دوره دیتا ساینس کاربردی

Data Collection for Space X

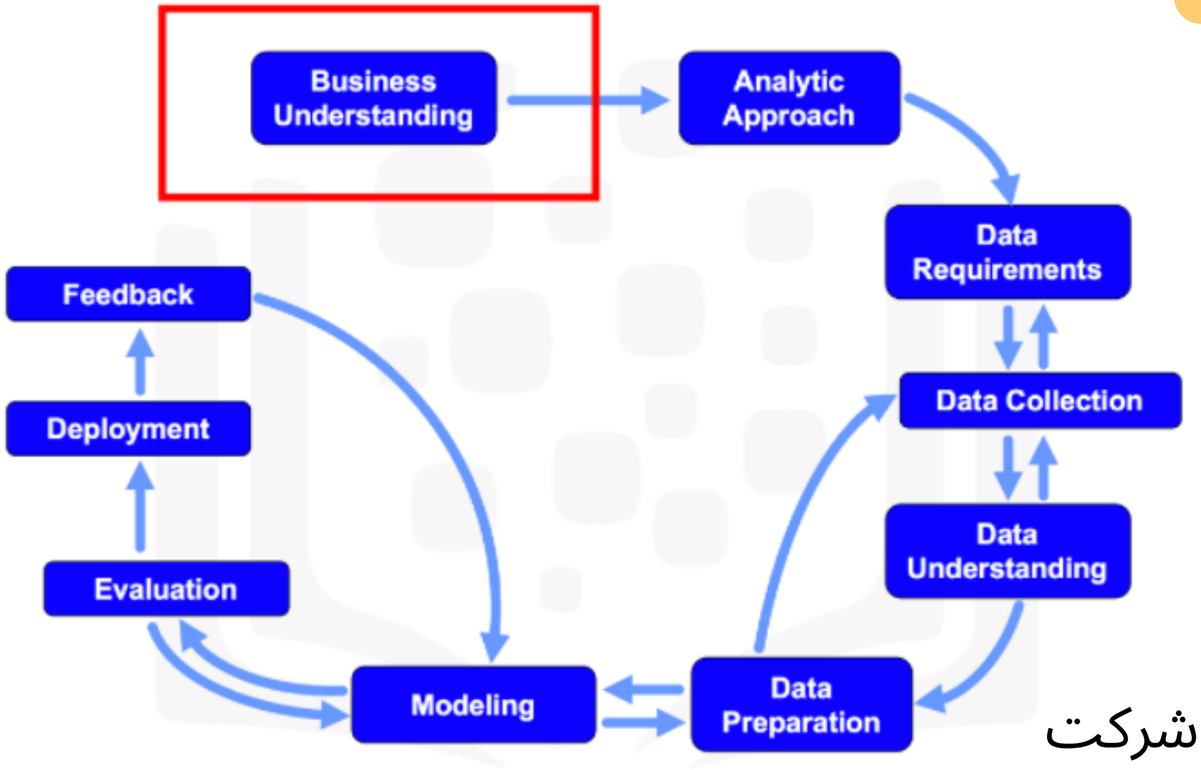
dataroadmap

مدرس: مونا حاتمی

جلسه یازدهم

Space X Project

Capston Project Overview

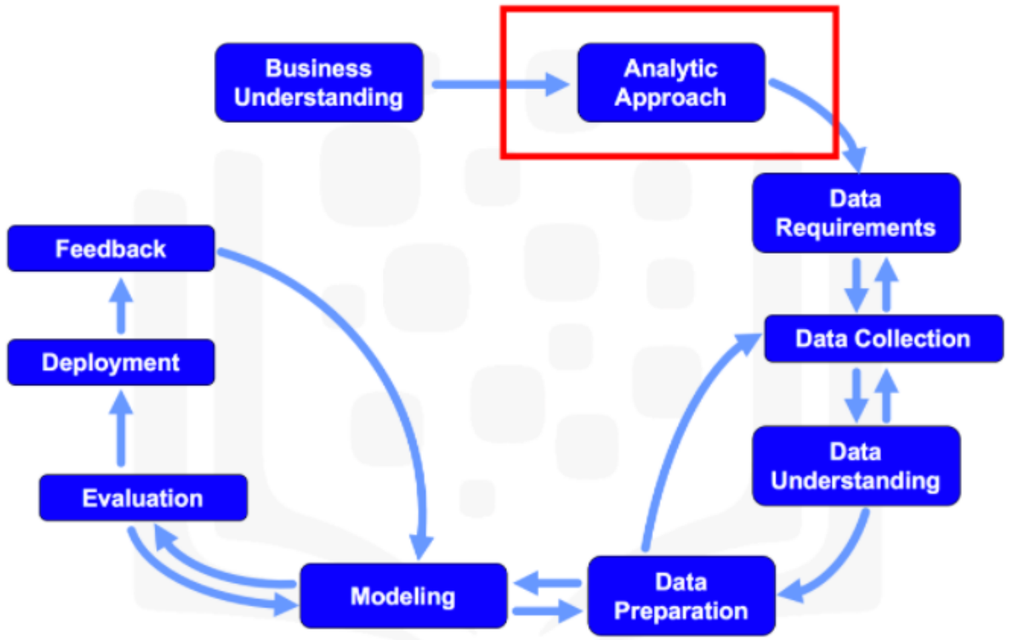


تخمین هزینه مسافرت فضایی برای شرکت Space Y

