# Python for Data Analysts

Team Algoritma

December 28, 2023

**Coursebook: Python for Data Analysts**

- Bagian 1 Audit Analytics untuk Bank Rakyat Indonesia
- Durasi: 7 Jam
- *Last Updated*: December 2023

---

- *Coursebook* ini disusun dan dikurasi oleh tim produk dan instruktur dari Algoritma Data Science School

## 1 Background

*Coursebook* ini merupakan bagian dari **BRI Audit Analytics** yang disiapkan oleh Algoritma. *Coursebook* ini ditujukan hanya untuk khalayak terbatas, yaitu individu dan organisasi yang menerima *coursebook* ini langsung dari organisasi pelatihan. Tidak boleh direproduksi, didistribusikan, diterjemahkan, atau diadaptasi dalam bentuk apapun di luar individu dan organisasi ini tanpa izin.

Algoritma adalah pusat pendidikan *data science* yang berbasis di Jakarta. Kami menyelenggarakan *workshop* dan program pelatihan untuk membantu para profesional dan mahasiswa dalam menguasai berbagai sub-bidang *data science* yaitu: *data visualization, machine learning,* statistik, dan lain sebagainya.

## 2 Training Objectives

- Introduction to the `pandas` library.
- Introduction to `DataFrame`

- Data Types
- Exploratory Data Analysis I
- Indexing and Subsetting

## 3 Python for Data Analysts

Python adalah *high-level programming language*, yang memudahkan manusia untuk memprogram sesuatu dengan menggunakan bahasa yang mudah dipahami. Selain itu, Python juga menyediakan file notebook yang bisa digunakan untuk melakukan analisis . Python memiliki notebook dokumen, yang memudahkan pengguna dalam melakukan analisis.

Notebook dalam Python adalah file yang mengandung **baris kode** (Python) dan **elemen text** (paragraf, heading, gambar). File notebook biasa digunakan dalam proses analisis karena selain bisa digunakan untuk menulis kode juga bisa digunakan untuk menulis analisa deskriptif. Ekstensi file notebook dalam Python adalah .ipynb.

Selanjutnya kita akan mengenal tentang dasar - dasar dalam pemrograman bahasa Python.

### 3.1 Variables and Keywords

Bahasa Python merupakan bahasa pemrograman yang sangat mudah dipahami. *Syntax* yang digunakan merupakan syntax umum yang sering kali ditemui di dunia nyata. Seperti *assignment* (=).

Kita akan sering menggunakan operator *assignment* untuk membuat sebuah variabel baru. Variabel adalah nama yang merujuk kepada suatu nilai.

[ ]:
```
perusahaan = 'Bank Rakyat Indonesia'

print(perusahaan)
```

```
Bank Rakyat Indonesia
```

Hal yang perlu diingat adalah Python merupakan bahasa pemrograman **case-sensitive**, sehingga perbedaan karakter akan merujuk pada hal yang berbeda. Seperti `perusahaan` dan `Perusahaan` adalah dua hal yang berbeda.

[ ]:
```
'perusahaan' == 'Perusahaan'
```

[ ]:
```
False
```

Berdasarkan sifat Python tersebut, penamaan variabel merupakan hal yang penting untuk diperhatikan. Sebagaimana Python yang bersifat *human-readable*. Terdapat beberapa kata yang tidak dapat digunakan sebagai nama variabel. Kata - kata tersebut adalah kata yang termasuk kedalam **Python Keywords**.

Saat ini terdapat 33 *Keywords*: `True`, `False`, `None`, `and`, `as`, `assert`, `break`, `class`, `continue`, `def`, `del`, `elif`, `else` `except`, `finally`, `for`, `from`, `global`, `if`, `import`, `in`, `is`, `lambda`, `nonlocal`, `not`, `or`, `pass` `raise`, `return`, `try`, `while`, `with`, `yield`.

Selain penggunaan *keyword*. terdapat beberapa aturan penamaan variable, diantaranya:

1. Tidak boleh diawali dengan angka.
2. Tidak menggunakan special karakter kecuali underscore ( _ ).

## 4 Introduction to Pandas Library

`pandas` adalah *library* yang digunakan sebagai *tools* analisis data dan struktur data pada Python. Pandas memiliki kemampuan pengolahan data seperti agregasi, manipulasi data, join dan lain sebagainya. Pandas mengolah data dalam bentuk objek `Dataframe`, sehingga memungkinkan pandas mengolah data dari berbagai format file.

pandas official documentation

## 4.1 Working with DataFrame

Untuk menggunakan `pandas`, kita akan menggunakan fungsi `import`. Ketika kita sudah melakukan *import* library `pandas`, maka kita bisa menggunakan seluruh fungsi dari pandas.

syntax: *pandas.function_name*

[ ]:
```
import pandas as pd
pd.__version__
```

[ ]:
```
'2.0.0'
```

Kita akan melakukan pembacaan data. Data yang digunakan adalah data `turnover.csv`

[ ]:
```
rice = pd.read_csv("data_input/rice.csv", index_col=0)
rice.head()
```

[ ]:
```
   receipt_id  receipts_item_id    purchase_time category sub_category
1     9622257          32369294  7/22/2018 21:19     Rice         Rice  \
2     9446359          31885876  7/15/2018 16:17     Rice         Rice
3     9470290          31930241  7/15/2018 12:12     Rice         Rice
4     9643416          32418582   7/24/2018 8:27     Rice         Rice
5     9692093          32561236  7/26/2018 11:28     Rice         Rice

        format  unit_price  discount  quantity yearmonth
1  supermarket    128000.0         0         1   2018-07
2   minimarket    102750.0         0         1   2018-07
3  supermarket     64000.0         0         3   2018-07
4   minimarket     65000.0         0         1   2018-07
5  supermarket    124500.0         0         1   2018-07
```

Pada code diatas, kita memanfaatkan fungsi `.read_csv()` dari *library* `pandas` yang digunakan untuk membuka file csv dari path yang diberikan dan menyimpannya dalam sebuah objek dataframe bernama `rice`.

