

A stylized map of Australia is shown in the background, with various colored regions. Overlaid on the map are several snakes: a purple snake in the top left, a green snake in the top right, and a blue snake in the bottom right. The text "BEDROCK DATA SCIENCE" is centered in a yellow box with a red border.

BEDROCK DATA SCIENCE

Session 2: Data structures

What is **numpy** ?

What is **with** ?

What is a **dictionary** ?

What is a **tuple** ?

What is **len()** ?

What is **list comprehension** ?

What is train & test set ?

What is **sum** ?

```
1 # import required libraries
2 import numpy as np
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 import csv
6 from sklearn.linear_model import LinearRegression
7
8 # Read the 'Advertising.csv' dataset
9 with open('Advertising.csv', mode='r') as infile:
10     reader = csv.reader(infile)
11     values = [(rows[1], rows[4]) for rows in reader]
12     data_dictionary = {i[0]:i[1:] for i in list(zip(values))}
13 # Assign TV advertising as predictor variable 'x' and sales as response variable 'y'
14 tv, sales = data_dictionary['TV'], data_dictionary['sales']
15 x, y = np.array(tv, dtype='float32').reshape(-1, 1), np.array(sales, dtype='float32')
16
17 # Split the data into training and test sets
18 number_of_points = len(x)
19 train_size = 0.8
20 num_train_points = int(train_size*number_of_points)
21
22 # Create indices to split the dataset
23 train_index = np.random.choice(range(len(x)), size=num_train_points, replace=False)
24 test_index = [i for i in range(len(x)) if i not in train_index]
25 test_index = np.array(test_index)
26
27 # Create boolean masks for training and test sets
28 mask = np.zeros(len(x), dtype='int')
29 mask[train_index] = 1
30 mask = mask==1
31 # Use the masks to create train and test data
32 x_train, y_train = x[mask], y[mask]
33 x_test, y_test = x[~mask], y[~mask]
34
35 # Write a function to compute the mean squared error of the predictions
36 def mse(y_true, y_prediction):
37     error = y_true - y_prediction
38     squared_error = error**2
39     mean_squared_error = 1/len(y_true)*sum(squared_error).item(0)
40     return mean_squared_error
41
42 # Use the sklearn function 'LinearRegression' to fit on the training set
43 model = LinearRegression()
44 model.fit(x_train, y_train)
45 # Now predict on the test set
46 y_pred_test = model.predict(x_test)
47
48 # Now compute the MSE with the predicted values and print it
49 test_mse = mse(y_test, y_pred_test)
50 print(f'The test MSE is {test_mse}')
```

What is **import** ?

What is **zip** ?

What is the ***** operator ?

What is a **conditional list comprehension** ?

What is **range()** ?

What is **item(0)** ?

```

8  # Read the 'Advertising.csv' dataset
9  with open('Advertising.csv', mode='r') as infile:
10     reader = csv.reader(infile)
11     values = [(rows[1],rows[4]) for rows in reader]
12     data_dictionary = {i[0]:i[1:] for i in list(zip(*values))}
13     # Assign TV advertising as predictor variable 'x' and sales as response variable 'y'
14     tv,sales = data_dictionary['TV'], data_dictionary['sales']
15     x,y = np.array(tv,dtype='float32').reshape(-1,1), np.array(sales,dtype='float32')
16
17     # Split the data into training and test sets
18     number_of_points = len(x)
19     train_size = 0.8
20     num_train_points = int(train_size*number_of_points)
21
22     # Create indices to split the dataset
23     train_index = np.random.choice(range(len(x)),size=num_train_points,replace=False)
24     test_index = [i for i in range(len(x)) if i not in train_index]
25     test_index = np.array(test_index)
26

```

What is a
tuple?

What is a dictionary?

What is list comprehension?

What is the zip operator?

What is the * operator?

Outline

- Data Structures
 - Sets
 - Tuples
 - Dictionaries
- Pythonic Syntax
 - List comprehension
 - Set comprehension
 - Dictionary comprehension
- Special built-in functions
 - Enumerate
 - Zip

Data Structures

Why do we need data structures?



In a land far away, where books were abundant...



The Grand Keeper of Books

Oh no, how am I going to go through this chaos?

Uh...your Highness, of course. Just give me a moment.

The Great keeper, I need the book "Become a Data Scientist in 30 days"



Why do we need data structures?



The Grand Keeper of
Books

Eureka! Let me
use my magic to
arrange them in a
row – this will be
easy!

