Intro to Python

```
#variables
my_name = "Nink"
age = 30
saving = 1000.25
love_hamburger = True
```

```
#type
type(love_hamburger)
```

```
bool
```

Data Types

- int
- float
- str
- bool

```
x = "100"
x = int(x)
print(x, type(x))
```

```
100 <class 'int'>
```

```
x = 100
x = str(x)
print(x, type(x))
```

```
100 <class 'str'>
```

Type Hint

(just help to expect that it should be which type but if wrong type then it will not error)

```
gpa: float = 3.45
print(gpa)
```

```
3.45
```

String Method

function ที่ถูกสร้างขึ้นมาสำหรับ class นึงๆ

```
len("hello world")
```

```
11
```

```
text = "a duck walks into a bar"
```

```
text.upper()
```

```
'A DUCK WALKS INTO A BAR'
```

```
text = text.replace('duck', 'lion')
print(text)
```

```
a lion walks into a bar
```

```
text.count('a')
```

```
4
```

```
#no of char exclude whitespaces
len(text) - text.count(" ")
```

```
18
```

```
print(text.title())
```

```
A Lion Walks Into A Bar
```

```
list_of_words = text.split(" ")
list_of_words
```

```
['a', 'lion', 'walks', 'into', 'a', 'bar']
```

```
"-".join(list_of_words)
```

```
'a-lion-walks-into-a-bar'
```

```
word = "hello world"
```

```
#slicing information
word[0:4]
```

```
'hell'
```

```
word[6:]
```

```
'world'
```

```
word.split(" ")[1]
```

```
'world'
```

```
#immutable (cannot update or change some charactor in string but we can create ne
programming = "Python"
```

```
cython = "C" + programming[1:]
cython
```

```
'Cython'
```

