Pandas Read Text File

Pandas offers several methods to read plain text (.txt) files and convert them to a Pandas DataFrame.

We can read text files in Pandas in the following ways:

- Using the `read_fwf()` function
- Using the `read_table()` function
- Using the `read_csv()` function

Using the above methods, let's read a sample text file named `data.txt` with the following content:

```python John 25 170 Alice 28 165 Bob 30 180

## Read Text Using read_fwf()

The acronym `fwf` in the `ead_fwf()` function in Pandas stands for fixed-width lines, and it is used to load DataFrames from files such as text files.

The text file should be separated into columns of fixed-width for it to be read using `read_fwf()`.

## Syntax

The syntax of `read_fwf()` in Pandas is:

```python pandas.read_fwf( filepath_or_buffer, colspecs = [], widths=None, infer_nrows=100, **kwds )

- **`filepath_or_buffer`**: specifies the file path or a file-like object from which the data will be read.
- **`colspecs`**: defines the column positions or ranges in the file.
- **`widths` (optional)**: an alternative to `colspecs` and can be used to define the width of each column in the file.
- **`infer_nrows` (optional)**: specifies the number of rows to be used for inferring the column widths if `widths` is not explicitly provided.
- **`**kwds` (optional)**: allows additional keyword arguments to be passed for further customization.

## Example: read_fwf()

The content of the `data.txt` file is the same as mentioned in the introduction section.

```python import pandas as pd
```

read the fixed-width file

df = pd.read_fwf('data.txt', colspecs=[(0, 5), (6, 10), (11, 15)], names = ['Name', 'Age', 'Height'])

print(df) Output:

```
Name  Age  Height
```

0 John 25 70 1 Alice 28 65 2 Bob 30 80

In the above example, we read a text file `'data.txt'` using `read_fwf()`.

Here,

- **`colspecs = [(0, 5), (6, 10), (11, 15)]`**: specifies the position of each column in the text file.
- **`names = ['Name', 'Age', 'Height']`**: specifies the names to be assigned to each column. The `names` argument is an example of a keyword argument in `**kwds`.

Read Text Using `read_table()`

The `read_table()` function in Pandas is used to read tabular data from a file or a URL. It is a convenient way to read data from delimited text files.

Syntax

The syntax of `read_table()` in Pandas is:

```python pandas.read_table( filepath_or_buffer, sep='\t', header='infer', names=None )

Here,

- **`filepath_or_buffer`**: specifies the path to the file to be read or a URL pointing to the file.
- **`sep`**: specifies the separator or delimiter used in the file to separate columns.
- **`header`**: specifies the row number (0-indexed) to be used as the column names.
- **`names`**: a list of column names for the DataFrame.

To learn more, please refer to the official documentation on `read_table()`.

## Example: `read_table()`

The content of the `data.txt` file is the same as mentioned in the introduction section.

```python import pandas as pd

# Load the text file into a DataFrame

df = pd.read_table('data.txt', sep=' \s+', header=None, names=['Name', 'Age', 'Height'])

print(df)

## Output

```python Name Age Height 0 John 25 170 1 Alice 28 165 2 Bob 30 180

Here, the `sep="\s+"` parameter is used to specify that the data is separated by one or more whitespace characters.

The separator is selected based on the format of the text file. For example, if the text file contained comma-separated values, we would have used `sep=','`.

## Read Text Using `read_csv()`

The `read_csv()` function in Pandas is used to read CSV files. It can also be used to read text files, as `read_csv()` allows the use of other separators like whitespace, tabs, semicolons, etc., in addition to commas.

### Syntax

The syntax of `read_csv()` in Pandas is given below:

```python pandas.read_csv( filepath_or_buffer, sep=',', header=0, names=['col1', 'col2', 'col3'], index_col='col1' )

Here,

- **`filepath_or_buffer`**: represents the path or buffer object containing the CSV data to be read.
- **`sep` (optional)**: specifies the delimiter used in the CSV file.
- **`header` (optional)**: indicates the row number to be used as the header or column names.
- **`names` (optional)**: a list of column names to assign to the DataFrame.
- **`index_col` (optional)**: specifies the column to be used as the index of the DataFrame.

These are some commonly used arguments of the `read_csv()` function. All of them are optional except `filepath_or_buffer`. There are many other optional arguments that can be used with `read_csv()`.

# Example: read_csv()

The content of the `data.txt` file is the same as mentioned in the introduction section. The values in the file are thus separated by whitespaces.

```python
import pandas as pd

# read the file using read_table()
df = pd.read_csv("data.txt", sep="\s+", header = None, names=['Name', 'Age', 'Height'])

print(df)
```

```
    Name  Age  Height
0   John   25     170
1  Alice   28     165
2    Bob   30     180
```

Here, `header = None` indicates that none of the rows in the text file is a header row.

Also, notice the use of `sep="\s+"` which indicates that the values in the files are separated by one or more spaces.

```
```