Selanjutnya mari kita amati dataframe kita. Secara *default*, pandas akan mengenali baris pertama dari data kita sebagai *header* dari tabel. Tetapi apabila kita tidak mengingikan, kita bisa menambahkan parameter `header = None` pada fungsi awal.

Selain itu coba amati bahwa kita menggunakan `0` untuk menunjukkan baris pertama dari data kita. Hal ini dikarenakan Python menggunakan `0-based indexing`, sehingga index pertama akan dimulai dari angka 0

## 4.2 Data Types

Dataframe dalam pandas akan memiliki beberapa *series*. **Series** dalam dataframe mengacu kepada kolom. Secara *default*, `pandas` akan secara otomatis menentukan tipe data dari masing - masing kolom. Tetapi, itu tidak selalu benar. Sehingga kita perlu melakukan pengecekan maupun perubahan sehingga tipe data masing - masing kolom sesuai.

Pada bagian ini kita akan mempelajari bagaimana mengecek tipe data dari sebuah dataframe, Jenis tipe data, dan mengubah kolom menjadi tipe data yang sesuai

[ ]:
```
print(rice.dtypes)
```

```
receipt_id            int64
receipts_item_id      int64
purchase_time        object
category             object
sub_category         object
format               object
unit_price          float64
discount              int64
quantity              int64
yearmonth            object
dtype: object
```

`dtypes` merupakan **atribut** dari objek dataframe yang menunjukkan jenis tipe data. Karena `rice` adalah objek pandas (dataframe), `dtypes` akan menunjukkan jenis tipe data untuk setiap kolomnya.

**Knowledge check: `.dtypes` and pandas attributes** Lihat kode di bawah ini - output apa yang akan muncul dari kode tersebut? Kenapa?

```
x = [2019, 4, 'data science']
x.dtypes
```

Hint: Coba ketik `type(x)` dan cek tipe data `x`.

[ ]:
```
## Your code below


## -- Solution code
```

Coba lihat beberapa contoh `DataFrame.dtypes`:

[ ]:
```
employees = pd.DataFrame({
    'name': ['Anita', 'Brian'],
    'age': [34, 29],
    'joined': [pd.Timestamp('20190410'), pd.Timestamp('20171128')],
    'degree': [True, False],
    'hourlyrate': [35.5, 29],
    'division': ['HR', 'Product']
})
employees.dtypes
```

[ ]:
```
name                 object
age                   int64
joined        datetime64[ns]
degree                  bool
hourlyrate           float64
division              object
dtype: object
```

[ ]:
```
employees
```

[ ]:
```
    name  age     joined  degree  hourlyrate division
0  Anita   34 2019-04-10    True        35.5       HR
1  Brian   29 2017-11-28   False        29.0  Product
```

Selanjutnya kita akan melihat tipe data dan isi dari seluruh kolom pada dataframe di atas:

- `name` [`object`]: value berupa text
- `age` [`int`]: value berupa integer
- `joined` [`datetime`]: value berupa tanggal dan waktu
- `degree` [`bool`]: value berupa True/False
- `hourlyrate` [`float`]: value berupa angka floating point
- `division` [`object`]: value berupa text

Di antara kolom-kolom ini, hanya `age` dan `hourlyrate` yang merupakan kolom dengan angka numerik. Salah satu tipe data yang menarik untuk dibahas selanjutnya adalah **categorical values**.

### 4.2.1 Categorical and Numerical Variable

Tipe data **numeric** adalah tipe data yang berupa angka, tipe data ini meliputi tipe data **Integer** dan **Float**. Integer menyimpan nilai berupa angka bulat tanpa koma, sedangkan Float menyimpan nilai angka desimal.

Tipe data **categoric** adalah tipe data yang menyimpan nilai kategorik. Nilai kategorik merupakan nilai yang dapat dikelompokkan menjadi beberapa kategori dan biasanya berulang di data kita. Contoh nilai kategorik adalah jenis kelamin, golongan darah, rating.

Alasan mengapa kita perlu menggunakan tipe data categorical:

1. ***Dari sisi "Business Perspective"***, hal ini dapat menginformasikan dan memandu seorang *Analyst* pada pertanyaan seperti metode statistik atau tipe plot mana yang digunakan untuk mengolah data.
2. ***Dari sisi teknikal***, ketika kita bekerja dengan tipe data categorical pada pandas, hal ini akan jauh menghemat memori dan menambah kecepatan komputasional.

Dari tipe data kolom pada dataframe `employees` kita akan menganalisis apa saja kolom yang seharusnya memiliki tipe data kategorik. Selanjutnya kita akan mengubah tipe data tersebut dengan menggunakan fungsi `astype()`.

