BASIC SESSION

#3 문자열, 함수, 모듈, 클래스

DATA&ANALYTICS

INDEX

0 지난시간 과제 풀이 우수자

교수님의강조 – 이승학
다이너마이트 – 고민성
머신129 – 최영재
민정아언니오빠들밥사조 – 나요셉
분석해조 – 김진비
불금엔코딩이조 – 이예진
상부상조 – 주민지
최장사이조 – 이정호
A+의 징조 – 김은욱

1 문자열 포맷팅

문자열 포맷팅

- 문자열 포맷팅이란 문자열 안에 어떤 값을 삽입하는 방법

발전 과정에 따른 종류

- %서식

```
name = '장성민'
class_of = 16
grade = 4

print('반갑습니다. 저는 %d학번 %d학년 %s입니다.'%(class_of, grade, name))
```

```
반갑습니다. 저는 16학번 4학년 장성민입니다.
```

- format

```
name = '장성민'
class_of = 16
grade = 4

print('반갑습니다. 저는 {}학번 {}학년 {}입니다.'.format(class_of, grade, name))
```

```
반갑습니다. 저는 16학번 4학년 장성민입니다.
```

- F-string

```
name = '장성민'
class_of = 16
grade = 4

print(f'반갑습니다. 저는 {class_of}학번 {grade}학년 {name}입니다.')
```

```
반갑습니다. 저는 16학번 4학년 장성민입니다.
```

2 문자열 메소드

문자열 메소드

- 문자열을 처리하는 다양한 기능의 메소드

1. [upper, lower, capitalize, title]
2. [startswith, endswith, swapcase]
3. [split, join, replace, in, find]
4. [isdigit, isalpha, isalnum, islower, isupper, isspace]
5. [strip, rstrip, lstrip]
6. [::-1]

모든 코드

2 문자열 메소드

문자열 메소드

example_str = ‘Tufo got too fat'

1. [upper, lower, capitalize, title]

upper

```
example_str.upper()
'TUFO GOT TOO FAT'
```

lower

```
example_str.lower()
'tufo got too fat'
```

capitalize

```
example_str.capitalize()
'Tufo got too fat'
```

title

```
example_str.title()
'Tufo Got Too Fat'
```

2. [startswith, endswith, swapcase]

startswith

```
example_str.startswith('Tufo')
True
```

endswith

```
example_str.endswith('Tufo')
False
```

swapcase

```
example_str.swapcase()
'tUFO GOT TOO FAT'
```

2 문자열 메소드

문자열 메소드

3. [split, join, replace, in, find]

split

join

replace

in

find

```
print(example_str.split())
print(example_str.split('too'))
```

```
['Tufo', 'got', 'too', 'fat']
['Tufo got ', ' fat']
```

```
print(' '.join(['Tufo', 'got', 'too', 'fat']))
print('/'.join(['Tufo', 'got', 'too', 'fat']))
```

```
Tufo got too fat
Tufo/got/too/fat
```

```
example_str.replace('Tufo', 'Cheese')
```

```
'Cheese got too fat'
```

```
print('Cheese' in example_str)
print('Cheese' in example_str.split())
```

```
False
False
```

```
print(example_str.find('Tufo'))
print(example_str.find('g'))
```

```
0
5
```

2 문자열 메소드

문자열 메소드

4. [isdigit, isalpha, isalnum, islower, isupper, isspace]

isdigit

```
print('01012345678'.isdigit())
print('010abc'.isdigit())
```

```
True
False
```

isalpha

```
print('010abc'.isalpha())
print('abcdefgh'.isalpha())
```

```
False
True
```

isalnum

```
print('01012345678'.isalnum())
print('010abc'.isalnum())
print('abcdefgh'.isalnum())
print(example_str.isalnum())
```

```
True
True
True
False
```

islower

```
print(example_str.islower())
print('abcdefgh'.islower())
```

```
False
True
```

isspace

```
print(example_str.isspace())
print('    '.isspace())
```

```
False
True
```

2 문자열 메소드

문자열 메소드

5. [strip, rstrip, lstrip]

```
print('    Tufo    '.strip())

print('////Tufo////'.strip('/'))

print('////Tufo////'.rstrip('/'))

print('////Tufo////'.lstrip('/'))
```

```
Tufo
Tufo
////Tufo
Tufo////
```

