

APPLICATION OF COLLECTIONS: LIST / TUPLE



- How are collections applied or used for data science?

	A	B	C
1	Product	Quantity Sold	Price
2	Squishy Banana	20000	1
3	Unicorn cushion	8000	23.7
4	Sushi Roller	5000	8

Imagine your company have the following excel data and you want to port it over to python to do further analysis

[[‘Squishy Banana’, 20000, 1],
[‘Unicorn cushion’, 8000, 23.7],
[‘Sushi Roller’, 5000, 8]]

We can actually port it over into a nested list, with each inner list representing one row of data!

We will teach the methods to read csv excel/csv data into python for processing in week 5!

APPLICATION OF COLLECTIONS: LIST + DICTIONARY



- How are collections applied or used for data science?

	A	B	C
1	Product	Quantity Sold	Price
2	Squishy Banana	20000	1
3	Unicorn cushion	8000	23.7
4	Sushi Roller	5000	8

Imagine your company have the following excel data and you want to port it over to python to do further analysis

There are many ways of combining different collection types to store tabular data. This and the previous slides seeks to illustrate just that. Along the way we will learn more about the best way of storing data!

[{Product: 'Squishy Banana', Quantity Sold: 20000, Price: 1},
{Product: 'Unicorn cushion', Quantity Sold: 8000, Price: 23.7},
[Product: 'Sushi Roller', Quantity Sold: 5000, Price: 8}]

We can actually port it over into a nested list, with each inner list representing one row of data!

IN-CLASS PRACTICE: EXTRACTING DATA FROM MIXED TYPE*



- Given the following list of historical three days video views data of two videos, find the average views per month of each videos:

```
hist_video_views = {'despacito':[15000,9800,10100], 'im_yours':[2500,1400,14100]}
```

SUMMARY*



- View jupyter notebook for summary of collections

SUMMARY

- What are collections?
- Lists
- Tuples
- Dictionaries
- Application of Collections