Hal yang harus diperhatikan ketika menggunakan fungsi `astype()` adalah kita harus melakukan assignment ulang pada kolom yang dilakukan perubahan.

Contoh:

```
# convert marital_status to category
employees['marital_status'] = employees['marital_status'].astype('category')

# convert experience to integer
employees['experience'] = employees['experience'].astype('int')
```

Kita akan melakukan perubahan terhadap kolom channel, packet_name, provinsi dan region.

[ ]:
```
## Your code below
```

Selanjutnya kita akan cek kembali apakah tipe data kolom tersebut sudah berubah menggunakan `dtypes`

[ ]:
```
## Your code below
```

### 4.3 Exploratory Data Analysis Tools

**Exploratory Data Analysis** (EDA) adalah proses kita melakukan eksplorasi awal terhadap data, untuk mengetahui lebih lanjut kondisi dan karakteristik data.

`pandas` memiliki built-in function untuk melakukan *Exploratory Data Analysis*. Berikut adalah fungsi yang umum digunakan untuk proses EDA.

- `.head()` dan `.tail()`
- `.shape` dan `.size`
- `.axes`
- `.dtypes`
- `.info()`
- `.describe()`

[ ]:
```
rice.head()
```

[ ]:
```
   receipt_id  receipts_item_id    purchase_time category sub_category
1     9622257          32369294  7/22/2018 21:19     Rice         Rice  \
2     9446359          31885876  7/15/2018 16:17     Rice         Rice
3     9470290          31930241  7/15/2018 12:12     Rice         Rice
4     9643416          32418582   7/24/2018 8:27     Rice         Rice
5     9692093          32561236  7/26/2018 11:28     Rice         Rice

         format  unit_price  discount  quantity yearmonth
1  supermarket    128000.0         0         1   2018-07
2   minimarket    102750.0         0         1   2018-07
3  supermarket     64000.0         0         3   2018-07
4   minimarket     65000.0         0         1   2018-07
5  supermarket    124500.0         0         1   2018-07
```

Fungsi `.head()` akan menampilkan 5 data pertama dari dataframe. Secara *default* fungsi ini akan menampilkan 5 data pertama. Tetapi apabila ingin mengubah banyak data yang ingin ditampilkan bisa ditambahkan sebuah nilai seperti `.head(3)`. Ini akan menampilkan 3 data pertama dalam daraframe `rice`.

Fungsi `.tail()` merupakan kebalikan dari `.head()`. Fungsi ini akan menampilkan 5 data terakhir.

Setelah mengetahui beberapa baris awal dari data kita. Seringkali muncul pertanyaan, berapa ukuran data kita. Untuk mengetahui hal tersebut kita akan melihat dari atribut `.shape` dan `.size` dari pandas dataframe.

Tidak seperti `.head()` dan `.tail()`, `.shape` dan `.size` merupakan atribut dari pandas dataframe. Perbedaan mendasar dari fungsi dan atribut adalah apabila fungsi selalu diakhiri dengan tanda

kurung. Sehingga Anda harus lebih berhati - hati dalam mengenali fungsi dan atribut.

[ ]:
```
rice.shape
```

[ ]:
```
(12000, 10)
```

Atribut `.shape` menunjukkan berapa banyak baris dan kolom yang ada pada dataframe kita. *Output* dari atribut ini adalah `(baris, kolom)`. Sehingga dari dataframe `rice` kita memiliki 12000 baris data dengan 10 kolom.

[ ]:
```
rice.size
```

[ ]:
```
120000
```

Berbeda dengan `.shape`, atribut `.size` menunjukkan banyak elemen yang ada dalam dataframe. Karena pada dataframe `rice` memiliki 12000 baris dengan 10 kolom, maka banyak elemen dari dataframe `rice` adalah `baris x kolom` yaitu 120000.

Kedua atribut tersebut hanya akan menampilkan informasi tentang banyak baris, kolom dan elemen. Tetapi tidak dapat menunjukkan informasi nama baris atau kolom. Untuk itu, kita bisa menggunakan atribut `.axes`. Atribut `.axes` akan mengembalikan nilai sebuah list dari dua elemen yaitu sumbu baris dan sumbu kolom.

Secara default, elemen **pertama** dari fungsi `.axes` menunjukkan sumbu baris (nama/index) dan elemen **kedua** menunjukkan sumbu kolom. Untuk mengambil informasi tersebut secara terpisah, Anda bisa menggunakan operator `[]` dan memasukkan urutan dari list. Ingat bahwa python menggunakan **0-based indexing**.

[ ]:
```
rice.axes
```

[ ]:
```
[Index([    1,     2,     3,     4,     5,     6,     7,     8,     9,    10,
        …
        11991, 11992, 11993, 11994, 11995, 11996, 11997, 11998, 11999, 12000],
       dtype='int64', length=12000),
 Index(['receipt_id', 'receipts_item_id', 'purchase_time', 'category',
        'sub_category', 'format', 'unit_price', 'discount', 'quantity',
        'yearmonth'],
       dtype='object')]
```

Bagaimana jika ingin mengambil elemen dari sumbu baris baris atau kolom saja ?