Data Structures

- list []
- tuple ()
- dictionary {key:value}
- set {}

```
# list => mutable object
shopping_items = ['egg', 'milk', 'bread', 'noodle']
```

```
shopping_items[0]
```

```
'egg'
```

```
shopping_items[0:3]
```

```
['egg', 'milk', 'bread']
```

```
print("orginal", shopping_items)
shopping_items[0] = 'banana'
print("new", shopping_items)
```

```
orginal ['egg', 'milk', 'bread', 'noodle']
new ['banana', 'milk', 'bread', 'noodle']
```

```
#list methods (function) (can search google 'list method')

shopping_items.append("jam")
print(shopping_items)
```

```
['banana', 'milk', 'bread', 'noodle', 'jam']
```

```
shopping_items.append("chocolate")
print(shopping_items)
```

```
['banana', 'milk', 'bread', 'noodle', 'jam', 'chocolate', 'chocolate', 'chocola
```

```
shopping_items.pop()
print(shopping_items)
```

```
['banana', 'milk', 'bread', 'noodle', 'jam']
```

```
shopping_items.append("banana")
```

```
shopping_items.count("banana")
```

```
2
```

```
shopping_items
```

```
['banana', 'milk', 'bread', 'noodle', 'jam', 'banana']
```

```
#mutable object should copy to another object. Otherwise, it will affect the same

print(shopping_items)
shopping_items_2 = shopping_items.copy()
```

```
['banana', 'milk', 'bread', 'noodle', 'jam', 'banana']
```

```
shopping_items_2[0] = 'apple'
print(shopping_items)
print(shopping_items_2)
```

```
['banana', 'milk', 'bread', 'noodle', 'jam', 'banana']
['apple', 'milk', 'bread', 'noodle', 'jam', 'banana']
```

```
#list can contain anything
students = [
    ('toy', 88),
    ('mary', 95),
    ('john', 90)
]
print(students)
```

```
[('toy', 88), ('mary', 95), ('john', 90)]
```

```
students[1][1]
```

```
95
```

```
# tuple => immutable object (cannot update value)

data = (1,2,3,"Nink","data", True)

students = (
    ('toy', 88), ('marry', 99)
)
print(students)
```

```
(('toy', 88), ('marry', 99))
```

```
type(data)
```

```
tuple
```

```
students[1]
```

```
('marry', 99)
```

```
# dictionary => mutable
customer = {
    "first_name": "Nink",
    "last_name": "Pornkamol",
    "age": 30,
    "love_hamburger": True,
    "fav_movies": ['Thor', 'Strange', 'Black adam']
}
```

```
customer["age"]
```

```
30
```

```
customer['fav_movies'][0] = "Snow"
customer
```

```
{'first_name': 'Nink',
 'last_name': 'Pornkamol',
 'age': 30,
 'love_hamburger': True,
 'fav_movies': ['Snow', 'Strange', 'Black adam']}
```

```
customer['city'] = 'London'
customer
```

```
{'first_name': 'Nink',
 'last_name': 'Pornkamol',
 'age': 30,
 'love_hamburger': True,
 'fav_movies': ['Snow', 'Strange', 'Black adam'],
 'city': 'London'}
```

```
if 'country' not in customer:
    customer['country'] = 'UK'
```

```
customer
```

```
{'first_name': 'Nink',
 'last_name': 'Pornkamol',
 'age': 30,
 'love_hamburger': True,
 'fav_movies': ['Snow', 'Strange', 'Black adam'],
 'city': 'London',
 'country': 'UK'}
```

```
del customer['country']
customer
```

```
{'first_name': 'Nink',
 'last_name': 'Pornkamol',
 'age': 30,
 'love_hamburger': True,
 'fav_movies': ['Snow', 'Strange', 'Black adam'],
 'city': 'London'}
```

```
type(customer)
```

```
dict
```

```
list(customer.keys())
```

```
['first_name', 'last_name', 'age', 'love_hamburger', 'fav_movies', 'city']
```

```
list(customer.values())
```

```
['Nink', 'Pornkamol', 30, True, ['Snow', 'Strange', 'Black adam'], 'London']
```

```
#convert from tuple to list

list(customer.items())
```

```
[('first_name', 'Nink'),
 ('last_name', 'Pornkamol'),
 ('age', 30),
 ('love_hamburger', True),
 ('fav_movies', ['Snow', 'Strange', 'Black adam']),
 ('city', 'London')]
```

```
#set => unique values

fruits = ['banana', 'banana', 'apple', 'lemon']
```

```
len(set(fruits))
```

```
3
```

```
unique_fruits = set(fruits)
```

```
unique_fruits.add("grape")
print(unique_fruits)
```

```
{'lemon', 'apple', 'banana', 'grape'}
```

```
fruits_friend = {'banana', 'orange', 'apple'}
```

```
fruits_friend.intersection(unique_fruits)
```

```
{'apple', 'banana'}
```

```
fruits_friend.union(unique_fruits)
```

```
{'apple', 'banana', 'grape', 'lemon', 'orange'}
```

```
#variables
#data types/ hint
#data structure: list tuple dict set
```

```
#fuction
#cpntrol flow
#OOP
```

```
#f-strings template
```

```
name = 'Nink'
age = 20
fav_lang = 'R'
```

```
template = f"Hello My name is {name} and I'm {age}. My fav lang is {fav_lang}"
```

```
print(template)
```

```
Hello My name is Nink and I'm 20. My fav lang is R
```

```
#create our own functions
```

```
def greeting(name: str) -> None :
    print(f"hello {name}!")
```

```
greeting("Anaa")
```

```
hello Anaa!
```

```
def add_two_nums(a: int, b: int) -> int :
    """
     input: two int nums
     output: sum of two nums
     """
    return a+b
```

```
add_two_nums(19, 30)
```

```
49
```

```
result = add_two_nums(10,30)
print(result)
```

```
40
```

```
#labda function (fit with what is not complex one like 1-2 lines)
greeting = lambda name: print(f"Hello {name}")
```

```
greeting("Annn")
```

```
Hello Annn
```

```
#default arguments
def cube(base=10, pow=3):
    return base ** pow
```

```
cube(base=3, pow=3)
```

```
27
```

```
# control flow
```