شرکت SpaceX ادعا کرده که موشکFalcon9 با $62M به ایستگاه بین المللی فضایی می رسونه در حالیکه رقبای دیگه این شرکت مبلغ $165M را اعلام کردند.

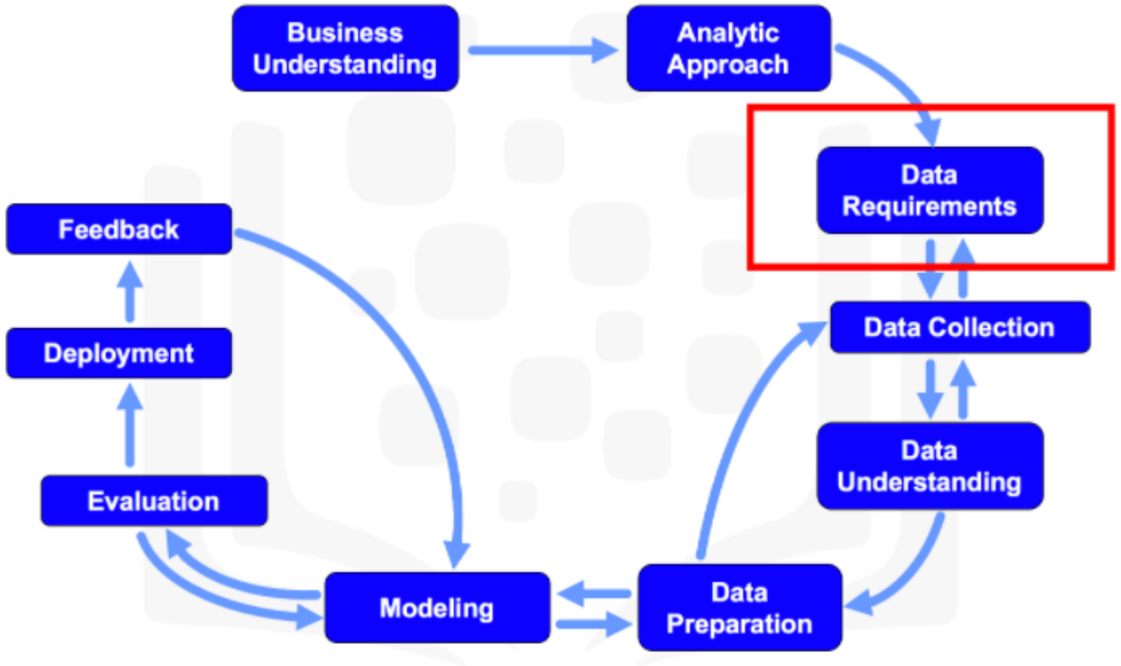
دلیل ارزون بودن قیمت شرکت اسپیس ایکس اینه که این شرکت میتونه استیج اول رو دوباره استفاده کنه.

