🎄 Python Programming 101 - DSB11

sunsun-datathyme\

1. variable
2. data type
3. data structure
4. control flow
5. function

```
  # variable
  x = 100
  y = 200
  print(x + y)
```

```
300
```

```
# naming variable => snake case
# data types => int, float, string , bool
my_name = "jay"
my_age = 25
gpa = 3.72
netflix = True
```

```
print(my_name, my_age, gpa, netflix)
```

```
jay 25 3.72 True
```

```
print("Hello!", "my name is", "jay")
```

```
Hello! my name is jay
```

```
# create multiple variables
# tuple unpacking
name, age, gpa = "jay", 25, 3.72
print(name, age, gpa)
```

```
jay 25 3.72
```

```
## replace value
my_name = "jay"
my_name = "Jay"
print(my_name)
```

```
Jay
```

```
## remove varaible
del my_name
```

```
my_name
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-3484297142.py in <cell line: 0>()
----> 1 my_name

NameError: name 'my_name' is not defined
```

🐣 data types conversion

👉 *int( ), folat( ), string ( ), bool( )*

```
## data types conversion
# int(), folat(), string(), bool()
bool(1)
```

```
True
```

```
str(100)
```

```
'100'
```

```
int("555")
```

```
555
```

```
float("3.41")
```

```
3.41
```

🏠 get input from user

```
## get input from user
```

```
input("What's your name: ")
```

```
What's your name: jay
'jay'
```

```
age = int(input("What's your age: "))
```

```
What's your age: 37
```

```
age
```

```
37
```

```
print(age, type(age))
```

```
37 <class 'int'>
```

```
"500" + "1000"
```

```
'5001000'
```

```
int("500") + int("1000")
```

```
1500
```

```
"I love " + "Python"
```

```
'I love Python'
```

🏠 fstring template

```
## fstring template
my_name = " jay"
my_age = 25
text = f"Hi! my name is {my_name} and my age is {my_age} year old."
print(text)
```

```
Hi! my name is  jay and my age is 25 year old.
```

🏠 function

```
## function
## defind function
def double(x):
    return x * 2
```

```
double(100)
```

```
200
```

```
def greeting(name, food):
    text = f"{name} likes to eat {food}"
    return text
```

```
greeting("jay", "french fries")
```

```
'jay likes to eat french fries'
```

```
## default argument
def greeting2(name="jay", food="hot dog"):
    text = f"{name} likes to eat {food}"
    return text
```

```
greeting2()
```

```
'jay likes to eat hot dog'
```

```
greeting2("jenny")
```

```
'jenny likes to eat hot dog'
```

```
greeting("jenny", "coke")
```

```
'jenny likes to eat coke'
```

✅ function more than one input

```
## function return more than one input
def greeting3(x):
    return x**2, x+2 , "useless"
```

```
x, y, _ = greeting3(5)
print(x, y)
```

```
25 7
```

🏠 modularity (modular programming)

```
# modularity (modular programming)
def f1():
    print("hi")

def f2():
    print("hello")
```

```
def f3():
    print("ni hao!")

def f4():
    f1() # load data
    f2() # clean data
    f3() # prep data
    print("done!!!")
```

```
f4()
```

```
hi
hello
ni hao!
done!!!
```

🏠 control flow

🎯 if-else, for, while

```
def grade(score):
    if score >= 90:
        return "Passed"
    else:
        return "Please retake the exam"
```

```
grade(92)
```

```
'Passed'
```

```
grade(82)
```

```
'Please retake the exam'
```

```
## if elif else
def grade_adv(score):
    if score >= 90:
        return "A"
    elif score >= 80:
        return "B"
    elif score >= 70:
        return "c"
    else:
        return "Please retake the exam!"
```

```
grade_adv(65)
```

```
'Please retake the exam!'
```

```
## multiple condition
## and, or
def grade_adv2(score):
    if score >= 90 and score <= 100:
        return "A"
    elif score >= 80 and score < 90:
        return "B"
    else:
        return "Retake the exam!"
```

```
def testing():
    if (1+1 == 2 or 2*2 == 4):
        print("correct")
    else:
        print("incorrect")
```

🏠 Python basic Data structure

1. list
2. tuple
3. dictionary
4. set

🎯 1. list (similar to vector in R)

```
## 1. list (similar to vector in R)
shopping_list = ["egg", "milk", "bread"]

## index starts at zero[0]
print(shopping_list[0])
print(shopping_list[1])
print(shopping_list[2])

## check number of items
print(len(shopping_list))
```

```
egg
milk
bread
3
```

```
## update value in list
## mutable data structure (can be updated data)
shopping_list[0] = "orange"
shopping_list[1] = "milk 2 gallons"
print(shopping_list)
```

```
['orange', 'milk 2 gallons', 'bread']
```

```
type(shopping_list)
```

```
list
```

