Dasar-Dasar Pemrograman 1

Tugas Pemrograman 03

Deadline: 23 November 2023 23:55
Waktu SCeLe

DATAFRAME & PENGOLAHAN DATA TABULAR

Pada Tugas Pemrograman kali ini, Anda diminta untuk mendefinisikan beberapa fungsi yang dapat membaca, memanipulasi, dan menganalisis data dalam bentuk **dataframe**. Sebagai pengenalan, **dataframe** adalah sebuah abstraksi dari sebuah data tabular yang direpresentasikan dalam **3-tuple**, yaitu sebuah **tuple** yang terdiri dari tiga informasi:

`(data, list nama kolom, list tipe data)`

"Data" merupakan *list of lists* yang menyimpan nilai-nilai pada tabel; **"*list* nama kolom"** adalah sebuah *list of strings* yang berisi *header* atau judul kolom pada tabel; **"*list* tipe data"** adalah *list of strings* yang berisi daftar nama tipe dari setiap kolom. Tipe dari sebuah kolom hanya ada satu dari tiga kemungkinan: **`"str"`**, **`"int"`**, dan **`"float"`**. Sebagai contoh, silakan Anda unduh dua buah file CSV (comma separated value): **test1.csv** dan **test2.csv.** Untuk **test1.csv**, bentuk dataframe-nya adalah:

Data:

```
[['budi', 19, 3.9],
 ['anto', 20, 3.6],
 ['rudi', 21, 3.85],
 ['rani', 19, 3.9],
 ['dewi', 21, 3.8]]
```

List nama kolom:

```
['nama', 'umur', 'ipk']
```

List tipe data:

```
['str', 'int', 'float']
```

Satu *cell* pada ***list of lists*** **Data** dapat bertipe **string (`"str"`)**, **integer (`"int"`)**, atau **float (`"float"`)**. Jika semua *cell* pada kolom tertentu berisi literal integer, maka ubah semuanya dalam tipe data integer; jika bukan integer, maka ada dua kemungkinan, yaitu float atau string; jika semua nilai pada suatu kolom berupa number dan ada minimal satu yang float, maka jadikan semua tipe data pada kolom tersebut sebagai float; jika *cell-cell* pada suatu kolom ada yang tidak bisa dikonversikan ke integer maupun float, maka tipe kolom tersebut adalah string. Sebagai contoh, untuk file **test2.csv**, kolom **ipk** mengandung nilai **"-"** yang berupa string atau non-number dan kolom **umur** mengandung **19.5** yang merupakan nilai bertipe float satu-satunya. Oleh karena itu, bentuk dataframe dari informasi tabular pada **test2.csv** adalah:

Data:

```
[['budi', 19.0, '3.9'],
 ['anto', 20.0, '3.6'],
 ['rudi', 21.0, '-'],
 ['rani', 19.5, '3.9'],
 ['dewi', 21.0, '3.8']]
```

List nama kolom:

```
['nama', 'umur', 'ipk']
```

List tipe data:

```
['str', 'float', 'str']
```

Silakan Anda bandingkan sekali lagi dataframe yang dihasilkan dari **test1.csv** dan **test2.csv**.

Selain file CSV **test1.csv** dan **test2.csv**, Anda juga diberikan sebuah dataset asli yang berukuran lebih besar yang bernama **abalone.csv**. Ini adalah sebuah dataset tabular yang terdiri dari 8 kolom dan berisi informasi terkait pengukuran hewan **abalone** yang ditemukan di laut. Detail informasi terkait dataset ini dapat dilihat pada link berikut:

https://archive.ics.uci.edu/dataset/1/abalone

Representasi dataframe dari dataset **abalone.csv** adalah:

Data (dipotong hanya top-5 baris saja):

```
[['M', 0.455, 0.365, 0.095, 0.514, 0.2245, 0.101, 0.15, 15],
 ['M', 0.35, 0.265, 0.09, 0.2255, 0.0995, 0.0485, 0.07, 7],
 ['F', 0.53, 0.42, 0.135, 0.677, 0.2565, 0.1415, 0.21, 9],
 ['M', 0.44, 0.365, 0.125, 0.516, 0.2155, 0.114, 0.155, 10],
 ['I', 0.33, 0.255, 0.08, 0.205, 0.0895, 0.0395, 0.055, 7],
 ...
 ]
```

List nama kolom:

```
['Sex', 'Length', 'Diameter', 'Height', 'Whole_weight',\
 'Shucked_weight', 'Viscera_weight', 'Shell_weight', 'Rings']
```

