1. get input:

```
name = input("What is your name")

//have to convert it to the numerical value
```

2. formatted string

```
f'{first variable} is not the same as the second variable {second variable}'
```

3. Method for string

```
.find()
.replace()
.len()
.upper()
.title()
```

4. for.. in ...:

```
...
// is the item in the list or no?
```

5. 2D List. [[],[],[],[]]

6.List methods

```
.append(34)
.insert(0,23)
.clear()
.remove(4)
.pop() //remove the last element
.index(23)
.count(5)
.sort()
.reverse()
.copy()
```

7. tuple

```
numbers = (1,2,3)
// CANNOT CHANGE TUPLE INFORMATION
```

8. Dictionary

```
customer= {

    "name"  = "johm",
    "age" = 10

}

.get(key)
// each key should be unique
```

9. Split

```
split(" ")  //split by a space
```

10. Exception

```
try:
    age = int(input("What is your age?))
```

```
except ValueError, ZeroDivisionError:
```

11. Creating a Class

```
Class Point:
        def run():
        …
        ….

point 1 = Point()
point1.run()
point1.x = 10
point1.y= 20
```

12. Constructor

```
def __init__(self, x, y):
        self.x = x
        self.y = y
```

13. Inheritance

```
class Mammal:
        def walk(self):

class Dog(Mammal):
        pass
```

14. Excel

```
import openpyxl as xl

# access the excel file
wb = xl.load_workbook("transaction.xlsx")

# access the first Sheet
sheet = wb["Sheet1"]

# access the a1 cell of Sheet 1
cell = sheet["a1"]  or cell = sheet.cell(1,1)


# add chart
from openpyxl.chart import BarChart, Reference

Reference(sheet,min_row =2, max_row=sheet.max_row, min_col = 4, max_col = 4)

chart = BarChart()
chart.add_data(values)
sheet.add_chart(chart, "e2")

wb.save("transaction.xlsx")
```

15. Machine Learning

```
import panda as pd

from sklearn.tree import DecisionTreeClassifier

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score

from sklearn.externals import joblib


df = pd.read_csv("….csv")

df.shape    # This gives the size of the sheet

df.describe()   # This gives the 25% mean…. and so on

df.values  # This returns two dimensional array basic information

X = df.drop(columns=["name of a column"])  # this creates a new table. Input data sets

y = df["name of the column"]  # Output dataset

X_train,X_test, y_train, y_test = train_test_split(X, y ,test_size=0.2) # This is a tuple so we can do the thing at the front


# Training the model
model = DecisionTreeClassifier()
model.fit(X, y)

model.fit(X_train_y_train)

joblib.dump(model, 'music.joblib')  # a binary file  # Th

model = joblib.load(model, 'music.joblib')


predictions= model.predict([[21,1],[22,0]])

predictions = model.predict(X_test)

score = accuracy_score(y_test, predictions)


Visualizing the Tree  #after model.fit

from sklearn import tree

tree.export_graphviz(model, out_file="music-recommender.dot",feature_names=['age','gender'],
class_name= sorted(y.unique()), label ='all',rounded = True, filled = True)
```

17. overfitting, where a model matches the training data almost perfectly, but does poorly in validation and other new data.

When a model fails to capture important distinctions and patterns in the data, so it performs poorly even in training data, that is called **underfitting**.

// 看mean absolute error
from sklearn.metrics import mean_absolute_error

// 控制 overfit, underfit 用 max_leaf_nodes

from sklearn.tree import DecisonTreeRegressor
RandomTreeRegressor 也行

```
def get_mae(max_leaf_nodes, train_X,val_X,train_y, val_y):
        model = DecisionTreeRegressor(max_lead_nodes = max_leaf_nodes, random_state = 0)
        model.fit(train_X, train_y)
        preds_val =model.predict(val_X)
        mae = mean_absolute_error(val_y,preds_val)
        return mae
```

```
compare MAE with differing values of max_leaf_nodes
for max_leaf_nodes in [5, 50, 500, 5000]:
    my_mae = get_mae(max_leaf_nodes, train_X, val_X,
train_y, val_y)
    print("Max leaf nodes: %d  \t\t Mean Absolute Error:
%d" %(max_leaf_nodes, my_mae))
```

16. Pandas

# Creating Data Frame 2乘2的表格
fruits = pd.DataFrame({"Apples":[30], “Bananas”:[21]})

pd.DataFrame({‘Benz’:[1400, “yes”], ‘Toyota’:[600, “No”]},index =[“Price”, “Made in China?”])