✅ list method

method is a function designed for a specific data structures/ types

```
## list method
## method is a function designed for a specific data structures/ types
shopping_list.append("butter")
print(shopping_list)
```

```
['orange', 'milk 2 gallons', 'bread', 'butter']
```

```
shopping_list.append("banana")
print(shopping_list)
```

```
['orange', 'milk 2 gallons', 'bread', 'butter', 'banana']
```

```
len(shopping_list)
```

```
5
```

```
## remove the last item .pop (ลบตัวสุดท้ายด้านขวา)
shopping_list.pop()
print(shopping_list)
```

```
['orange', 'milk 2 gallons', 'bread', 'butter']
```

```
# remove item (เลือกได้ว่าจะลบตัวไหน)
shopping_list.remove("bread")
print(shopping_list)
```

```
['orange', 'milk 2 gallons', 'butter']
```

```

```

```
## insert item
shopping_list.insert(1, "chocolate") # (index, "item")
print(shopping_list)
```

```
['orange', 'chocolate', 'milk 2 gallons', 'butter']
```

```
shopping_list.insert(2, "coke") # (index, "item")
print(shopping_list)
```

```
['orange', 'chocolate', 'coke', 'milk 2 gallons', 'butter']
```

```
## sort data
shopping_list.sort()
print(shopping_list)
```

```
['butter', 'chocolate', 'coke', 'milk 2 gallons', 'orange']
```

```
## sort data descending order
shopping_list.sort(reverse=True) # from Z to A
print(shopping_list)
```

```
['orange', 'milk 2 gallons', 'coke', 'chocolate', 'butter']
```

✅ combine two lists

```
## combine two lists
full_list = ["egg", "milk"] + ["butter", "coke"]
print(full_list)
```

```
['egg', 'milk', 'butter', 'coke']
```

🎯 for loop

```
## for loop
for item in full_list:
    print(item)
```

```
egg
milk
butter
coke
```

```
fruits = ["banana", "orange", "strawberry"]

for fruit in fruits:
    print(fruit)
```

```
banana
orange
strawberry
```

```
for fruit in fruits:
    if fruit == "banana":
        print("The banana is very delicious")
    else:
        print(fruit)
```

```
The banana is very delicious
orange
strawberry
```

```
## mutable vs. immutable
## string is immutable
text = "a duck walk into a bar"
text = text.replace("duck", "lion")
print(text)
```

```
a lion walk into a bar
```

```
language = "Python"
new_language = "C" + language[1: ]
print(new_language)
```

```
Cython
```

```
## mutable
friends = ["jay", "jenny", "joe"]
friends[0] = "jayler"
print(friends)
```

```
['jayler', 'jenny', 'joe']
```

```
## immutable
my_name = "Jayler"
my_name = "T" + my_name[1: ]
my_name
```

```
'Tayler'
```

```
## if-else
## for loop

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

```
0
1
2
3
4
```

```
list(range(5))
```

```
[0, 1, 2, 3, 4]
```

```
for i in range(5):
    print("Hello")
```

```
Hello
Hello
Hello
Hello
Hello
```

```
for i in range(5):
    print(i, "Hello")
```

```
0 Hello
1 Hello
2 Hello
3 Hello
4 Hello
```

```
for i in range(10):
    print(i+1, "Hello")
```

```
1 Hello
2 Hello
3 Hello
4 Hello
5 Hello
6 Hello
7 Hello
8 Hello
9 Hello
10 Hello
```

```
for i in range(1, 6):
    print(i)
```

```
1
2
3
4
5
```

🎯 While loop

```
## while loop

alive = True

while (alive is True):
    print("live learn repeat")
    alive += 1
```

```
live learn repeat
```

```
# while loop
count = 0

while (count < 5):
    print("Hi!")
    count += 1
```

```
Hi!
Hi!
Hi!
Hi!
Hi!
```

```
while True:
    user_input = input("What do you want to eat? ")
    print(user_input)
    if user_input == "I'm full!":
        print("Bye!!")
        break
```

```
What do you want to eat? banana
banana
What do you want to eat? coke
coke
What do you want to eat? I'm full!
I'm full!
Bye!!
```

```
play = True

while play:
    user_input = input("What do you want to eat? ")
    print(user_input)
    if user_input == "full":
        print("Bye Bye!!")
        play = False
```

```
What do you want to eat? pizza
pizza
What do you want to eat? water
water
What do you want to eat? apple
apple
What do you want to eat? orange
orange
What do you want to eat? full
full
Bye Bye!!
```

```
alive = True

while alive:
    print("Live Learn Repeat!")
    ui = input("Do you want to stop? ")
    if ui == "yes":
        alive = False
```