Control flow

- if
- for
- while

```
score = int(input("Score: ")) #from 100

if score >= 80:
    print("Passed")
else:
    print("Failed")
```

```
Score:  50
Failed
```

```
#get input from user
username = input("What is your username: ")
```

```
What is your username:  nink
```

```
username
```

```
'nink'
```

```
def grading(score: int) -> None:
    if score >= 80:
        print("Distiction")
    elif score >= 50:
        print("Passed")
    else:
        print("Failed")
```

```
grading(60)
```

```
Passed
```

```
# for loop
shopping_list = ['egg', 'milk', 'bread']
new_list =[]  #empty list
for item in shopping_list:
    # print(f"I have to buy {item.upper()}")
    new_list.append(item.upper())
```

```
new_list
```

```
['EGG', 'MILK', 'BREAD']
```

```
shopping_list = ['egg', 'milk', 'bread']
```

```
#list comprehension
new_list = [item.upper() for item in shopping_list]
print(new_list)
```

```
['EGG', 'MILK', 'BREAD']
```

```
#if-else + for loop
scores = [95,90,75,79,82]

grades = []

for score in scores:
    if score >=80:
        grades.append("Passed")
    else:
        grades.append("Failed")

print(grades)
```

```
['Passed', 'Passed', 'Failed', 'Failed', 'Passed']
```

```
#While loop

#while True:
    #do something

count = 0

while count <5 :
    print("hello:", count)
    count += 1          #meaning count = count +1 and if forgot this line then it
```

```
hello 0
hello 1
hello 2
```

```
hello 3
hello 4
```

```python
# multiple condition

weather: str = 'sunny'
weekday: bool = True

if weather == 'sunny' and weekday:
    print("Go to Starbucks")
else:
    print("Stay home")
```

```
Go to Starbucks
```

```python
#while loop
count = 0

while True:
    print(count)
    count += 1
    if count == 10:
        print("Progran Ends.")
        break
```

```
0
1
2
3
4
5
6
7
8
9
Progran Ends.
```

```
def breaker(n):
    count = 0
    while True:
        print(count)
        count += 1
        if count == n:
            print("Progran Ends.")
            break
```

```
breaker(2)
```

```
0
1
Progran Ends.
```

OOP

```
#class เหมือนแม่พิมพ์ object เหมือนพิมปั๊มที่1 ที่ 2
#In class, there are attribut as variable and method as function we can manipulat
class Dog:
    def __init__(self, name, age, color):  #init is speacial method
        self.name = name
        self.age = age
        self.color = color

    def hello(self):
        print(f"Hi my name is {self.name}!")

    def get_older(self):
        self.age += 1
        print(f"I am getting older one year. I am now {self.age}")
```

```
dog1 = Dog("milo",2 , "black")
dog2 = Dog("rambo", 3, "golden")

dog1.hello()
dog2.hello()
```

```
Hi my name is milo!
Hi my name is rambo!
```

```
dog1.get_older()
```

```
I am getting older one year. I am now 5
```

```
dog2.get_older()
```

```
I am getting older one year. I am now 5
```

```
# ATM
```

```
class ATM:
    def __init__(self, name, bal):  #__ = double underscore or dunder
        self.name = name
        self.bal = bal

    def check_bal(self):
        message = f"Account: {self.name}, Balance: {self.bal}"
        print(message)

    def deposit(self, money):
        self.bal += money
        print(f"New Balance: {self.bal}")
        print("Deposit successfully!")

    def change_name(self, new_name):
        self.name = new_name
        print(f"New Name: {self.name}")
        print("Your account name is changed")
```

```
scb = ATM("nink", 500)
```

```
print(scb.name, scb.bal)
```

```
nink 500
```

```
scb.check_bal()
```

```
Account: nink, Balance: 500
```

```
scb.deposit(900)
```

```
New Balance: 1400
Deposit successfully!
```

```
scb.change_name("marry")
```

```
New Name: marry
Your account name is changed
```

```
scb.name
```

```
'marry'
```

```
ttb = ATM("John", 15000)
```

```
ttb.check_bal()
```

```
Account: John, Balance: 15000
```

```
#Homework
#1. pao ying chub -> python
#2  Continue for ATM => at least 5 methods
```