Capston Project Overview



- پیش بینی اینکه آیا اسپیس ایکس مجدد از فاز اول استفاده میکنه یا نه؟

Capston Project Overview



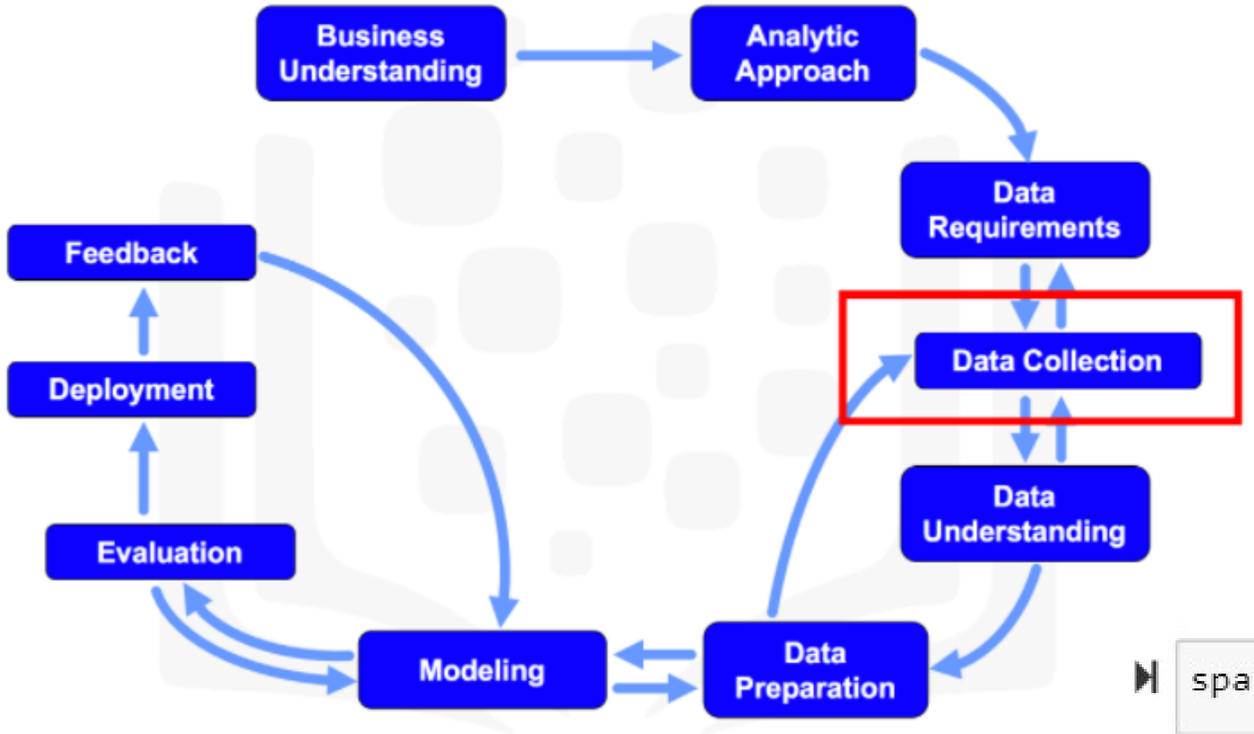
اطلاعات مورد نیاز از وبسایت اسپیس ایکس

SpaceX

SpaceX designs, manufactures and launches advanced rockets and spacecraft.

SpaceX

Capston Project Overview




```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
response = requests.get(spacex_url)
```