[ ]:
```
# Your code
```

Penggunaan atribut `.axes` untuk mengambil elemen dari sumbu kolom akan memberikan hasil berupa nama - nama kolom. Hal ini sama dengan penggunaan atribut `.columns`.

Apabila Anda ingin mendapatkan informasi lengkap terkait nama kolom, banyak baris serta tipe data untuk tiap baris pandas menyediakan fungsi `.info()` yang dapat digunakan. Jika Anda melakukan proses pengubahan tipe data dengan baik pada bagian sebelumnya, maka output yang Anda dapatkan akan seperti ini.

[ ]:
```
rice.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 12000 entries, 1 to 12000
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   receipt_id        12000 non-null  int64
 1   receipts_item_id  12000 non-null  int64
 2   purchase_time     12000 non-null  object
 3   category          12000 non-null  object
 4   sub_category      12000 non-null  object
 5   format            12000 non-null  object
 6   unit_price        12000 non-null  float64
 7   discount          12000 non-null  int64
 8   quantity          12000 non-null  int64
 9   yearmonth         12000 non-null  object
dtypes: float64(1), int64(4), object(5)
memory usage: 1.0+ MB
```

Fungsi `.info()` akan memberikan informasi terkait kondisi data kita. Informasi yang diberikan meliputi banyak baris dalam data, tipe data setiap kolom dan nilai kosong / *missing value* setiap kolom.

Fungsi `.info()` hanya menampilkan informasi tentang dataframe saja. Untuk melakukan *Exploratory Data Analysis* sering kali membutuhkan perhitungan statistika deskriptif untuk menggambarkan kondisi dari data kita. Fungsi yang digunakan adalah `.describe()`

[ ]:
```
rice.describe()
```

[ ]:
```
         receipt_id  receipts_item_id     unit_price       discount  \
count  1.200000e+04      1.200000e+04   12000.000000   12000.000000
mean   7.650135e+06      2.457950e+07   70013.146313     835.305750
std    1.873838e+06      7.171105e+06   29905.391437    6207.475704
min    3.173994e+06      9.282023e+06    9395.000000   -4100.000000
25%    5.983209e+06      1.820694e+07   61900.000000       0.000000
50%    7.443618e+06      2.234337e+07   63500.000000       0.000000
75%    9.149786e+06      3.108379e+07   66000.000000       0.000000
max    1.146321e+07      3.842939e+07  219400.000000  320000.000000

           quantity
count  12000.000000
mean       1.332917
std        0.980304
min        1.000000
25%        1.000000
50%        1.000000
75%        1.000000
max       19.000000
```

Fungsi `.describe()` akan menampilkan informasi statistika deskriptif pada data kita. Secara default, fungsi `.describe()` hanya menampilkan statistika deskriptif untuk tipe data **numerik**. Informasi yang ditampilkan oleh fungsi `.describe()` meliputi:

- Count
- Mean
- Standard Deviation
- Minimum Value
- 25th Percentile (Q1)
- 50th Percentile (Q2/Median)
- 75th Percentile (Q3)
- Maximum Value

Untuk menampilkan statistika deskriptif pada kolom dengan tipe data categorik bisa menambahkan parameter `include` kedalam fungsi `.describe()`. Parameter ini bisa diisi **tipe data** yang ingin di tampilkan atau `all` jika untuk seluruh kolom

[ ]:
```
rice.describe(include='all')
```

[ ]:

| | receipt_id | receipts_item_id | purchase_time | category | sub_category | \ |
|---|---|---|---|---|---|---|
| count | 1.200000e+04 | 1.200000e+04 | 12000 | 12000 | 12000 | |
| unique | NaN | NaN | 11609 | 1 | 1 | |
| top | NaN | NaN | 6/14/2018 0:00 | Rice | Rice | |
| freq | NaN | NaN | 4 | 12000 | 12000 | |
| mean | 7.650135e+06 | 2.457950e+07 | NaN | NaN | NaN | |
| std | 1.873838e+06 | 7.171105e+06 | NaN | NaN | NaN | |
| min | 3.173994e+06 | 9.282023e+06 | NaN | NaN | NaN | |
| 25% | 5.983209e+06 | 1.820694e+07 | NaN | NaN | NaN | |
| 50% | 7.443618e+06 | 2.234337e+07 | NaN | NaN | NaN | |
| 75% | 9.149786e+06 | 3.108379e+07 | NaN | NaN | NaN | |
| max | 1.146321e+07 | 3.842939e+07 | NaN | NaN | NaN | |

| | format | unit_price | discount | quantity | yearmonth |
|---|---|---|---|---|---|
| count | 12000 | 12000.000000 | 12000.000000 | 12000.000000 | 12000 |
| unique | 3 | NaN | NaN | NaN | 12 |
| top | minimarket | NaN | NaN | NaN | 2018-07 |
| freq | 7088 | NaN | NaN | NaN | 1000 |
| mean | NaN | 70013.146313 | 835.305750 | 1.332917 | NaN |
| std | NaN | 29905.391437 | 6207.475704 | 0.980304 | NaN |
| min | NaN | 9395.000000 | -4100.000000 | 1.000000 | NaN |
| 25% | NaN | 61900.000000 | 0.000000 | 1.000000 | NaN |
| 50% | NaN | 63500.000000 | 0.000000 | 1.000000 | NaN |
| 75% | NaN | 66000.000000 | 0.000000 | 1.000000 | NaN |
| max | NaN | 219400.000000 | 320000.000000 | 19.000000 | NaN |

Coba gunakan fungsi `.describe()` untuk menampilkan statistika deskriptif untuk kolom - kolom yang bertipe `category`

[ ]:
```
# Your code below
```

Kita sudah mempelajari `.dtypes` di subbab sebelumnya. Mari kita latihan dengan menggunakan data `rice`. Apakah kolom-kolom yang ada di Dataframe tersebut sudah benar? Jika belum, lis kolom apa saja yang harus diubah dan perubahan tipe datanya!