List tipe data:

```
['str', 'float', 'float', 'float', 'float', 'float', \
 'float', 'float', 'int']
```

Tugas Anda

Anda diberikan sebuah template kode yang perlu Anda lengkapi, yaitu file **tp3.py**. Di dalam file tersebut, terdapat **14 fungsi**. Dari 14 fungsi, 7 fungsi sudah didefinisikan dan siap digunakan oleh peserta. **Sedangkan 7 fungsi sisanya adalah tugas peserta untuk mendefinisikannya**. **Docstring** dari masing-masing fungsi sudah lengkap dan Anda bisa membaca langsung penjelasannya pada file **tp3.py** tersebut. Berikut adalah beberapa ringkasan penjelasan dari 14 fungsi tersebut:

1. **get_type(a_str) - sudah didefinisikan**

   Fungsi ini akan mengembalikan tipe dari literal string a_str, yang akan mengembalikan string “int”, “float” atau “str”. Beberapa contoh dari penggunaan fungsi ini:

   ```
   >>> get_type("0.5")
   "float"

   >>> get_type("5")
   "int"

   >>> get_type("5.a")
   ```

```
"str"
```

2. **read_csv(file_name, delimiter = ',') - perlu Anda definisikan**

   Fungsi ini bertugas untuk membaca sebuah file **comma separated value**, melakukan parsing, dan mengembalikan **dataframe yang berupa 3-tuple**. Salah satu tantangan yang cukup sulit saat mendefinisikan fungsi ini adalah ketika Anda perlu melakukan *inference* terkait tipe-tipe dari masing-masing kolom pada data tabular (apakah “int”, “float”, atau “str”). Ada dua asumsi terkait file CSV tersebut: [1] selalu ada header (nama kolom) pada baris pertama [2] nama kolom yang diberikan sudah dijamin unik:

```
>>> dataframe = read_csv("test1.csv")

>>> dataframe[0]
[['budi', 19, 3.9], ['anto', 20, 3.6], ['rudi', 21, 3.85],
['rani', 19, 3.9], ['dewi', 21, 3.8]]

>>> dataframe[1]
['nama', 'umur', 'ipk']

>>> dataframe[2]
['str', 'int', 'float']

>>> dataframe = read_csv("test2.csv")

>>> dataframe[0]
[['budi', 19.0, '3.9'], ['anto', 20.0, '3.6'], ['rudi', 21.0,
'-'], ['rani', 19.5, '3.9'], ['dewi', 21.0, '3.8']]

>>> dataframe[1]
['nama', 'umur', 'ipk']

>>> dataframe[2]
['str', 'float', 'str']
```

3. **to_list(dataframe) - sudah didefinisikan**

   Fungsi ini mengembalikan bagian *list of lists of items* atau tabel data pada dataframe (**elemen pertama pada 3-tuple dataframe**). Gunakan fungsi ini kedepannya jika ada keperluan untuk akses bagian data/tabel pada dataframe.

```
>>> dataframe = read_csv("test1.csv")
```

```
>>> to_list(dataframe)
[['budi', 19, 3.9], ['anto', 20, 3.6], ['rudi', 21, 3.85],
['rani', 19, 3.9], ['dewi', 21, 3.8]]
```

4. **get_column_names(dataframe) - sudah didefinisikan**

   Fungsi ini mengembalikan *list of column names* (**elemen kedua pada 3-tuple dataframe**). Gunakan fungsi ini kedepannya jika ada keperluan untuk akses daftar nama kolom pada sebuah dataframe.

```
>>> dataframe = read_csv("test1.csv")

>>> get_column_names(dataframe)
['nama', 'umur', 'ipk']

>>> dataframe = read_csv("abalone.csv")

>>> get_column_names(dataframe)
['Sex', 'Length', 'Diameter', 'Height', 'Whole_weight',
'Shucked_weight', 'Viscera_weight', 'Shell_weight', 'Rings']
```

5. **get_column_types(dataframe) - sudah didefinisikan**

   Fungsi ini mengembalikan *list of column types* (**elemen ketiga pada 3-tuple dataframe**). Gunakan fungsi ini kedepannya jika ada keperluan untuk akses daftar tipe kolom pada sebuah dataframe.

```
>>> dataframe = read_csv("test1.csv")

>>> get_column_types(dataframe)
['str', 'int', 'float']

>>> dataframe = read_csv("abalone.csv")

>>> get_column_types(dataframe)
['str', 'float', 'float', 'float', 'float', 'float', 'float',
'float', 'int']
```