Import Libraries

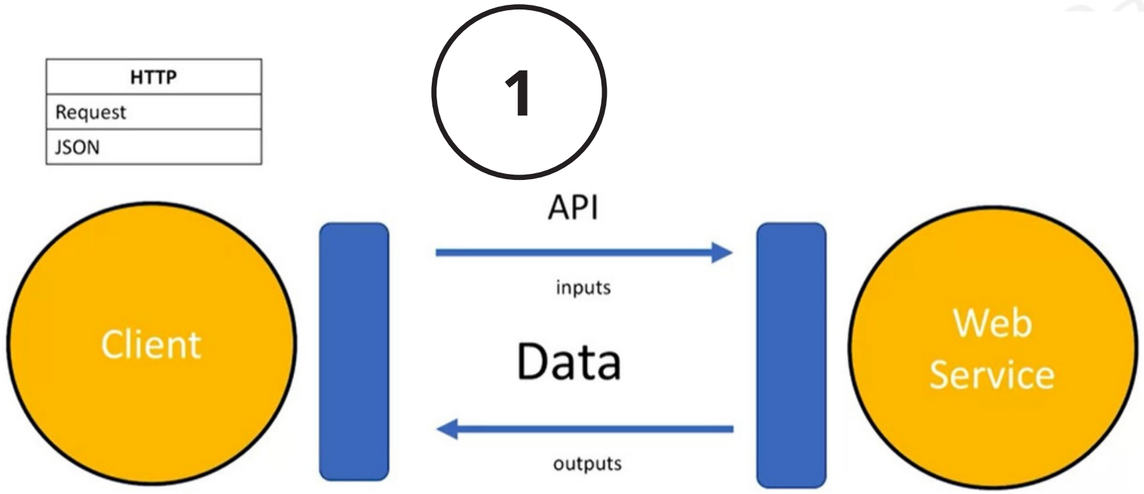
```
# Requests allows us to make HTTP requests which we will use to get data fro
import requests
# Pandas is a software library written for the Python programming language f
import pandas as pd
# NumPy is a library for the Python programming language, adding support for
import numpy as np
# Datetime is a library that allows us to represent dates
import datetime

# Setting this option will print all collumns of a dataframe
pd.set_option('display.max_columns', None)
# Setting this option will print all of the data in a feature
pd.set_option('display.max_colwidth', None)
```