[ ]:
```
rice.dtypes
```

[ ]:
```
receipt_id            int64
receipts_item_id      int64
purchase_time        object
category             object
sub_category         object
format               object
unit_price          float64
discount              int64
quantity              int64
yearmonth            object
dtype: object
```

Bandingkan lis yang sudah Anda buat dengan kode di bawah ini. Kita ubah tipe data kolom `purchase_time` menjadi `datetime` dan mengubah tipe data kategorikal juga:

[ ]:
```
rice['purchase_time'] = rice['purchase_time'].astype('datetime64[ns]')
rice[['category', 'sub_category', 'format']] = rice[['category',␣
↪'sub_category', 'format']].astype('category')
rice.dtypes
```

[ ]:
```
receipt_id                   int64
receipts_item_id             int64
purchase_time       datetime64[ns]
category                  category
sub_category              category
format                    category
unit_price                 float64
discount                     int64
quantity                     int64
yearmonth                   object
dtype: object
```

**Knowledge Check: Exploratory Data Analysis** Misalkan kita memiliki dataframe pandas bernama `inventory`

1. Code `inventory.dtypes` dan mendapatkan keluaran berikut. Kolom manakah yang mungkin memerlukan konversi tipe karena sepertinya tipe datanya salah? Pilih semua yang sesuai.

   ☐ `unit_instock`: int64
   ☐ `harga_diskon`: float64
   ☐ `nama_item`: objek
   ☐ `unit_terjual`: objek

2. Apabila ingin mengetahui jumlah kolom di `inventory`. Manakah dari kode berikut yang akan mencetak jumlah kolom saja di `inventory`?

   ☐ `inventory.shape`
   ☐ `inventory.axes[0]`
   ☐ `inventory.axes[1]`

### 4.4 Indexing & Subsetting with Pandas

*Indexing* digunakan untuk memilih dan mengambil sebagian data yang diperlukan dalam proses analisa data yang sedang dikerjakan.

Contoh:

- Membandingkan sales pada tahun 2018 vs 2019
- Identifikasi peluang penjualan pada segmen pasar (ex: Wholesale vs Retail)
- Melihat quarter terbaik untuk setiap tahun yang dapat digunakan untuk tujuan promosi
- dan sebagainya

Terdapat beberapa cara untuk melakukan indexing dan subsetting menggunakan `pandas`:

- `.head()`
- `.tail()`
- `.select_dtypes()`
- `.drop()`
- `[]` operator
- `.loc`
- `.iloc`
- Conditional subsetting

Misalkan kita akan melakukan analisis terhadap data integer saja. Maka kita bisa menggunakan fungsi `.select_dtypes()` untuk memilih kolom yang sesuai dengan tipe data yang diinginkan. Parameter yang digunakan adalah `include` dan `exclude`.

Contoh:

- `include = 'category'` artinya kita memilih semua kolom yang bertipe kategorik saja
- `include = ['int', 'float']` artinya kita memilih semua kolom yang bertipe integer dan float
- `exclude = 'category'` artinya kita memilih semua kolom yang **bukan** bertipe kategorik

[ ]:
```
rice.select_dtypes(include='int')
```

[ ]:

| | receipt_id | receipts_item_id | discount | quantity |
|---|---|---|---|---|
| 1 | 9622257 | 32369294 | 0 | 1 |
| 2 | 9446359 | 31885876 | 0 | 1 |
| 3 | 9470290 | 31930241 | 0 | 3 |
| 4 | 9643416 | 32418582 | 0 | 1 |
| 5 | 9692093 | 32561236 | 0 | 1 |
| … | … | … | … | … |
| 11996 | 5760491 | 17555486 | 0 | 1 |
| 11997 | 5598782 | 16999147 | 0 | 1 |

```
11998      5735850          17434503      3000         1
11999      5678748          17317935      3000         1
12000      5702411          17341559      3000         1

[12000 rows x 4 columns]
```

Selain menggunakan `.select_dtypes()`, kita bisa menggunakan fungsi `.drop()` untuk membuang baris atau kolom yang tidak ingin digunakan berdasarkan `axis`. Nilai `axis = 0` menunjukkan baris sedangkan `axis = 1` menunjukkan kolom. Secara *default*, parameter axis bernilai 0.