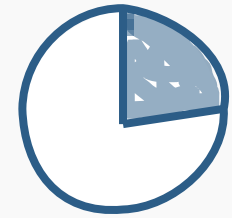
Like a List



Why do we need data structures?



Searching for the book



Time it took

Like a List



Found the
book!

Why do we need data structures?



The Grand Keeper of
Books

Here you
go, your
Highness!

Like a List



I need another
book "Zen of
Python" - could
you please get it
for me?

Why do we need data structures?



The Grand Keeper of
Books

Hmm... "Zen" ...let
me search for it.

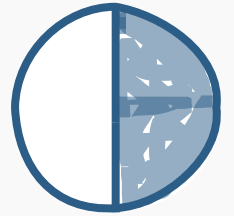
Like a List



Why do we need data structures?



Like a List



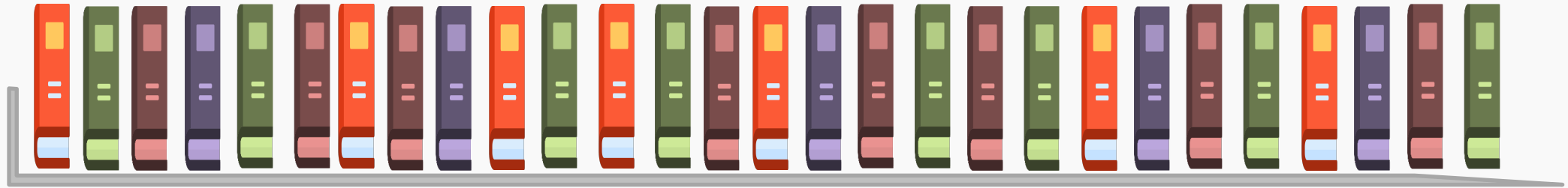
Time it took



Why do we need data structures?

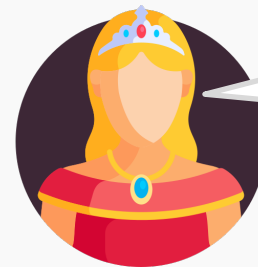


Like a List



The Grand Keeper of
Books

Here you go,
your Highness!



Thank you once again!

...although, respected Keeper,
that was slow. Father is
sending 10,000 more books
your way!

Why do we need data structures?

10,000 books!!!? How
am I going to arrange it
so that I can fetch
these books fast & also
give information about
it?



The Grand Keeper of
Books

Like a List



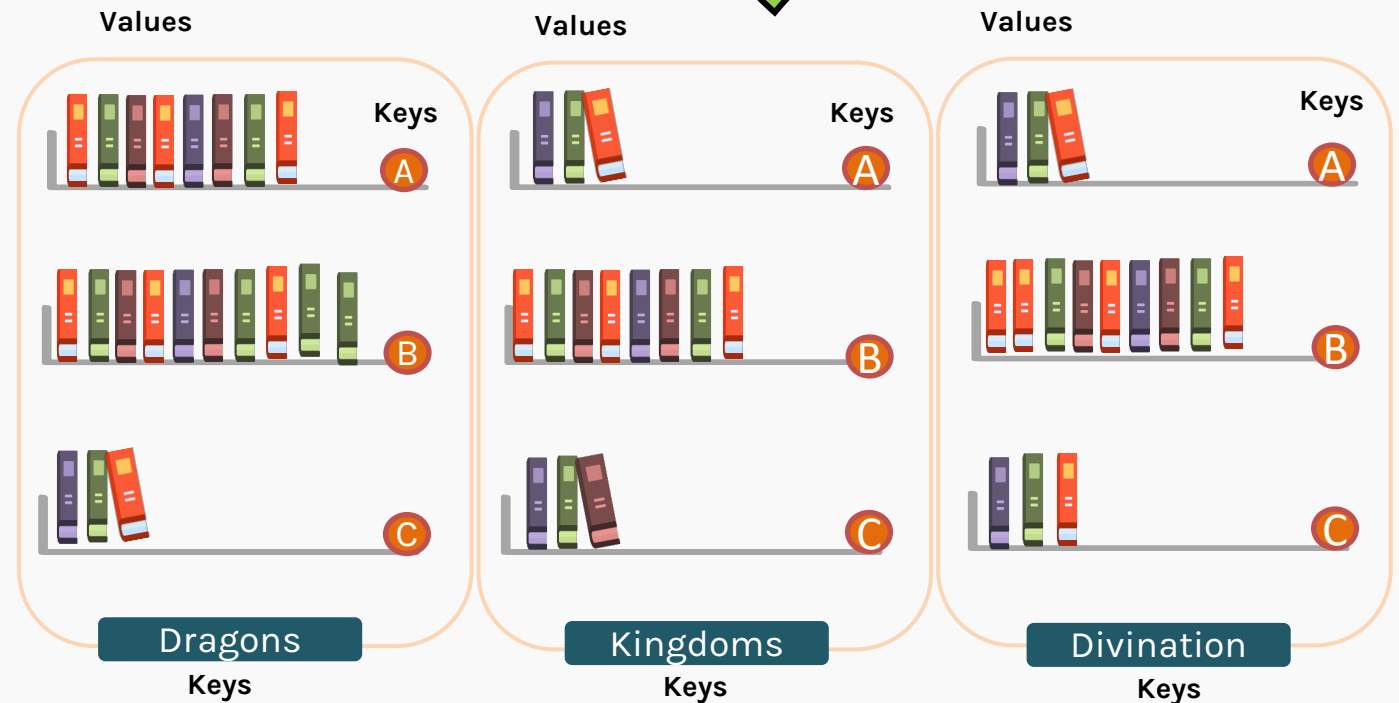
Why do we need data structures?