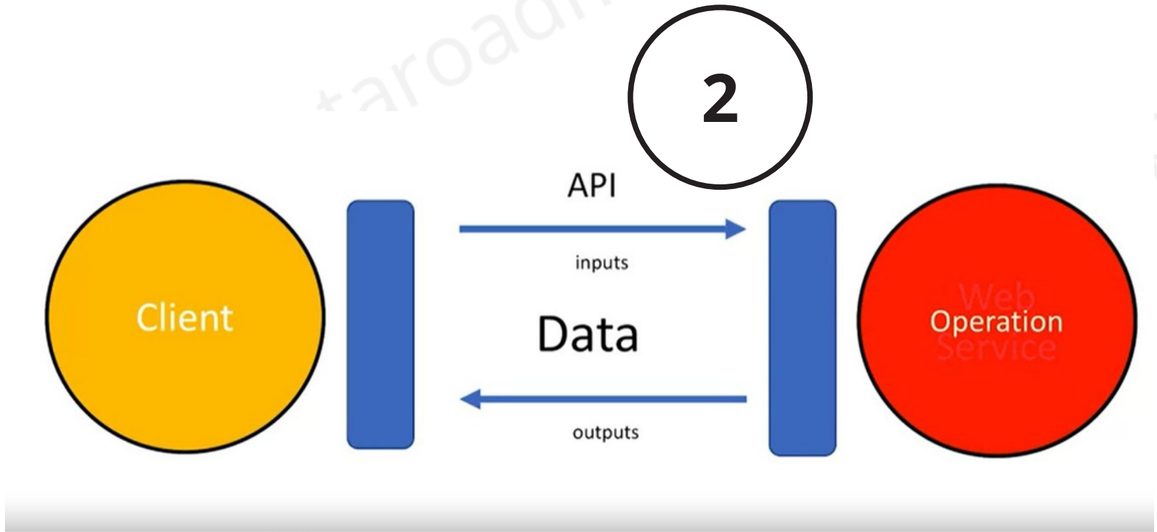
Requesting data following URL

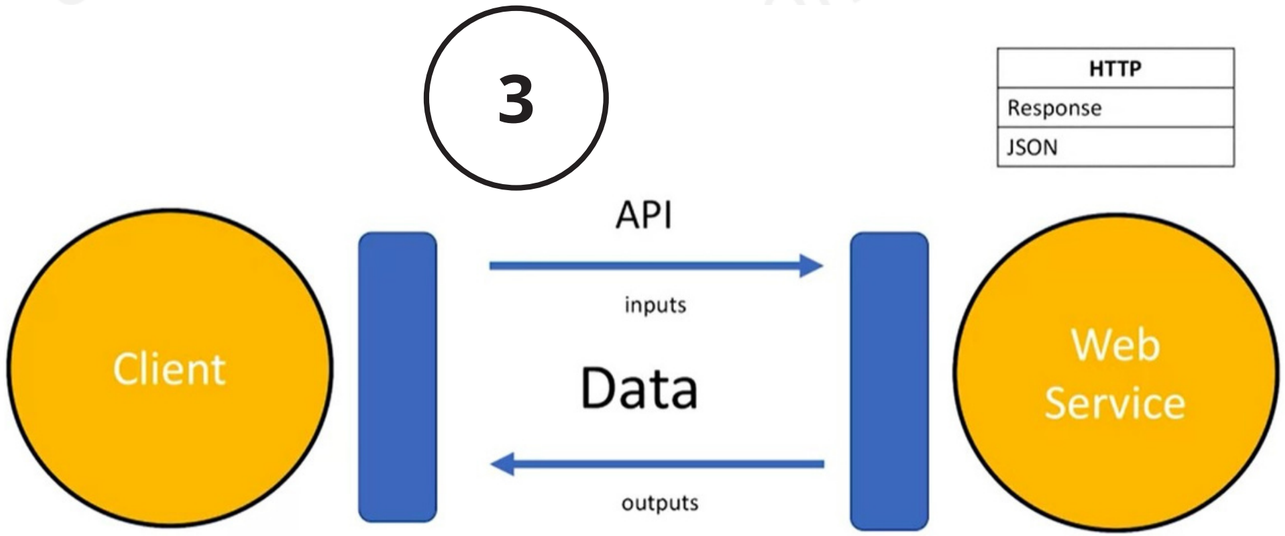
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
response = requests.get(spacex_url)
```

Requesting data from API







Response Statuse Code

```
response
```

```
<Response [200]>
```

```
response.status_code
```

```
200
```

2XX

Successful Requests

200 OK
201 Created
202 Accepted
203 Non-Authoritative Information
204 No Content
205 Reset Content
206 Partial Content
207 Multi-Status
208 Already Reported

Response Statuse Code

```
response.content
```

```
e,"legs":false,"reused":false,"landing_attempt":false,"landing_success":null,"landing_type":null,"landpad":nul
to_update":true,"tbd":false,"launch_library_id":null,"id":"5eb87cdaffd86e000604b32b"},{"fairings":{"reused":fa
covery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"https://images2.imgbox.com/6c/c
hHs_o.png","large":"https://images2.imgbox.com/4a/80/k1oAkY0k_o.png"},"reddit":{"campaign":null,"launch":null,
a":null,"recovery":null},"flickr":{"small":[],"original":[]},"presskit":null,"webcast":"https://www.youtube.co
h?v=v0w9p3U8860","youtube_id":"v0w9p3U8860","article":"http://www.spacex.com/news/2013/02/11/falcon-1-flight-3
n-summary","wikipedia":"https://en.wikipedia.org/wiki/Trailblazer_(satellite)"},"static_fire_date_utc":null,"s
ire_date_unix":null,"net":false,"window":0,"rocket":"5e9d0d95eda69955f709d1eb","success":false,"failures":[{"t
0,"altitude":35,"reason":"residual stage-1 thrust led to collision between stage 1 and stage 2"}],"details":"R
```

```
data=response.json()
data
```

```
5]: [{'fairings': {'reused': False,
    'recovery_attempt': False,
    'recovered': False,
    'ships': []},
   'links': {'patch': {'small': 'https://images2.imgbox.com/94/f2/NN6Ph45r_o.png',
     'large': 'https://images2.imgbox.com/5b/02/QcxHUb5V_o.png'},
    'reddit': {'campaign': None,
     'launch': None,
     'media': None,
```

json file to dataframe

```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(data)
```

```
# Get the head of the dataframe
data.head(5)
```

]:

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	capsules	
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 33, 'altitude': None, 'reason': 'merlin engine failure'}]	Engine failure at 33 seconds and loss of vehicle	[]	[]	[]	[5eb0e4b5b
1	None	NaN	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 301, 'altitude': 289, 'reason': 'harmonic	Successful first stage burn and transition to second stage, maximum altitude 289 km, Premature	[]	[]	[]	[5eb0e4b6b

json file to dataframe

1

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	S
0	1	2010-06-04	Falcon 9	6123.547647	LEO	CCSFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCSFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCSFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B

2

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	capsules	
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 33, 'altitude': None, 'reason': 'merlin engine failure'}]	Engine failure at 33 seconds and loss of vehicle	[]	[]	[]	[5eb0e4b5b
1	None	NaN	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 301, 'altitude': 289, 'reason': 'harmonic	Successful first stage burn and transition to second stage, maximum altitude 289 km, Premature	[]	[]	[]	[5eb0e4b6b

data.info()

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 187 entries, 0 to 186
Data columns (total 43 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   static_fire_date_utc        121 non-null    object 
 1   static_fire_date_unix       121 non-null    float64
 2   net                         187 non-null    bool   
 3   window                      117 non-null    float64
 4   rocket                      187 non-null    object 
 5   success                     186 non-null    object 
 6   failures                    187 non-null    object 
 7   details                     134 non-null    object 
 8   crew                        187 non-null    object 
 9   ships                       187 non-null    object 
 10  capsules                    187 non-null    object 
 11  payloads                    187 non-null    object 
 12  launchpad                   187 non-null    object 
 13  flight_number               187 non-null    int64  
 14  name                        187 non-null    object 
 15  date utc                    187 non-null    object 
```