[ ]:
```
rice.drop(2).head()
```

[ ]:
```
   receipt_id  receipts_item_id       purchase_time category sub_category
1     9622257          32369294 2018-07-22 21:19:00     Rice         Rice  \
3     9470290          31930241 2018-07-15 12:12:00     Rice         Rice
4     9643416          32418582 2018-07-24 08:27:00     Rice         Rice
5     9692093          32561236 2018-07-26 11:28:00     Rice         Rice
6     9504155          32030785 2018-07-17 18:05:00     Rice         Rice

         format  unit_price  discount  quantity yearmonth
1  supermarket    128000.0         0         1   2018-07
3  supermarket     64000.0         0         3   2018-07
4   minimarket     65000.0         0         1   2018-07
5  supermarket    124500.0         0         1   2018-07
6   minimarket     63500.0         0         1   2018-07
```

Pada code diatas, kita menghilangkan baris index ke-2 dan menggabungkan dengan fungsi `.head()` untuk melihat 5 data pertama. Apabila kita ingin menghapus kolom, kita bisa mengubah nilai `axis` menjadi `1` dan menspesifikkan kolom mana yang ingin dihilangkan.

Apabila terdapat beberapa kolom / baris yang ingin dihilangkan, kita bisa memasukkannya kedalam sebuah list.

Contoh:

```
rice.drop(['purchase_time', 'category'], axis = 1)
```

[ ]:
```
# Your code below
```

Perhatikan bahwa fungsi `.drop()` tidak akan mengubah dataframe awal `rice`. Fungsi tersebut hanya menghilangkan kolom/baris dari dataframe. Apabila kita ingin mengubah dataframe maka terdapat dua cara

1. Assignment ulang dataframe yang sudah di drop `py rice = rice.drop('category', axis = 1)`
2. Menambahkan parameter `inplace = True` pada fungsi `.drop()` `py rice.drop('category', axis = 1, inplace = True)`

Pada umumnya, seringkali Anda ingin melakukan *subsetting* pada sebagian baris tertentu saja. Proses *subsetting* tersebut dapat dilakukan dengan menggunakan operasi `[]`.

Operasi `[]` akan melakukan *subsetting* dengan cara mengiris (*slicing*) index pada dataframe. Formula penulisan operasi tersebut adalah `[start:end]` dengan tetap mengikuti aturan *indexing* dari Python (dimulai dari 0) dimana `start` *inclusive* dan `end` *exclusive*.

Code dibawah ini akan melakukan *slicing* untuk 4 data pertama yaitu index ke 0 ,1, 2, dan 3.

[ ]:
```
rice[0:4]
```

[ ]:
```
   receipt_id  receipts_item_id       purchase_time category sub_category
1     9622257          32369294 2018-07-22 21:19:00     Rice         Rice  \
2     9446359          31885876 2018-07-15 16:17:00     Rice         Rice
3     9470290          31930241 2018-07-15 12:12:00     Rice         Rice
4     9643416          32418582 2018-07-24 08:27:00     Rice         Rice

         format  unit_price  discount  quantity yearmonth
1  supermarket    128000.0         0         1   2018-07
2   minimarket    102750.0         0         1   2018-07
3  supermarket     64000.0         0         3   2018-07
4   minimarket     65000.0         0         1   2018-07
```

Berbeda dengan drop yang harus menentukan nilai axis, operator `[]` dapat melakukan pengambilan untuk nilai kolom dengan langsung memasukkan nama kolomnya. Seperti biasa, apabila ingin mengambil beberapa kolom harus dimasukkan ke dalam sebuah list.

contoh:

```
# Mengambil kolom category saja
rice['category']

# Mengambil kolom purchase_time dan category
rice[['purchase_time', 'category']]
```

**Knowledge Check : Slicing**

1. Proses slicing pada beberapa baris dalam dataframe menggunakan operator `[]` dengan memasukkan nilai start dan end. Bagaimana cara kita melakukan slicing pada dataframe apabila ingin mengambil baris ke 8 sampai 12 ?

Hint : Ingat bahwa Python menggunakan 0-based indexing

- ☐ `rice[7:12]`
- ☐ `rice[8:12]`
- ☐ `rice[7:13]`
- ☐ `rice[8:13]`

Selain menggunakan `drop()` dan operator `[]`. Kita bisa juga menggunakan `.loc` dan `.iloc`.

Perbedaan dari `.loc` dan `.iloc` adalah:

- `.loc` merujuk pada nama baris atau kolomnya
- `.iloc` merujuk pada index baris atau kolomnya (harus integer)

Kita akan berfokus pada penggunaan `.iloc` terlebih dahulu. *Syntax* `.iloc` > dataframe.iloc[baris, kolom]

[ ]:
```
rice.iloc[3:5 , 1:3]
```

[ ]:
```
   receipts_item_id       purchase_time
4          32418582 2018-07-24 08:27:00
5          32561236 2018-07-26 11:28:00
```

Pada *cell* diatas, kita akan melakukan subset pada data `rice` untuk index baris ke 3 dan 4, serta kolom index ke 1 dan 2. Output dari `.iloc` adalah `pandas` dataframe.

Berikut adalah contoh latihan penggunaan `.iloc` yang bisa Anda kerjakan:

1. Tampilkan baris dengan index 1 dan kolom dengan index 2, artinya adalah data pada baris kedua dan kolom `packet_name`.
2. Mengambil baris ke 2 sampai 5 dan kolom dengan `packet_name` sampai `revenue`.
3. Menampilkan semua data pada kolom `packet_name` dan `revenue` saja.

Gabungkan apa yang telah Anda pelajari, coba gunakan `.iloc` untuk subset **2 baris terakhir** dan **4 kolom pertama** dari dataframe di atas! Jika Anda melakukannya dengan benar, output `x` akan berbentuk `pandas` dataframe(`type(x)` adalah objek `DataFrame`).