6. [::-1]

```
print(example_str)

print(example_str[0:3])

print(example_str[:3])

print(example_str[:3:1])

print(example_str[:3:2])
```

```
Tufo got too fat
Tuf
Tuf
Tuf
Tf
```

```
print(example_str)

print(example_str[:-1])

print(example_str[::-1])
```

```
Tufo got too fat
Tufo got too fa
taf oot tog ofuT
```

3 함수

함수(function)는 특정한 작업을 하는 코드 조각이다

- 함수를 호출(call)할 때는 함수명 뒤에 ( )을 붙여준다

```
sum([1,2,3])
6

sum((4,5,6))
15
```

- 함수에 따라서는 실행에 필요한 어떤 값들을 넘겨줘야 하는데, 이를 인자(argument)라고 한다.

```
def function_name (argument1, argument2 ...):
    과정 1
    과정 2
    return output

function_name(argument1, argument2 ...)
```

종류

- 내장함수, 모듈함수, 사용자 정의함수, 무명함수(lambda함수)가 있다.

3 함수

종류

- 내장함수, 모듈함수, 사용자 정의함수, 무명함수(lambda함수)가 있다.

내장함수 모듈함수 사용자 정의함수 무명함수

Built-in Functions				
abs()	divmod()	input()	open()	staticmethod()
all()	enumerate()	int()	ord()	str()
any()	eval()	isinstance()	pow()	sum()
basestring()	execfile()	issubclass()	print()	super()
bin()	file()	iter()	property()	tuple()
bool()	filter()	len()	range()	type()
bytearray()	float()	list()	raw_input()	unichr()
callable()	format()	locals()	reduce()	unicode()
chr()	frozenset()	long()	reload()	vars()
classmethod()	getattr()	map()	repr()	xrange()
cmp()	globals()	max()	reversed()	zip()
compile()	hasattr()	memoryview()	round()	__import__()
complex()	hash()	min()	set()	apply()
delattr()	help()	next()	setattr()	buffer()
dict()	hex()	object()	slice()	coerce()
dir()	id()	oct()	sorted()	intern()

3 함수

종류

- 내장함수, 모듈함수, 사용자 정의함수, 무명함수(lambda함수)가 있다.

내장함수

모듈함수

사용자 정의함수

```
def function_name (argument1, argument2 ...):
    과정 1
    과정 2
    return output

function_name(argument1, argument2 ...)
```

무명함수

```
함수이름 = lambda 인자 : 수식
```

```
function = lambda x, y : x + y
function(10, 20)
```

4 Args 인자

Argument is the actual value of this variable that passed to a function
인자는 정의된 함수를 부를 때 매개변수의 값으로 넘겨주는 실제의 값이다.

가변인자
- 가변인자는 인자의 개수가 정해지지 않아 자유롭게 내부 인자가 변할 수 있는 것

```
def mySum(x):
    return sum([x])

print(mySum(5))

print(mySum(1,3,5))
```

```
5
---------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent
<ipython-input-91-f3464e0f71ce> in <module>
      4 print(mySum(5))
      5
----> 6 print(mySum(1,3,5))

TypeError: mySum() takes 1 positional argument but 3 were given
```

```
def mySum(*x):
    return sum([x])

print(mySum(5))

print(mySum(1,3,5))
```

```
5
9
```

4 Args 인자

Packing 과 Unpacking

- Packing: 가변인자를 묶는 것
- Unpacking: 묶여진 가변인자를 푸는 것

```
def function_example(a=0, b=1, c=2, d=3, e=4, f=5):
    return a + b + c + d + e + f
```

```
function_example()
```

```
15
```

```
# 1.
function_example([1, 3, 5, 7, 9, 11])

# 2.
function_example(1, 3, 5, 7, 9, 11)
```

```
lst = [1, 3, 5, 7, 9, 11]
function_example(*lst) # unpacking
```

```
36
```

4 Args 인자

Args 와 Kwargs

- args: arguments의 약자 -> *(Asterisk) 한번
- kwargs: multiple keyword arguments의 약자 -> **(Asterisk) 두번

```
args = [1, 3, 5, 7, 9]

kwargs = {'민정': '1차시', '예원': '2차시', '성민': '3차시', '영진': '4차시'}
```

