Data Cleaning Basics: Takeaways

Syntax

READING A CSV IN WITH A SPECIFIC ENCODING

- Reading in a CSV file using Latin encoding:

```
laptops = pd.read_csv('laptops.csv',encoding='Latin-1')
```

- Reading in a CSV file using UTF-8:

```
laptops = pd.read_csv('laptops.csv',encoding='UTF-8')
```

- Reading in a CSV file using Windows-1251:

```
laptops = pd.read_csv('laptops.csv',encoding='Windows-1251')
```

MODIFYING COLUMNS IN A DATAFRAME

- Renaming An Existing Cplumn:

```
laptops["manufacturer"]= laptops["MANUfacturer"]
```

- Converting A String Column To Float:

```
laptops["screen_size"]= laptops["screen_size"].str.replace('"','').astype(float)
```

- Converting A String Column To Integer:

```
laptops["ram"] = laptops["ram"].str.replace('GB','')
laptops["ram"] = laptops["ram"].astype(int)
```

STRING COLUMN OPERATIONS

- Extracting Values From The Beginning Of Strings:

```
laptops["gpu_manufacturer"]= (laptops["gpu"]
                               .str.split(n=1,expand=True)
                               .iloc[:,0]
                          )
```

- Extracting Values From The End Of Strings:

```
laptops["cpu_speed_ghz"]= (laptops["cpu"]
                       .str.replace("GHz","")
                       .str.rsplit(n=1,expand=True)
                       .iloc[:,1]
                       .astype(float)
                       )
```

- Reordering Columns And Exporting Cleaned Data:

```
specific_order= ['manufacturer','model_name','category','screen_size_inches',
    'screen', 'cpu', 'cpu_manufacturer', 'cpu_speed','ram_gb',
    'storage_1_type','storage_1_capacity_gb','storage_2_type',
    'storage_2_capacity_gb','gpu', 'gpu_manufacturer','os',
    'os_version', 'weight_kg', 'price_euros']
reordered_df = laptops[specific_order]
reordered_df.to_csv("laptops_cleaned.csv",index=False)
```

FIXING VALUES

- Replacing Values Using A Mapping Dictionary:

  ``` mapping_dict = { 'Android': 'Android', 'Chrome OS': 'Chrome OS', 'Linux': 'Linux', 'Mac OS': 'macOS', 'No OS': 'No OS', 'Windows': 'Windows', 'macOS': 'macOS' }

  laptops["os"] = laptops["os"].map(mapping_dict)```
  ```

- Dropping Missing Values:

  ```
  laptops_no_null_rows = laptops.dropna(axis=0)
  ```

- Filling Missing Values:

  ```
  laptops.loc[laptops["os"] == "No OS", "os_version"] = "No OS"
  ```

Concepts

- Computers, at their lowest levels, can only understand binary. Encodings are systems for representing all other values in binary so a computer can work with them. The first standard was ASCII, which specified 128 characters. Other encodings popped up to support other languages, like Latin-1 and UTF-8. UTF-8 is the most common encoding and is very friendly to work with in Python 3.
- When converting text data to numeric data, we usually follow the following steps:
  - Explore the data in the column.
  - Identify patterns and special cases.
  - Remove non-digit characterse.
  - Convert the column to a numeric dtype.
  - Rename column if required.

Resources

- Python Encodings
- Indexing and Selecting Data