Bonus: jika Anda ingin tantangan ekstra, coba lakukan operasi ini tiga kali dengan cara;

1. Gunakan hanya `.iloc[ , ]`
2. Gunakan `tail(2)` untuk mendapatkan 2 baris dan gabungkan dengan `.iloc`

3. Gunakan `rice.shape[0]-2:` untuk mendapatkan 2 baris terakhir di operasi `.iloc`

[ ]:
```
## Your code below
```

Berbeda dengan `.iloc`, `.loc` melakukan *slicing* dengan menggunakan `label` atau nama baris dan kolom. Kita tetap bisa menggunakan `integer` dalam `.loc` tetapi itu akan diinterpretasikan sebagai label.

Syntax `.loc`: > dataframe.loc[ baris , kolom]

Mari kita baca lagi `csv` yang sama, tetapi kali ini kita atur `receipt_id` sebagai label baris:

[ ]:
```
rice = pd.read_csv("data_input/rice.csv", index_col=1)
rice = rice.drop('Unnamed: 0', axis=1)
rice.head()
```

[ ]:
```
            receipts_item_id    purchase_time category sub_category
receipt_id
9622257             32369294  7/22/2018 21:19     Rice         Rice  \
9446359             31885876  7/15/2018 16:17     Rice         Rice
9470290             31930241  7/15/2018 12:12     Rice         Rice
9643416             32418582   7/24/2018 8:27     Rice         Rice
9692093             32561236  7/26/2018 11:28     Rice         Rice
```

```
                format  unit_price  discount  quantity yearmonth
receipt_id
9622257     supermarket    128000.0         0         1   2018-07
9446359      minimarket    102750.0         0         1   2018-07
9470290     supermarket     64000.0         0         3   2018-07
9643416      minimarket     65000.0         0         1   2018-07
9692093     supermarket    124500.0         0         1   2018-07
```

Subsetting baris transaksi yang mempunyai id receipt 9643416 and 5735850 dapat dilakukan dengan `.loc` sebagai berikut:

[ ]:
```
rice.loc[[9643416, 5735850], :]
```

[ ]:
```
            receipts_item_id     purchase_time category sub_category
receipt_id
9643416             32418582    7/24/2018 8:27     Rice         Rice  \
5735850             17434503  12/13/2017 19:17     Rice         Rice

                format  unit_price  discount  quantity yearmonth
receipt_id
9643416      minimarket     65000.0         0         1   2018-07
5735850     supermarket     59990.0      3000         1   2017-12
```

### 4.4.1 Conditional Subsetting

Selain menggunakan `.loc` dan `.iloc`, kita dapat melakukan *subsetting* berdasarkan kondisi tertentu. Hal ini dinamakan *Conditional Subsetting*.

Dengan menggunakan *Conditional Subsetting*, Kita dapat memilih data menggunakan kriteria tertentu. Berikut adalah contoh *conditional subsetting* yang bisa dilakukan pada dataframe `rice`

- `.format == 'supermarket'` untuk memilih semua transaksi yang formatnya adalah supermarket
- `.unit_price >= 400000` untuk memilih semua transaksi dengan harga satuan sama dengan atau lebih besar dari 400000.
- `.quantity != 0` untuk memilih semua transaksi yang kuantitas pembeliannya **bukan** 0

Syntax penulisan untuk *conditional subsetting* adalah:

df[ df[‘column_name’] ]

Contoh *comparison* operator:

- `==`: sama dengan
- `!=`: tidak sama dengan / bukan
- `>` : Lebih dari
- `<` : Kurang dari
- `>=`: Lebih dari sama dengan
- `<=`: Kurang dari sama dengan

```
[ ]: rice = pd.read_csv("data_input/rice.csv", index_col=1)
     rice = rice.drop('Unnamed: 0', axis=1)
     rice[rice.discount != 0].head()
```

```
[ ]:            receipts_item_id    purchase_time category sub_category
     receipt_id
     9767244             32763935   7/29/2018 8:25     Rice         Rice  \
     9483559             31978405  7/16/2018 16:24     Rice         Rice
     9841041             33010608   7/29/2018 8:21     Rice         Rice
     9284943             31450853   7/6/2018 12:05     Rice         Rice
     9853312             33070898  7/30/2018 13:54     Rice         Rice

                     format  unit_price  discount  quantity yearmonth
     receipt_id
     9767244     supermarket     59900.0      4100         1   2018-07
     9483559     supermarket     46900.0      4300         1   2018-07
     9841041     supermarket    117900.0     10100         1   2018-07
     9284943     supermarket     59900.0      4100         1   2018-07
     9853312     supermarket    119890.0      8110         1   2018-07
```