Keep some columns

```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]
```

```
data.head()
```

:

	rocket	payloads	launchpad	cores	flight_number	date_utc
0	5e9d0d95eda69955f709d1eb	[5eb0e4b5b6c3bb0006eeb1e1]	5e9e4502f5090995de566f86	[{'core': '5e9e289df35918033d3b2623', 'flight': 1, 'gridfins': False, 'legs': False, 'reused': False, 'landing_attempt': False, 'landing_success': None, 'landing_type': None, 'landpad': None}]	1	2006-03-24T22:30:00.000Z
1	5e9d0d95eda69955f709d1eb	[5eb0e4b6b6c3bb0006eeb1e2]	5e9e4502f5090995de566f86	[{'core': '5e9e289ef35918416a3b2624', 'flight': 1, 'gridfins': False, 'legs': False, 'reused': False, 'landing_attempt': False, 'landing_success': None, 'landing_type': None, 'landpad': None}]	2	2007-03-21T01:10:00.000Z
2	5e9d0d95eda69955f709d1eb	[5eb0e4b6b6c3bb0006eeb1e3, 5eb0e4b6b6c3bb0006eeb1e4]	5e9e4502f5090995de566f86	[{'core': '5e9e289ef3591814873b2625', 'flight': 1, 'gridfins': False, 'legs': False, 'reused': False, 'landing_attempt': False, 'landing_success': None, 'landing_type': None, 'landpad':	3	2008-08-03T03:34:00.000Z

data.info()

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 187 entries, 0 to 186
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   rocket         187 non-null    object
 1   payloads       187 non-null    object
 2   launchpad      187 non-null    object
 3   cores          187 non-null    object
 4   flight_number  187 non-null    int64 
 5   date_utc       187 non-null    object
dtypes: int64(1), object(5)
memory usage: 8.9+ KB
```

Explore dataset

```
data['cores'][0]
```

```
[{'core': '5e9e289df35918033d3b2623',
  'flight': 1,
  'gridfins': False,
  'legs': False,
  'reused': False,
  'landing_attempt': False,
  'landing_success': None,
  'landing_type': None,
  'landpad': None}]
```

```
len(data['cores'][0])
```

```
1
```

.map()

```
data['cores'].map(len)
```

```
0      1
1      1
2      1
3      1
4      1
      ..
182    1
183    1
184    1
185    1
186    1
Name: cores, Length: 187, dtype: int64
```

```
set(data['cores'].map(len))
```

```
{1, 3}
```

'payloads' & 'cores'

payloads	launchpad	cores
[5eb0e4b5b6c3bb0006eeb1e1]	5e9e4502f5090995de566f86	[{'core': '5e9e289df35918033d3b2623', 'flight': 1, 'gridfins': False, 'legs': False, 'reused': False, 'landing_attempt': False, 'landing_success': None, 'landing_type': None, 'landpad': None}]
[5eb0e4b6b6c3bb0006eeb1e2]	5e9e4502f5090995de566f86	[{'core': '5e9e289ef35918416a3b2624', 'flight': 1, 'gridfins': False, 'legs': False, 'reused': False, 'landing_attempt': False, 'landing_success': None, 'landing_type': None, 'landpad': None}]

Filter some rows

```
set(data['payloads'].map(len))
```

```
{1, 2, 3, 16}
```

```
# We will remove rows with multiple cores because those
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]
```

lambda

```
# Since payloads and cores are lists of size 1 we will also
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])
```