The Grand Keeper of Books

Let me build
a library
instead

Like a List



This process is called dictionaries in Python.



Dictionaries in Python – you will use them a lot!

In Python, dictionaries are **ordered** collection of **mutable objects with immutable keys**.
Elements within a dictionary have a **key & value**.

$$\text{Dict} = \{\text{Key}_1:\text{Value}_1, \dots, \text{Key}_n:\text{Value}_n\}$$

```
>>> dictA = {'India':'New Delhi', 'USA':'Washington DC', 'Germany':'Berlin', 'Sri Lanka':'Colombo'}

>>> dictB = {'Apples':1, 'Pineapple':4, 'Grapes':3}

>>> print(dictA)
{'India': 'New Delhi', 'USA': 'Washington DC', 'Germany': 'Berlin', 'Sri Lanka': 'Colombo'}

>>> print(dictB)
{'Apples': 1, 'Pineapple': 4, 'Grapes': 3}
```

DictA

Key - Type	Value - Type
India<str>	New Delhi<str>
USA<str>	Washington DC<str>
Germany<str>	Berlin<str>

DictB

Key - Type	Value - Type
Apples<str>	1 <int>
Pineapple<str>	4 <int>
Grapes <str>	3 <int>

Dictionaries in Python

Important Dictionary Methods to Remember

Python Code	Function
<pre>dictA = {'India':'Delhi','USA':'Washington'}, dictA = {'name':'Sam','age':23}</pre>	Creates a dictionary with the key value pair provided

Dictionaries in Python

Important Dictionary Methods to Remember

Python Code	Function
<pre>dictA = {'India':'Delhi','USA':'Washington'}, dictA = {'name':'Sam','age':23}</pre>	Creates a dictionary with the key-value pair provided
<pre>dictA.items()</pre>	Gets a list of the key-value pairs in the dictionary

Dictionaries in Python

Important Dictionary Methods to Remember

Python Code	Function
<pre>dictA = {'India': 'Delhi', 'USA': 'Washington'}, dictA = {'name': 'Sam', 'age': 23}</pre>	Creates a dictionary with the key-value pair provided
<pre>dictA.items()</pre>	Gets a list of the key-value pairs in the dictionary
<pre>dictA.values()</pre>	Gets a list of the values in the dictionary

Dictionaries in Python

Important Dictionary Methods to Remember

Python Code	Function
<pre>dictA = {'India': 'Delhi', 'USA': 'Washington'}, dictA = {'name': 'Sam', 'age': 23}</pre>	Creates a dictionary with the key-value pair provided
<pre>dictA.items()</pre>	Gets a list of the key-value pairs in the dictionary
<pre>dictA.values()</pre>	Gets a list of the values in the dictionary
<pre>dictA.keys()</pre>	Gets a list of the keys in the dictionary