```
Live Learn Repeat!
Do you want to stop? no
Live Learn Repeat!
Do you want to stop? no
Live Learn Repeat!
Do you want to stop? nope
Live Learn Repeat!
Do you want to stop? yea
Live Learn Repeat!
Do you want to stop? yes
```

data structure

```
## data structure
## list, tuple , dict , set

laptops = ["dell" ,"lenovo", "macbook"]

result = [] ## empty list

for laptop in laptops:
    tmp = laptop.upper()
    result.append(tmp)

print(result)
```

```
['DELL', 'LENOVO', 'MACBOOK']
```

```
## list comprehention
laptops = ["dell" ,"lenovo", "macbook"]
```

```
laptops_upper = [laptop.upper() for laptop in laptops]

print(laptops_upper)
```

```
['DELL', 'LENOVO', 'MACBOOK']
```

```
## tuple immutable
x = (1, 2, 3)
print(x, type(x))

name , age = ("jay", 25)
print(name, age)
```

```
(1, 2, 3) <class 'tuple'>
jay 25
```

```
## tuple and list
## can be keep multiple data type

["jay", 25, ["R", "Python", "SQL"], ("Econimic", "Marketing"), True]
```

```
['jay', 25, ['R', 'Python', 'SQL'], ('Econimic', 'Marketing'), True]
```

```
(1, (2, 3, 4), [4, 5, 6])
```

```
(1, (2, 3, 4), [4, 5, 6])
```

set (no duplicates)

```
## set (no duplicates) unique value

fruits = {"orange", "orange", "lemon", "lemon", "lemon", "grape"}
```

```
fruits
```

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

```
## set operation
a = {"orange", "banana"}
b = {"orange", "grapde", "pineapple"}
```

```
## interset (inner join)
a & b
```

```
{'orange'}
```

```
## union (full join)
a | b
```

```
{'banana', 'grapde', 'orange', 'pineapple'}
```

```
## difference
a - b
```

```
{'banana'}
```

🎯 dictionary

key-value pair (like a JSON)

```
## dictionary
## key-value pair
my_dict = {
  "name": "John Doe",
  "age": 30,
  "city": "New York",
  "isEmployed": True,
  "hobbies": ["reading","hiking","coding"],
  "address": {
    "street": "123 Main St",
    "zipcode": "10001"
  }
}
```

```
my_dict
```

```
{'name': 'John Doe',
 'age': 30,
 'city': 'New York',
 'isEmployed': True,
 'hobbies': ['reading', 'hiking', 'coding'],
 'address': {'street': '123 Main St', 'zipcode': '10001'}}
```

```
## key must be immutable
## dictionary is mutable
user = {
    "name": "jay",
    "age": 25,
    "location": "BKK",
    "streaming": {"netflix": True,
                  "amazon": False},
    "fav_movies": ["Superman", "Dark Knight", "Marvel"]
}
```

```
user
```

```
{'name': 'jay',
 'age': 25,
 'location': 'BKK',
 'streaming': {'netflix': True, 'amazon': False},
 'fav_movies': ['Superman', 'Dark Knight', 'Marvel']}
```

```
## update value
user["age"] = 28
user["name"] = "Jayler"
print(user["age"], user["name"])
```

```
28 Jayler
```

```
user
```

```
{'name': 'Jayler',
 'age': 28,
 'location': 'BKK',
 'streaming': {'netflix': True, 'amazon': False},
 'fav_movies': ['Superman', 'Dark Knight', 'Marvel']}
```

```
## create new key
user["dog_name"] = "Labubu"
user
```

```
{'name': 'Jayler',
 'age': 28,
 'location': 'BKK',
 'streaming': {'netflix': True, 'amazon': False},
 'fav_movies': ['Superman', 'Dark Knight', 'Marvel'],
 'dog_name': 'Labubu'}
```

```
## delete key
del user["dog_name"]
user
```

```
{'name': 'Jayler',
 'age': 28,
```

```
 'location': 'BKK',
 'streaming': {'netflix': True, 'amazon': False},
 'fav_movies': ['Superman', 'Dark Knight', 'Marvel']}
```

```
user["fav_movies"][0]
```

```
'Superman'
```

```
user["fav_movies"][-1]
```

```
'Marvel'
```

```
user["fav_movies"][0:3]
```

```
['Superman', 'Dark Knight', 'Marvel']
```

```
# slicing
user["streaming"]["netflix"]
```

```
True
```

```
user["fav_movies"][0: ]
```

```
['Superman', 'Dark Knight', 'Marvel']
```

🎯 import modules

```
# import modules
import math
# from import pi, log, exp
```

```
math.pi
```

```
3.141592653589793
```

```
math.log(5)
```

```
1.6094379124341003
```

```
math.exp(5)
```

```
148.4131591025766
```

```

```