Workshop

October 28, 2019

Welcome to the **Workshop of Data Science**.

In this module you will learn how to code with Python and create simple machine learning with Linear Regression, let's start!

1 Intro to Python

When you are learning programming languages, always start with code **Hello world**, because with that you will start programming and understanding it.

To do that with Python, simply use

```
print("Hello world")
```

[ ]:

You can using your name too, for examples:

```
print("My name is Faul")
```

[ ]:

When you want to change your name, with that code you should changed it, so complicated. So you can use **variables**.

For examples, we have variables called **Name** and simply type your name, and then use this:

```
name = "Faul"
print("My name is", name)
```

With this type, when you want to change, just change **name** variable itself.

[ ]:

You can create your own input with input function, like this

```
name = input("Please input your name: ")
print("My name is", name)
```

[ ]:

So, which one do you prefer? It's up to you.

[ ]:

1.1 Data Types

Now let's talk about data types, we know that in general data types is differenced between **number** and **word**.

In Python, data types is differentiate between **Integer, float, string**

Examples:

```
gpa = 3.56
semester = 7
department = "Informatics"
```

Who is integer, float or string?

[ ]:

To know that, just use function `type(gpa)` or `type(semester)` or `type(department)`, and you can see what are the data type.

[ ]:

1.2 Containers

So basically containers means how you can store many data in just one variable, like this one

```
name_single = "Faul"
name_multi = ["Faul", "Arga", "Clara"]
```

In **name_single** you just save one data, *Faul*, while in **name_multi** you can store many data. That's called containers.

[ ]:

In general there are four containers, **List, Tuple, Dictionary, and Set**

List what you see just above, with **[ ]** one.

You can check out with `type(name_multi)`

[ ]:

Next question is, how to get **Arga**? What do you want to do?

[ ]:

Introducing **index**, so index mean that you can always get your data with him, index start from zero, which means

```
name_multi[0] is equals "Faul"
name_multi[1] is equals "Arga"
name_multi[2] is equals "Clara"
```

[ ]:

How to add another data? You can used function `.append()`, `.extend()`, `.insert()` You can try of these examples, and you know what is difference between them:

```
name_multi.append("Lita")
print(name_multi)

name_multi.extend("Lisa", "Gina", "Fajar")
print(name_multi)

name_multi.insert(2, "Beni")
print(name_multi)
```

I suggest to check one by one to see the difference.

[ ]:

To know where your data is, you can use `.index(data)` and if you want to remove, just `.remove(data)`

```
name_multi.index("Fajar")

name_multi.remove("Lisa")
```

[ ]:

Exercise:

Create simple list called gpa_multi and the value is **2.9, 2.85, 3.75, 3.49, 3.1, 3, 2.87, 3.21** - Add 3.95 and 3.32 into your data - Add 3.3 in third position - Remove 2.85 from your data - Where the position of 3.49?

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

Tuple actually have same as List, but they are using **( )**, and they are immutable, which is you can't adding data.

```
name_multi_tuple = ("Faul", "Arga", "Clara")
```

Of course, you still can get your data using index too.

```
name_multi_tuple[2] is returned value "Clara"
```

[ ]:

[ ]:

Dictionary is special cases, which is you don't use index for this type, but using **key** for returning your **value**, and using **{ }**

```
city_lived = {"Faul" : "Surabaya", "Arga" : "Jakarta", "Clara": "Balikpapan"}
```

`Faul` in there is **key** and `Surabaya` is the **value**

[ ]:

[ ]:

Again, to get the value, simply using **dict[key]**

```
city_lived["Faul"] is returned value "Surabaya"
```

To append, just using **dict[key] = "value"**

```
city_lived["Lisa"] = "Makassar"
print(city_lived)
```

[ ]:

[ ]:

Sets is as same as list or tuples, but using { }. Their main characteristic is just only unique data is store.

```
city_lived_tuples = {"Jakarta", "Surabaya", "Balikpapan", "Makassar", "Jakarta", "Balikpapan"}
print(city_lived_tuples)
```

Unfortunately they can using index for getting the values.

[ ]:

[ ]:

Exercise:
From **gpa_multi** above, create your own dictionary, which contain `name` and `gpa` from **gpa_multi**. Feel free to create name itself.
Who is get gpa 3.3 and 3.1?

[ ]:

[ ]:

1.3 Control Flow

In this section, we are gonna talking about how Python works, systematically. **If-elif-else** is the most common reason when you want to create so many condition.

```
gpa = 3.01
if gpa > 3.5:
    print("Great result!")
elif gpa > 3:
    print("Good one")
else:
    print("Don't give up, get better!")
```

[ ]:

How if I get gpa of 3?