# Creating Series 一个list而已
ingredients = pd.Series(["4 cups",'1 cup','2 large','1 can'],index=["Flour",'Milk','Eggs','Spam'],
name =“Dinner")

# convert the table ‘animals’ to another new csv called cows_and_goats.csv
animals.to_csv(‘cows_and_goats.csv')

```
desc = reviews.description
```
or
```
desc = reviews["description"]
```
`desc` is a pandas `Series` object, with an index matching the `reviews` DataFrame. In general, when we select a single column from a DataFrame, we'll get a Series.

iloc[先横着，后竖着]

```
first_description = reviews.description.iloc[0]
```

```
first_row = reviews.iloc[0]

# accessing the first ten columns of the description column, which is column index number 1

# four ways to access the above information
first_descriptions = reviews.iloc[:10,1]
first_descriptions = reviews.description.iloc[:10]
reviews.loc[:9, “description"]
desc.head(10)
```

```
# This will return part of the country column. It contains all rows of Italy.

italian_wines = reviews.loc[reviews.country==‘Italy']

# return reviews 里有country和points的
reviews.loc[(reviews.country == 'Italy') |(reviews.points >= 90)]
```

```
# isin 就选择了固定的value，排除了其他国家的。
top_oceania_wines=reviews.loc[reviews.country.isin(["Australia",'New Zealand’]) & (reviews.points>=95)]
```

```
# 给price那栏里有值的数据
reviews.loc[reviews.price.notnull()]
```

总结

```
# access basic information of a column of a table
reviews.points.mean()
# What unique taster_name are represented in the dataset?
reviews.taster_name.unique()
# How often does each taster_name appear in the dataset
reviews.taster_name.value_counts()
reviews.points.describe()
```

18. MAPS

```
从现有的数据，转化成新的数据。Transform. 原有的数据还会保留。
review_sth_mean = review.sth.mean()
review.sth.map(lambda p: p - review_sth_mean)

或者这样:
review_points_mean = reviews.points.mean()
reviews.points - review_points_mean


转化整个DataFrame
reviews.apply(roe)…
```

19 Grouping and Sorting

出来表格
把某种同类的数据放在一组里。把一样names的这列group起来，同时找到最小price的每一个group

```
reviews.groupby(“names”).price.min()
```

然后可以多个同时来 用agg ()

```
reviews.groupby([“country”]).price.agg([len,min,max])
```

Sorting:

```
countries_reviewed = countries_reviewed.reset_index()
countries_reviewed.sort_values(by=‘len’,  ascending =
True/False)
```

sort by index:

```
countries_reviewed.sort_index()
sort by more than one column at a time
countries_reviewed.sort_values(by=['country', 'len'])
```

20. DataTypes (dtype) and Missing Values

告诉我name是什么type，不是string 而是object

```
reviews.name.dtype
```

转化已知int到float.
reviews.prices.astype(‘float64’)

NaN values are always of the float64 dtype.

把null的📄变成别的值用fillna（“随别写”）
reviews.names.fillna(“sorry”)

21. Renaming and Branding

把points改叫成scores 用rename（）。 改column。
reviews.rename(columns = {“points”: “scores”})

改index。
reviews.rename(index ={0: “first entry”, 1: “second entry”})

Both the row index and the column index can have their own name attribute. The complimentary rename_axis() method may be used to change these names. For example:

reviews.rename_axis(“wines",axis='rows').rename_axis("fields", axis='columns')

合并两个csv 用concat

canadian_youtube = pd.read_csv("../input/youtube-new/CAvideos.csv")

```
british_youtube = pd.read_csv("../input/youtube-new/GBvideos.csv")
```

```
pd.concat([canadian_youtube, british_youtube])
```

.join() 让两个DataFrame合并。

```
powerlifting_combined = powerlifting_meets.set_index("MeetID").join(powerlifting_competitors.set_index("MeetID"))
```

22. Missing Value 处理

整个列drop掉（如果基本上这列没有什么有用的数）；填充一个数值；或者在填充数值基础上再加一列boolean。知道哪个row加了新的数值。