```
data[['cores','payloads']]
```

]:

	cores	payloads
0	{'core': '5e9e289df35918033d3b2623', 'flight': 1, 'gridfins': False, 'legs': False, 'reused': False, 'landing_attempt': False, 'landing_success': None, 'landing_type': None, 'landpad': None}	5eb0e4b5b6c3bb0006eeb1e1
1	{'core': '5e9e289ef35918416a3b2624', 'flight': 1, 'gridfins': False, 'legs': False, 'reused': False, 'landing_attempt': False, 'landing_success': None, 'landing_type': None, 'landpad': None}	5eb0e4b6b6c3bb0006eeb1e2
3	{'core': '5e9e289ef3591855dc3b2626', 'flight': 1, 'gridfins': False, 'legs': False, 'reused': False, 'landing_attempt': False, 'landing_success': None, 'landing_type': None, 'landpad': None}	5eb0e4b7b6c3bb0006eeb1e5
4	{'core': '5e9e289ef359184f103b2627', 'flight': 1, 'gridfins': False, 'legs': False, 'reused': False, 'landing_attempt': False, 'landing_success': None, 'landing_type': None, 'landpad': None}	5eb0e4b7b6c3bb0006eeb1e6

'date_utc'

t	payloads	launchpad	cores	flight_number	date_utc
)	[5eb0e4b5b6c3bb0006eeb1e1]	5e9e4502f5090995de566f86	[{'core': '5e9e289df35918033d3b2623', 'flight': 1, 'gridfins': False, 'legs': False, 'reused': False, 'landing_attempt': False, 'landing_success': None, 'landing_type': None, 'landpad': None}]	1	2006-03-24T22:30:00.000Z
)	[5eb0e4b6b6c3bb0006eeb1e2]	5e9e4502f5090995de566f86	[{'core': '5e9e289ef35918416a3b2624', 'flight': 1, 'gridfins': False, 'legs': False, 'reused': False, 'landing_attempt': False, 'landing_success': None, 'landing_type': None, 'landpad': None}]	2	2007-03-21T01:10:00.000Z

```
# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

variables

```
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

Booster Version

```
BoosterVersion
```

```
[]
```

```
rockets = requests.get("https://api.spacexdata.com/v4/rockets/").json()
```

```
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

for loop

```
a=[2,5,9]
for i in a:
    print(1)
```

```
1
1
1
```

```
a=[2,5,9]
for i in a:
    print(1+i)
```

```
3
6
10
```

Booster Version

```
response1 = requests.get("https://api.spacexdata.com/v4/rockets/"+str(data['rocket'][0])).json()
response1
```

```
: {'height': {'meters': 22.25, 'feet': 73},
   'diameter': {'meters': 1.68, 'feet': 5.5},
   'mass': {'kg': 30146, 'lb': 66460},
   'first_stage': {'thrust_sea_level': {'kN': 420, 'lbf': 94000},
    'thrust_vacuum': {'kN': 480, 'lbf': 110000},
    'reusable': False,
    'engines': 1,
    'fuel_amount_tons': 44.3,
    'burn_time_sec': 169},
   'second_stage': {'thrust': {'kN': 31, 'lbf': 7000},
    'payloads': {'composite_fairing': {'height': {'meters': 3.5, 'feet': 11.5},
      'diameter': {'meters': 1.5, 'feet': 4.9}},
     'option_1': 'composite fairing'},
    'reusable': False,
    'engines': 1,
    'fuel_amount_tons': 3.38,
    'burn_time_sec': 378},
   'engines': {'isp': {'sea_level': 267, 'vacuum': 304},
    'thrust_sea_level': {'kN': 420, 'lbf': 94000},
    'thrust_vacuum': {'kN': 480, 'lbf': 110000},
    'number': 1,
    'type': 'merlin',
    'version': '1C',
    'layout': 'single',
    'engine_loss_max': 0,
    'propellant_1': 'liquid oxygen',
    'propellant_2': 'RP-1 kerosene',
    'thrust_to_weight': 96},
   'landing_legs': {'number': 0, 'material': None},
   'payload_weights': [{'id': 'leo',
     'name': 'Low Earth Orbit',
     'kg': 450,
     'lb': 992}],
   'flickr_images': ['https://imgur.com/DaCfMsj.jpg',
    'https://imgur.com/azYafd8.jpg'],
   'name': 'Falcon 1',
   'type': 'rocket',
```

Booster Version

```
# Call getBoosterVersion
getBoosterVersion(data)
```

the list has now been update

```
BoosterVersion
```

```
228]: ['Falcon 1',
       'Falcon 1',
       'Falcon 1',
       'Falcon 1',
       'Falcon 1',
       'Falcon 9',
       'Falcon 9',
       'Falcon 9',
       'Falcon 9',
       'Falcon 9',
       'Falcon 9',
       'Falcon 9',
       'Falcon 9',
       'Falcon 9',
       'Falcon 9',
       'Falcon 9',
       'Falcon 9',
       'Falcon 9',
       'Falcon 9',
```

Payloadmass & Orbit

```
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

Payloadmass & Orbit

```
response2 = requests.get("https://api.spacexdata.com/v4/payloads/"+data['payloads'][0]).json()
response2
```

```
5]: {'dragon': {'capsule': None,
   'mass_returned_kg': None,
   'mass_returned_lbs': None,
   'flight_time_sec': None,
   'manifest': None,
   'water_landing': None,
   'land_landing': None},
  'name': 'FalconSAT-2',
  'type': 'Satellite',
  'reused': False,
  'launch': '5eb87cd9ffd86e000604b32a',
  'customers': ['DARPA'],
  'norad_ids': [],
  'nationalities': ['United States'],
  'manufacturers': ['SSTL'],
  'mass_kg': 20,
  'mass_lbs': 43,
  'orbit': 'LEO',
  'reference_system': 'geocentric',
  'regime': 'low-earth',
  'longitude': None,
  'semi_major_axis_km': None,
  'eccentricity': None,
  'periapsis_km': 400,
  'apoapsis_km': 500,
  'inclination_deg': 39
```

Payloadmass & Orbit

```
# Call getPayloadData
getPayloadData(data)
```

```
PayloadMass[0:4]
```

```
[20, None, 165, 200]
```

```
Orbit[0:4]
```

```
['LEO', 'LEO', 'LEO', 'LEO']
```

LaunchSite

```
# Takes the dataset and uses the launchpad column to call the API and append the data to the lis
def getLaunchSite(data):
    for x in data['launchpad']:
        response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
        Longitude.append(response['longitude'])
        Latitude.append(response['latitude'])
        LaunchSite.append(response['name'])
```

```
# Call getLaunchSite
getLaunchSite(data)
```

Cores

```
# Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
                Serial.append(None)
            Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
            Flights.append(core['flight'])
            GridFins.append(core['gridfins'])
            Reused.append(core['reused'])
            Legs.append(core['legs'])
            LandingPad.append(core['landpad'])
```

if in python

```
# it is not part of this project
a=10
b=20
if a==b:
    print(a-b)
else:
    print(a+b)
```

```
30
```

```
# it is not part of this project
a=10
b=20
if a==b:
    print(a-b)
elif a<b:
    print(a+b+20)
else:
    print(a+b)
```

```
50
```

Create Dictionary

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

```
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

Dictionary to dataframe

```
# Create a data from launch_dict
data2 = pd.DataFrame(launch_dict)
```

```
data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 94 entries, 0 to 93
Data columns (total 17 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   FlightNumber    94 non-null     int64  
 1   Date            94 non-null     object 
 2   BoosterVersion  94 non-null     object 
 3   PayloadMass     88 non-null     float64
 4   Orbit           94 non-null     object 
 5   LaunchSite      94 non-null     object 
 6   Outcome         94 non-null     object 
 7   Flights         94 non-null     int64  
 8   GridFins        94 non-null     bool   
 9   Reused          94 non-null     bool   
 10  Legs            94 non-null     bool   
 11  LandingPad      64 non-null     object 
```

Filter some rows

```
data_falcon9 = data2[data2['BoosterVersion']=='Falcon 9']
data_falcon9
```

]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flig
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	
5	8	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	
6	10	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	
7	11	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	
8	12	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	
...	...	...	...	...	...	...	...	
89	102	2020-09-03	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	
90	103	2020-10-06	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	
91	104	2020-10-18	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	

Filter some rows

```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

```
data_falcon9.to_csv('data_falcon9.csv', index=False)
```

Assignment:

تمرین:

کدهای ارائه شده در درس را بررسی و اجرا کنید.