### 4.4.2 Referencing and Copying

Pada Python, tanda sama dengan (=) bisa digunakan untuk merujuk ke sebuah Dataframe. Sekarang mari kita coba untuk dengan menggunakan dataframe `rice`

```
[ ]: rice = pd.read_csv("data_input/rice.csv", index_col=1)
     rice = rice.drop('Unnamed: 0', axis=1)

     rice.head()
```

```
[ ]:            receipts_item_id    purchase_time category sub_category
     receipt_id
     9622257             32369294  7/22/2018 21:19     Rice         Rice  \
     9446359             31885876  7/15/2018 16:17     Rice         Rice
     9470290             31930241  7/15/2018 12:12     Rice         Rice
     9643416             32418582   7/24/2018 8:27     Rice         Rice
     9692093             32561236  7/26/2018 11:28     Rice         Rice

                     format  unit_price  discount  quantity yearmonth
     receipt_id
     9622257     supermarket    128000.0         0         1   2018-07
     9446359      minimarket    102750.0         0         1   2018-07
     9470290     supermarket     64000.0         0         3   2018-07
     9643416      minimarket     65000.0         0         1   2018-07
     9692093     supermarket    124500.0         0         1   2018-07
```

Lalu, mari kita buat dataframe baru, `rice_july`, yang menyimpan data penjualan beras khususnya pada bulan Juli 2018. Misalkan pada periode tersebut, kita memberikan diskon 15 persen untuk

semua produk kita, maka kita juga mengganti semua nilai dalam `rice_july.discount` menjadi 15.

[ ]:
```
rice_july = rice
rice_july['discount'] = 15
rice_july.head()
```

[ ]:
```
            receipts_item_id    purchase_time category sub_category
receipt_id
9622257             32369294  7/22/2018 21:19     Rice         Rice  \
9446359             31885876  7/15/2018 16:17     Rice         Rice
9470290             31930241  7/15/2018 12:12     Rice         Rice
9643416             32418582   7/24/2018 8:27     Rice         Rice
9692093             32561236  7/26/2018 11:28     Rice         Rice

                format  unit_price  discount  quantity yearmonth
receipt_id
9622257     supermarket    128000.0        15         1   2018-07
9446359      minimarket    102750.0        15         1   2018-07
9470290     supermarket     64000.0        15         3   2018-07
9643416      minimarket     65000.0        15         1   2018-07
9692093     supermarket    124500.0        15         1   2018-07
```

Data pada `rice_july` sudah berubah pada kolom `discount`. Tetapi apabila kita lihat pada data `rice`, nilai pada kolom `discount` juga berubah

[ ]:
```
rice.head()
```

[ ]:
```
            receipts_item_id    purchase_time category sub_category
receipt_id
9622257             32369294  7/22/2018 21:19     Rice         Rice  \
9446359             31885876  7/15/2018 16:17     Rice         Rice
9470290             31930241  7/15/2018 12:12     Rice         Rice
9643416             32418582   7/24/2018 8:27     Rice         Rice
9692093             32561236  7/26/2018 11:28     Rice         Rice

                format  unit_price  discount  quantity yearmonth
receipt_id
9622257     supermarket    128000.0        15         1   2018-07
9446359      minimarket    102750.0        15         1   2018-07
9470290     supermarket     64000.0        15         3   2018-07
9643416      minimarket     65000.0        15         1   2018-07
9692093     supermarket    124500.0        15         1   2018-07
```

Penjelasan dari apa yang terjadi pada dua kode sebelumnya adalah ketika kita mengeksekusi kode `rice_july = rice` kita **TIDAK** membuat sebuah objek baru. Kita hanya membuat sebuah variabel bernama `rice_july` yang me*refer* satu objek yang sama.

Untuk mengatasi permasalahan tersebut, kita bisa menggunakan fungsi `.copy()`. Fungsi `copy()`

akan **menduplikasi objek** dan **membuat sebuah objek baru**, sehingga terdapat dua objek yang berbeda dan perubahan pada salah satu objek **tidak** akan berpengaruh pada objek yang lain.

[ ]:
```
rice = pd.read_csv("data_input/rice.csv", index_col=1)
rice = rice.drop('Unnamed: 0', axis=1)

rice_july = rice.copy()
rice_july['discount'] = 15
rice_july.head()
```

[ ]:
```
            receipts_item_id    purchase_time category sub_category
receipt_id
9622257             32369294  7/22/2018 21:19     Rice         Rice  \
9446359             31885876  7/15/2018 16:17     Rice         Rice
9470290             31930241  7/15/2018 12:12     Rice         Rice
9643416             32418582   7/24/2018 8:27     Rice         Rice
9692093             32561236  7/26/2018 11:28     Rice         Rice

                format  unit_price  discount  quantity yearmonth
receipt_id
9622257     supermarket    128000.0        15         1   2018-07
9446359      minimarket    102750.0        15         1   2018-07
9470290     supermarket     64000.0        15         3   2018-07
9643416      minimarket     65000.0        15         1   2018-07
9692093     supermarket    124500.0        15         1   2018-07
```

[ ]:
```
rice.head()
```

[ ]:
```
            receipts_item_id    purchase_time category sub_category
receipt_id
9622257             32369294  7/22/2018 21:19     Rice         Rice  \
9446359             31885876  7/15/2018 16:17     Rice         Rice
9470290             31930241  7/15/2018 12:12     Rice         Rice
9643416             32418582   7/24/2018 8:27     Rice         Rice
9692093             32561236  7/26/2018 11:28     Rice         Rice

                format  unit_price  discount  quantity yearmonth
receipt_id
9622257     supermarket    128000.0         0         1   2018-07
9446359      minimarket    102750.0         0         1   2018-07
9470290     supermarket     64000.0         0         3   2018-07
9643416      minimarket     65000.0         0         1   2018-07
9692093     supermarket    124500.0         0         1   2018-07
```