6. **head(dataframe, a_int) - sudah didefinisikan**

   Fungsi ini mengembalikan **string** yang merupakan **representasi tabel** (a_int sebagai jumlah baris pertama yang di-*return*) dengan format:

```
 kolom_1|     kolom_2|     kolom_3|     ...
-----------------------------------------
value_11|    value_12|    value_13|     ...
value_21|    value_22|    value_23|     ...
  ...          ...          ...
```

Space setiap kolom dibatasi hanya 15 karakter dan right-justified. **Jangan pakai print()! tetapi *return* string!**

Contoh:

```
>>> dataframe = read_csv("test1.csv")

>>> head(dataframe, top_n = 3)
           nama|            umur|             ipk
-----------------------------------------------
           budi|              19|             3.9
           anto|              20|             3.6
           rudi|              21|            3.85

>>> head(dataframe)
           nama|            umur|             ipk
-----------------------------------------------
           budi|              19|             3.9
           anto|              20|             3.6
           rudi|              21|            3.85
           rani|              19|             3.9
           dewi|              21|             3.8
```

7. **info(dataframe) - perlu Anda definisikan**

   Fungsi ini mengembalikan **string** yang merupakan **representasi tabel** dan **jumlah baris** dengan format:

```
Total Baris = xxxxx baris

Kolom          Tipe
-----------------------------
kolom_1        tipe_1
kolom_2        tipe_2
 ...
```

Space setiap kolom dibatasi hanya 15 karakter dan right-justified. **Jangan pakai print()! tetapi *return* string!**

Contoh:

```
>>> dataframe = read_csv("abalone.csv")

>>> info(dataframe)
Total Baris = 4177 baris

Kolom            Tipe
------------------------------
Sex              str
Length           float
Diameter         float
Height           float
Whole_weight     float
Shucked_weight   float
Viscera_weight   float
Shell_weight     float
Rings            int
```

8. **satisfy_cond(value1, condition, value2) - sudah didefinisikan**

   Fungsi ini mengembalikan **boolean** dari perbandingan antara value1 dan value2. Perbandingan akan dilakukan berdasarkan **condition** yang diberikan. Gunakan fungsi ini untuk membandingkan dua nilai dengan kondisi yang diinginkan. Fungsi ini digunakan atau dipanggil pada fungsi **select_rows()** di nomor 9.

```
>>> value1 = 5
>>> condition = "<"
>>> value2 = 10
>>> result = satisfy_cond(value1, condition, value2)
>>> result
True
```

9. **select_rows(dataframe, col_name, condition, value) - perlu Anda definisikan**

   Fungsi ini mengembalikan **dataframe** baru dengan baris-baris yang dipilih hanya yang nilai **col_name** memenuhi '**condition**' terkait '**value**' tertentu. Fungsi ini perlu memanggil fungsi **satisfy_cond(value1, condition, value2)** yang sudah didefinisikan sebelumnya!

   Contoh: pada data yang ada di **test1.csv**, kita ingin membuat **dataframe** yang hanya berisi mahasiswa berumur **< 21 tahun**:

```
>>> dataframe = read_csv("test1.csv")
```

```
>>> head(dataframe)
           nama|            umur|             ipk
-----------------------------------------------
           budi|              19|             3.9
           anto|              20|             3.6
           rudi|              21|            3.85
           rani|              19|             3.9
           dewi|              21|             3.8

>>> new_dataframe = select_rows(dataframe, "umur", "<", 21)

>>> head(new_dataframe)
           nama|            umur|             ipk
-----------------------------------------------
           budi|              19|             3.9
           anto|              20|             3.6
           rani|              19|             3.9
```

10. select_cols(dataframe, selected_cols) - perlu Anda definisikan

Fungsi ini mengembalikan dataframe baru dengan kolom-kolom yang dipilih hanya yang terdapat pada '**selected_cols**' saja.