[ ]:

Looping are the best way if you want to get many things frequently. In this case we are using **for** and **while**

```
#with for
for i in range(5):
    print(i)

#with while
i = 0
while i < 5:
    print(i)
    i+=1
```

[ ]:

[ ]:

Can you see the difference?

[ ]:

Exercise:
Combined the custom input with the control flow above, create code that make like this output:

```
How many input do you want? 3
Input your gpa: 3.5
"Good one"
Input your gpa: 2.75
"Don't give up, get better!"
Input your gpa: 3.75
"Great result!"
```

[ ]:

[ ]:

---

2 Introduce Pandas and Numpy

Pandas and Numpy are the main package if you want to analyzing data and computing some of your data to Python.

Pandas is mainly for analyzing data and changing some data, filtering and many more.

Numpy is mainly for computation, such as statistical value, linear algebra, and many more.

```
#Let's import the package
import pandas as pd
import numpy as np
```

[ ]:

```
# Now it's time to import the data
data = pd.read_csv("iris.csv")
data.head()
```

[ ]:

`.head()` is using for getting first five of your data, if you want to get last five of your data, using `.tail()`

[ ]:

For checking the column names, using `.columns`

[ ]:

Getting statistical values? Use `.describe()` and you are all set!

[ ]:

What do you see? A bunch of numbers?

[ ]:

If you want to only filter selected columns, using `data[[column name]]` if more than one, just `data[[column name 1, column name 2]]`

[ ]:

Then, if you want to filter based on data, you can use data `[data[column name] > value itself]`

[ ]:

Quick question: How many data that **sepal_length** is more than 5?

[ ]:

So, what does **Numpy** do?
Numpy doing computation of your data. How to do that?

```
sepal_width_value = data[['sepal_width']].values
print(np.sum(sepal_width_value)
```

[ ]:

As simple as that, for more information you can check it yourself by using `Shift + Tab`.

[ ]:

You can do the computation from Pandas DataFrame.

[ ]:

[ ]:

3 Creating Simple ML (Linear Regression)

In this section we are want to create simple machine learning based on data House Price.
So our aim is to predict the house price based on several condition of house itself.

```
data_house = pd.read_csv("data.csv")
data_house.head()
```

[ ]:

How to check how many your data in this data_house? Simply using `.shape()`

```
data_house.shape()
```

[ ]:

3.1 Analyzing Your Data

In this phase you should know your data, is it some missing values, and some strange data and many more. This phase generally is one of the longest part, with acquiring data.
Check how many **bedrooms** in this data using `.value_counts()`

[ ]:

Do you see strange things? Let's find it out.

[ ]:

Now, let's check maximum **price** you can get in this data, and filter the data based on it.

[ ]:

Don't you feel confused?

[ ]:

[ ]:

3.2 Visualize your Data

How if you want to know how many **condition** of this data? You can visualize using **Matplotlib** and **Seaborn**

```
import matplotlib.pyplot as plt
import seaborn as sns

#to visualize how many condition
plt.title("How many Condition in Data")
sns.countplot(data['condition'])
```

[ ]:

What is the most **condition** in the data? Can you conclude?

[ ]:

How with **Price**? I want to know the distribution. Let's check with `.distplot()`

```
plt.title("Distribution of Price of Data")
sns.distplot(data['price']);
```

[ ]:

Can you see something? Is it too hard? Let's add some code.

```
#just add this code to see better viz
ax = sns.distplot(data['price']);
ax.set_yscale('log')
```

[ ]:

Exercise: - How can I visualize **view** and **yr_built**?

[ ]:

3.3 Selecting your Data

This is the most important things, since that you should select to create the ML works.

One way is using correlation plot to select the features we want to get models. Using `data.corr()` or `data.corr(method='spearman')`

[ ]:

[ ]:

The value of correlation is between -1 and 1, with meaning of these: - (-1) means negative correlation (when A increased, B decreased) - (0) means no correlation (don't have an impact) - (1) means positive correlation (when A increased, B increased too)

Select 5 best features, and then save as **data_model**

[ ]:

3.4 Machine Learning

In this phase you will create simple machine learning.

[ ]:

4 Bonus

```
def modelling(floors, bed, bath, sqft_above, sqft_living):
    prediction = model.predict([[floors, bed, bath, sqft_above, sqft_living]])
    return prediction[0]

input_floors = float(input(("Masukkan jumlah lantai: ")))
input_bedrooms = float(input(("Masukkan jumlah kamar tidur: ")))
```

```
input_bathrooms=float(input(("Masukkan jumlah kamar mandi: ")))
input_sqft_above=float(input(("Masukkan luas ruangan atas: ")))
input_sqft_living=float(input(("Masukkan luas ruangan tamu: ")))

modelling(input_floors, input_bedrooms, input_bathrooms, input_sqft_above, input_sqft_living)
```

[ ]: