011-02-Python_basics_solution

April 18, 2024

1 011-02 - Python Basics - Solution Notebook

- Written by Alexandre Gazagnes
- Last update: 2024-02-01

1.1 About

1.1.1 Using Jupyter

You have 2 options: - Locally:

```
- **Install Anaconda https://www.anaconda.com/ or Jupyter https://jupyter.org/install on your m
```

- Online:
  - **Use Google Colab https://colab.research.google.com/** (you have to be connected to your google account)

1.1.2 Material

All the material for this course could be found here. - https://github.com/AlexandreGazagnes/CentraleSupElec-NLP-Public-Ressources

1.1.3 Python / Jupyter ?

Few Questions : - Why Python - Python vs R ? - What is Data Analysis ? - What are we talking about ? - What is Jupyter ?

1.1.4 Context

This notebook is about some basic core features of python programming language such as string, function, dataframes etc etc

1.1.5 Usefull Ressources about Google Colab

- On Youtube :
  - https://www.youtube.com/watch?v=8KeJZBZGtYo
  - https://www.youtube.com/watch?v=JJYZ3OE_lGo
  - https://www.youtube.com/watch?v=tCVXoTV12dE

1.1.6 Usefull Ressources about Anaconda and Jupyter

- On Youtube :
  - https://www.youtube.com/watch?v=ovlID7gefzE
  - https://www.youtube.com/watch?v=IMrxB8Mq5KU
  - https://www.youtube.com/watch?v=Ou-7G9VQugg
  - https://www.youtube.com/watch?v=5pf0_bpNbkw

1.1.7 Usefull Ressources about Git and GitHub

- On Youtube :
  - https://www.youtube.com/watch?v=RGOj5yH7evk
  - https://www.youtube.com/watch?v=3RjQznt-8kE&list=PL4cUxeGkcC9goXbgTDQ0n_4TBzOO0ocP

1.1.8 Teacher

- More info :
  - https://www.linkedin.com/in/alexandregazagnes/
  - https://github.com/AlexandreGazagnes

1.2 Preliminaries

1.2.1 System

These commands will display the system information:

Uncomment theses lines if needed.

[ ]:
```
# pwd
```

[ ]:
```
# cd ..
```

[ ]:
```
# ls
```

These commands will install the required packages:

Please note that if you are using google colab, all you need is already installed

[ ]:
```
# !pip install pandas matplotlib seaborn plotly scikit-learn
```

or copy the file requirements.txt and :

[ ]:
```
#! pip install -r requirements.txt
```

Try to use a virtual enviromement with venv, virtualenv or pipenv

[ ]:
```
#! python3 -m venv .venv # create the .venv folder
#! source .venv/bin/activate # activate the virtual env
#! pip install -r requirements.txt # install the requirements.txt
```

1.2.2 Imports

Import strings libraries (Built-In) :

[ ]:
```
import string

# import secrets
```

Import data libraries:

[ ]:
```
import pandas as pd  # DataFrame

# import numpy as np      # Matrix and advanced maths operations
```

Import Graphical libraries:

[ ]:
```
import matplotlib.pyplot as plt  # Visualisation
import seaborn as sns  # Visualisation

# import plotly.express as px   # Visualisation (not used here)
```

2 Import Ml Librairies

[ ]:
```
from sklearn.base import BaseEstimator, TransformerMixin  # Machine Learning
```

:warning:**These imports must be done, it is not possible to use this notebook without pandas, matplotlib etc.**

2.1 Basics of Python

2.1.1 Strings

A simple String :

[ ]:
```
text = "Hello World"
```

Output :

[ ]:
```
text
```

Print :

[ ]:
```
print(text)
```

Type of `text` :

[ ]:
```
type(text)
```

Specific string methods :

Lower :

[ ]:
```
text.lower()
```

Upper :

[ ]:
```
text.upper()
```

Strip :

[ ]:
```
text = "    Hello World     "
text
```

[ ]:
```
text.strip()
```

Is Alpha :

[ ]:
```
text.isalpha()
```

[ ]:
```
text = "hello"
text.isalpha()
```

Length :

[ ]:
```
len(text)
```

'o' is in text ?

[ ]:
```
"o" in text
```

Please use thins link if needed: * https://www.w3schools.com/python/python_strings.asp * https://www.geeksforgeeks.org/python-string/

Sort the text :

[ ]:
```
sorted(text)
```

Use a basic for loop for filtering :

[ ]:
```
new_txt = ""
for i in text:
    if i != "o":
        new_txt = new_txt + i
        # new_txt+=i
```

[ ]:
```
new_txt
```

Basic list comprehension :

[ ]:
```
[i for i in text]
```

Use list comprehension for filtering :

[ ]:
```
[i for i in text if "l" != i]
```

Transform `text` in list :

[ ]:
```
list(text)
```

Concatenate 2 strings :

[ ]:
```
txt1 = "hello"
txt2 = "world"

txt = txt1 + txt2
txt
```

Better :

[ ]:
```
txt = f"{txt1} {txt2}"
print(txt)
```

Split :

[ ]:
```
txt.split(" ")
```

Use the `\n` to add beak lines :

[ ]:
```
txt = "\n\n\n Hello World\n\n\n"
txt
```

With print :

[ ]:
```
print(txt)
```

Another option :

[ ]:
```
text = """

Hello
World
!


"""

text
```

With print :

[ ]:
```
print(text)
```

Indexing :

[ ]:
```
text = "hello world"
text[0]
```

[ ]:
```
text[2]
```

[ ]:
```
text[-1]
```

Slicing :

[ ]:
```
text[0:3]
```

[ ]:
```
text[:3]
```

[ ]:
```
text[2:4]
```

[ ]:
```
text[-2:]
```

Usefull builtin library :

[ ]:
```
string.ascii_letters
```

[ ]:
```
string.ascii_uppercase
```

[ ]:
```
punct = string.punctuation
punct
```

2.1.2 Functions

Let's create a simple function to clean our `text` variable :

[ ]:
```
text = """

Hello


World


!
"""
```

[ ]:
```
def clean(txt):
    """A very simple function"""

    txt = txt.lower()
    txt = txt.split()
    txt = [i.strip() for i in txt if i]

    return txt
```

[ ]:
```
cleaned_text = clean(text)
cleaned_text
```

We can add some optional arguments :

[ ]:
```python
def clean(txt, lower=True):
    """No so simple function"""

    if lower:
        txt = txt.lower()

    txt = txt.replace("\n", " ")

    txt = txt.split(" ")

    txt = [i.strip() for i in txt if i]

    return txt
```

[ ]:
```python
clean(text)
```

And build a much better function :

[ ]:
```python
def clean(
    txt,
    return_type: str,
    lower: bool = True,
    remove_punct: bool = True,
    remove_small_words: bool = True,
    small_word_n_char: int = 3,
):
    """More complex function"""

    # check if return_type is OK
    if not return_type in ["str", "list"]:
        raise AttributeError(
            f"return_type is not good : recieved {return_type}, expected in␣
 ↪['str', 'list]"
        )

    # if lower, apply the lower method
    if lower:
        txt = txt.lower()

    # remove breaklines
    txt = txt.replace("\n", " ")

    # if remove_punct, remove punctuation
    if remove_punct:
        for c in string.punctuation:
            txt = txt.replace(c, "")
```

```
    # split
    txt = txt.split(" ")

    # strip
    txt = [i.strip() for i in txt if i]

    # remove_small_words if needed
    if remove_small_words:
        txt = [i for i in txt if len(i) > small_word_n_char]

    # manage the return type
    if return_type == "list":
        return txt
    elif return_type == "str":
        return " ".join(txt)
    else:
        return -1
```

[ ]:
```
clean(text, return_type="list")
```

[ ]:
```
clean(text, return_type="str")
```

2.1.3 Transformers

You are not supposed to be familiar with custom transformers, but take 5 minutes to read this piece of code :

[ ]:
```
class StringCleaner(BaseEstimator, TransformerMixin):

    def __init__(
        self,
        return_type: str,
        lower: bool = True,
        remove_punct: bool = True,
        remove_small_words: bool = True,
        small_word_n_char: int = 3,
    ):

        # check if return_type is OK
        if not return_type in ["str", "list"]:
            raise AttributeError(
                f"return_type is not good : recieved {return_type}, expected in␣
↪['str', 'list]"
            )

        self.return_type = return_type
```

```
        self.lower = lower
        self.remove_punct = remove_punct
        self.remove_small_words = remove_small_words
        self.small_word_n_char = small_word_n_char

    def fit(self, txt, y=None):
        return self

    def transform(self, txt, y=None):

        # if lower, apply the lower method
        if self.lower:
            txt = txt.lower()

        # if remove_punct, remove punctuation
        if self.remove_punct:
            for c in string.punctuation:
                txt = txt.replace(c, "")

        # remove break lines
        txt = txt.replace("\n", " ")

        # split
        txt = txt.split(" ")

        # strip
        txt = [i.strip() for i in txt if i]

        # remove_small_words if needed
        if self.remove_small_words:
            txt = [i for i in txt if len(i) > self.small_word_n_char]

        # manage the return type
        if self.return_type == "list":
            return txt
        elif self.return_type == "str":
            return " ".join(txt)
        else:
            return -1
```

Let's create a `text` variable :

[ ]:
```
text = "\n\nHello my FRIeND !!! "
```

And let's use our custom transformer :

[ ]:
```
transformer = StringCleaner(return_type="list")
transformer.fit(text)
```

```
new_text = transformer.transform(text)
new_text
```

2.1.4 DataFrame

In order to create a dataframe, we can create a list of dictionnaries :

[ ]:
```
data = [
    {"_id": 0, "text": "My cat is Blue"},
    {"_id": 1, "text": "My cat is Red"},
    {"_id": 2, "text": "My cat is Dark. A very intense and beautiful dark "},
]

data
```

Then we can transform our dictionnaries in vectors inside a dataframe :

[ ]:
```
df = pd.DataFrame(data)

df
```

Type of df :

[ ]:
```
type(df)
```

Let's create a new column :

[ ]:
```
df["_len"] = df.text.apply(lambda i: len(i))

df
```

Selecting a column :

[ ]:
```
df.text
```

or :

[ ]:
```
df.loc[:, "text"]
```

or :

[ ]:
```
df.iloc[:, 1]
```

Selecting a row

[ ]:
```
df.iloc[0]
```

or :

[ ]:
```
df.loc[0, :]
```

Our df :

[ ]:
```
df
```

Selecting specific values in a dataframe :

[ ]:
```
df.loc[df._len > 15, :]
```

Describe numeric columns :

[ ]:
```
df.describe(include="number").round(2)
```

Describe non numeric columns :

[ ]:
```
df.describe(exclude="number").round(2)
```