Contoh: pada data yang ada di **abalone.csv**, kita ingin membuat **dataframe** baru yang hanya terdiri dari kolom **Length**, **Diameter**, dan **Rings**:

```
>>> dataframe = read_csv("abalone.csv")

>>> new_dataframe = select_cols(dataframe, \
                                ["Length", "Diameter", "Rings"])

>>> info(new_dataframe)
Total Baris = 4177 baris

Kolom           Tipe
------------------------------
Length          float
Diameter        float
Rings           int

>>> head(new_dataframe, top_n = 5)
           Length|        Diameter|           Rings
-----------------------------------------------
            0.455|           0.365|              15
             0.35|           0.265|               7
             0.53|            0.42|               9
             0.44|           0.365|              10
             0.33|           0.255|               7
```

11. count(dataframe, col_name) - perlu Anda definisikan

Fungsi ini mengembalikan **dictionary** yang berisi **frequency count** dari setiap nilai unik pada kolom **col_name**. Perhatikan bahwa tipe nilai pada col_name **harus string**! Fungsi ini dapat menggunakan fungsi lainnya untuk mempermudah pencarian.

```
>>> dataframe = read_csv("abalone.csv")

>>> count(dataframe, "Sex")
{'M': 1528, 'F': 1307, 'I': 1342}
```

12. mean_col(dataframe, col_name) - perlu Anda definisikan

Fungsi ini mengembalikan **float** berupa rata-rata nilai pada kolom '**col_name**' di dataframe.

```
>>> dataframe = read_csv("abalone.csv")

>>> mean_col(dataframe, "Diameter")
0.407881254488869

>>> mean_col(dataframe, "Height")
0.1395163993296614
```

13. show_scatter_plot(x, y, x_label, y_label) - sudah didefinisikan

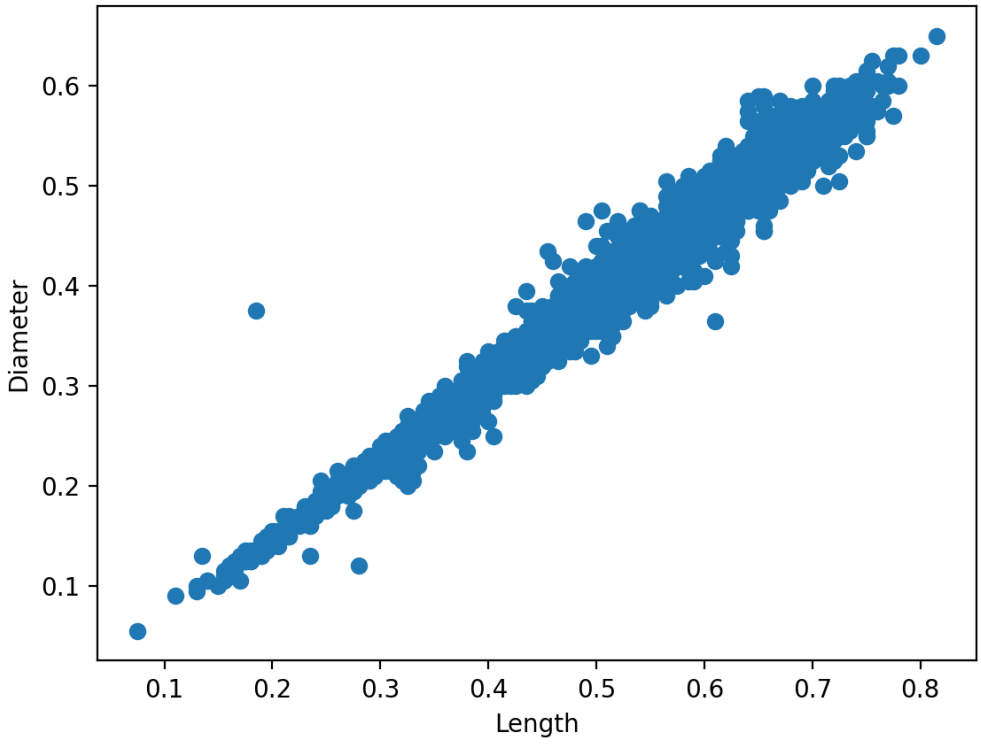
Fungsi ini bertujuan untuk menampilkan ***scatter plot*** dari parameter **x** (*list of numbers*) dan **y** *(list of numbers)*. ***Scatter plot*** adalah jenis diagram yang menggunakan titik-titik untuk menunjukkan hubungan antara dua variabel. Berikut link untuk penjelasan lebih mengenai *scatter plot*. Dalam menampilkan *scatter plot*, python memiliki *library* ***matplotlib*** yang membantu dalam menampilkan *scatter plot*. Berikut adalah link untuk instalasi *matplotlib* dan *library* yang berisi fungsi - fungsi untuk menggunakan *matplotlib*. Perhatikan bahwa ***matplotlib*** **perlu di-*import*** walaupun sudah diunduh. **Mahasiswa dilarang untuk mengunduh library selain *matplotlib* untuk menampilkan *scatterplot.***

Fungsi ini **tidak mengembalikan nilai** apapun, namun secara otomatis menampilkan *scatter plot* dari parameter yang diberikan.