Dictionaries in Python

Important Dictionary Methods to Remember

Python Code	Function
<pre>dictA = {'India': 'Delhi', 'USA': 'Washington'}, dictA = {'name': 'Sam', 'age': 23}</pre>	Creates a dictionary with the key value pair provided
<pre>dictA.items()</pre>	Gets a list of the key-value pairs in the dictionary
<pre>dictA.values()</pre>	Gets a list of the values in the dictionary
<pre>dictA.keys()</pre>	Gets a list of the keys in the dictionary
<pre>dictA['place of birth'] = 'Singapore'</pre>	Sets the value of the key associated with it
<pre>del dictA['age']</pre>	Deletes the key-value pair from the dictionary

Tuples in Python

In Python, tuples are **ordered** collections of **immutable** objects. They are like lists but the difference is that they are **immutable** and enclosed with **()** instead of **[]**.

$$\text{Tuple} = (e_1, e_2 \dots e_N)$$

```
#Initialize a tuple
```

```
In [2]: tupleA = ('India', 'USA', 'Germany')
```

```
#Initialize another tuple
```

```
In [3]: tupleB = ('violet', 'indigo', 'blue', 'yellow', 'orange', 'red')
```

```
In [5]: tupleA + tupleB
```

```
Out[5]:
```

```
('India', 'USA', 'Germany', 'violet', 'indigo', 'blue', 'yellow', 'orange', 'red')
```

Program execution is faster when manipulating a tuple than it is for the equivalent list.

Tuples in Python



Important to remember – Since parentheses are also used to define operator precedence in expressions, Python evaluates a number within () as an object of its tuple. But if you want to tell Python to define a singleton tuple, you need to put a comma.

Type of the Variable

In [7]: a = (2,3)

Type(a) - tuple

In [8]: b = (2)

Type(b) - integer

In [9]: c = ('string')

Type(c) - string

In [10]: d = (2,)

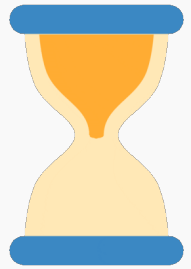
Type(d) - tuple

In [10]: d = (2,)

Type(e) - integer

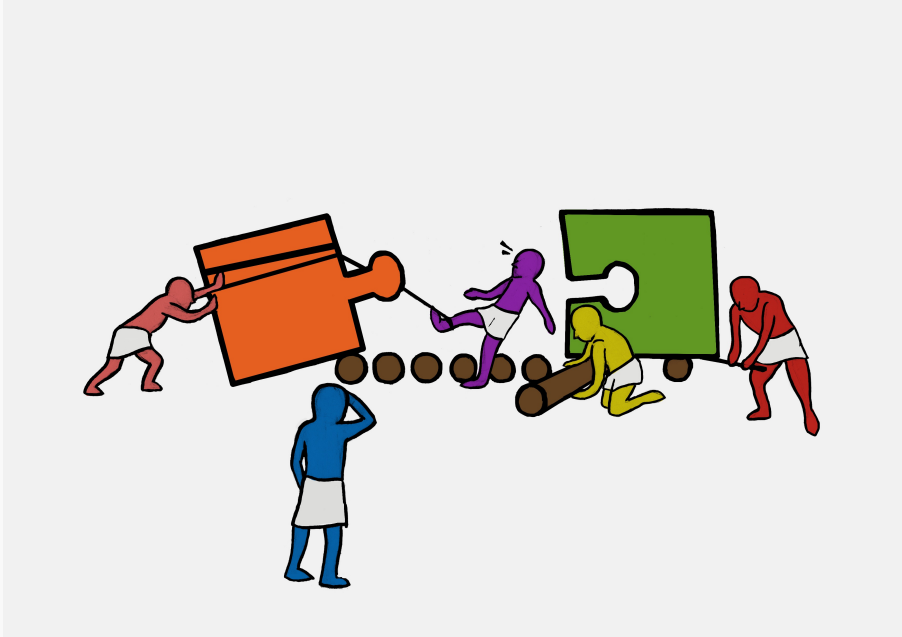
In [11]: e,f = 2,3

Type(f) - integer



Digestion Time

Exercise



Exercise #3

Description

🏆 Exercise: The Book Keeper's Apprentice - Dicts & List

Congratulations on becoming the book keeper's apprentice! The highness has once again come to the library asking for a book. Unfortunately, the great keeper is not in today and the new books aren't arranged yet. You need to do all the work.

The aim of this exercise is to understand the implications of using a dictionary over a list and getting comfortable creating dictionaries.

Instructions

Import the necessary libraries and the data required for this exercise that is stored in `books_dict`, `authors_list`, `books_list`.