14. scatter(dataframe, col_name_x, col_name_y) - perlu Anda definisikan

Fungsi ini akan menampilkan *scatterplot* antara kolom **col_name_x** dan **col_name_y** pada dataframe. Anda wajib panggil fungsi **show_scatter_plot(x, y, x_label, y_label)** untuk menampilkan gambar *scatterplot*. Jadi, yang dikerjakan pada pada fungsi ini sejatinya adalah mengumpulkan nilai-nilai pada kolom **col_name_x** dan **col_name_y** dalam struktur data list, dan kemudian di-pass ke fungsi **show_scatter_plot(...)**. Berikut adalah contoh menampilkan *scatterplot* antara kolom "Length" dan "Diameter" pada dataset **abalone.csv**.

```
>>> dataframe = read_csv("abalone.csv")
>>> show_scatter(dataframe, "Length", "Diameter")
```



Contoh Skenario Penggunaan

Anda tidak perlu membuat program utama yang berisi menu. Anda cukup uji fungsi-fungsi yang Anda definisikan dengan beberapa test case berikut, dan apakah fungsi yang Anda definisikan sudah sesuai dengan ekspektasi.

memuat dataframe dari tabel pada file abalone.csv

```
>>> dataframe = read_csv("abalone.csv")
```

cetak 10 baris pertama

```
>>> head(dataframe, top_n = 10)
```

cetak informasi dataframe

```
>>> info(dataframe)
```

kembalikan dataframe baru, dengan kolom Length > 0.49

```
>>> new_dataframe = select_rows(dataframe, "Length", ">", 0.49)
>>> head(new_dataframe, top_n = 5)
```

kembalikan dataframe baru, dimana Sex == "M" DAN Length > 0.49

```
>>> new_dataframe = select_rows(select_rows(dataframe, "Length",\
                                            ">", 0.49), \
                                "Sex", "==", "M")
>>> head(new_dataframe, top_n = 5)
```

kembalikan dataframe baru yang hanya terdiri dari kolom Sex, Length, Diameter, dan Rings

```
>>> new_dataframe = select_cols(dataframe, ["Sex", "Length",\
                                            "Diameter", "Rings"])
>>> head(new_dataframe, top_n = 5)
```

hitung mean pada kolom Length (pada **dataframe original**)

```
>>> mean_col(dataframe, "Length"))
```

melihat unique values pada kolom Sex, dan frekuensi kemunculannya (pada **dataframe original**)

```
>>> count(dataframe, "Sex")
```

tampilkan scatter plot antara kolom "Height" dan "Diameter"

```
>>> scatter(dataframe, "Height", "Diameter")
```

Catatan:

- Anda tidak boleh menggunakan library lain untuk menyelesaikan permasalahan ini, seperti library **pandas** (https://pandas.pydata.org/). Library eksternal yang diperbolehkan adalah **matplotlib** untuk membuat gambar scatter plot.
- **Anda boleh membuat asumsi untuk hal-hal yang tidak secara eksplisit ada di deskripsi soal**.

Bonus (10 poin)

Anda akan mendapatkan nilai bonus **hingga** 10 poin jika mencoba mendefinisikan fungsi-fungsi baru untuk manipulasi tabel, seperti melakukan filtering yang lebih kompleks dan sebagainya. Anda dapat melihat dokumentasi **dataframe** dari library **pandas** untuk mendapatkan inspirasi:
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

Rubrik Penilaian

Penilaian akan mengacu pada Rubrik Penilaian TP DDP1

Plagiarisme dan keterlambatan pengumpulan tidak akan ditoleransi. Anda diperbolehkan berdiskusi dengan teman terkait ide implementasi tugas ini. Harap menuliskan kolaborator jika berdiskusi dengan mahasiswa lain. Perlu diingat bahwa **implementasi kode dilakukan secara individu**. Tim pengajar akan melakukan *code similarity checking* pada implementasi kode mahasiswa, dan jika terbukti melakukan kecurangan/plagiarisme akan mendapat **sanksi berupa nilai 0** pada TP ini.

Pengumpulan

Berkas yang perlu dikumpulkan adalah file tp3.py yang di-zip dengan format penamaan sebagai berikut:

- [Kelas]_[KodeAsdos]_[NPM]_[NamaLengkap]_TP03.zip

Contoh:

A_XYZ_2306123456_NamaSaya_TP03.zip

Good luck and happy coding ^^