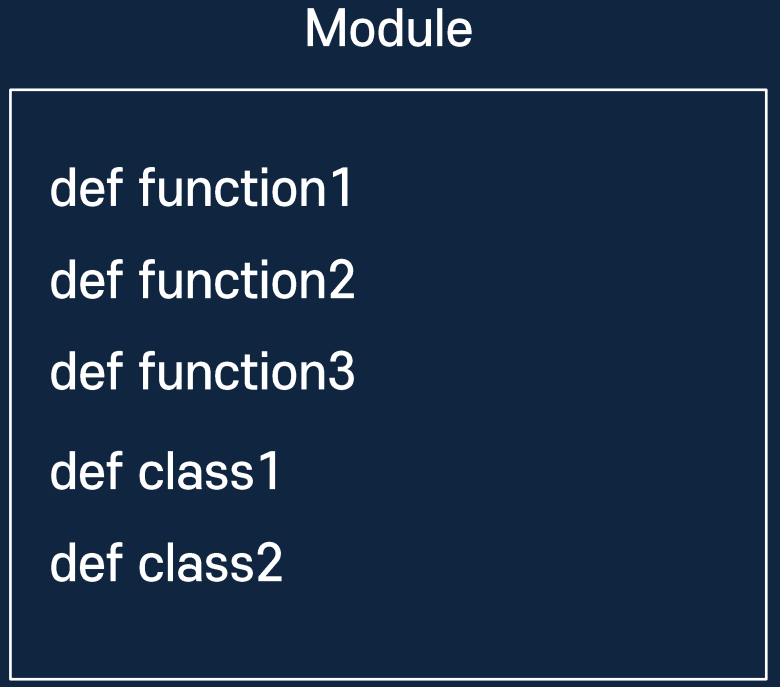
활용예시

```
('XGBClassifier', XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
    colsample_bynode=1, colsample_bytree=0.8699804362619725,
    gamma=1.0615579637577817, learning_rate=0.031871635154535276,
    max_delta_step=0, max_depth=6, min_child_weight=2, missing=None,
    n_estimators=137, n_jobs=-1, nthread=None,
    objective='binary:logistic', random_state=0, reg_alpha=0,
    reg_lambda=1, scale_pos_weight=1, seed=None, silent=None,
    subsample=0.795570685735397, verbosity=1), 0.7680137453399286),

('CatBoostClassifier', CatBoostClassifier(**cat_best_params, od_type='Iter' ,verbose=False),
                       0.76985012
```

5 모듈과 클래스

모듈 : 어떤 언어가 되던 모듈과 클래스에 대해서 이해하고 있어야 합니다. 모듈이란 상수, 변수, 함수, 클래스를 포함하는 코드 패키지입니다. 파이썬 내부에도 여러 모듈이 제공되어 있지만 우리는 다양한 기능을 사용하기 위해서 외부에서 다른 개발자들이 미리 만든 모듈을 불러와 해당 모듈에 들어있는 기능들을 사용하곤 합니다.

1. 모듈 만들기
2. 모듈을 불러오는 여러가지 방법
3. 모듈의 별명을 설정하는 방법

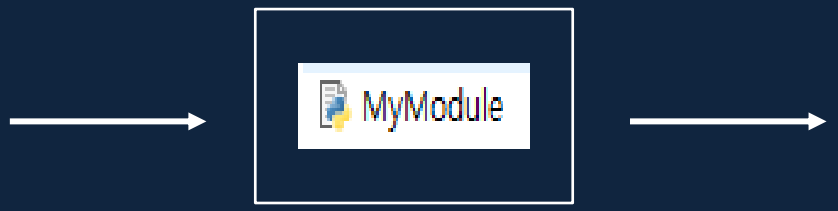


5 모듈

1. 모듈 만들기

```
%%writefile MyModule.py

def MyAdd(a, b):
    return a + b

def MyMax(iterable):
    answer = 0
    for i in iterable:
        if answer < i:
            answer = i
    return answer

def HelloWorld():
    return 'HelloWorld'
```

```
Overwriting MyModule.py
```



MyModule.py - C:₩Users₩82105₩Desktop

File Edit Format Run Options Window

```
def MyAdd(a, b):
    return a + b

def MyMax(iterable):
    answer = 0
    for i in iterable:
        if answer < i:
            answer = i
    return answer

def HelloWorld():
    return 'HelloWorld'
```

[매직메소드]

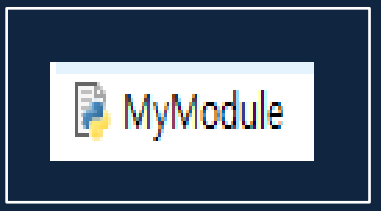
5 모듈

2. 모듈을 불러오는 여러가지 방법

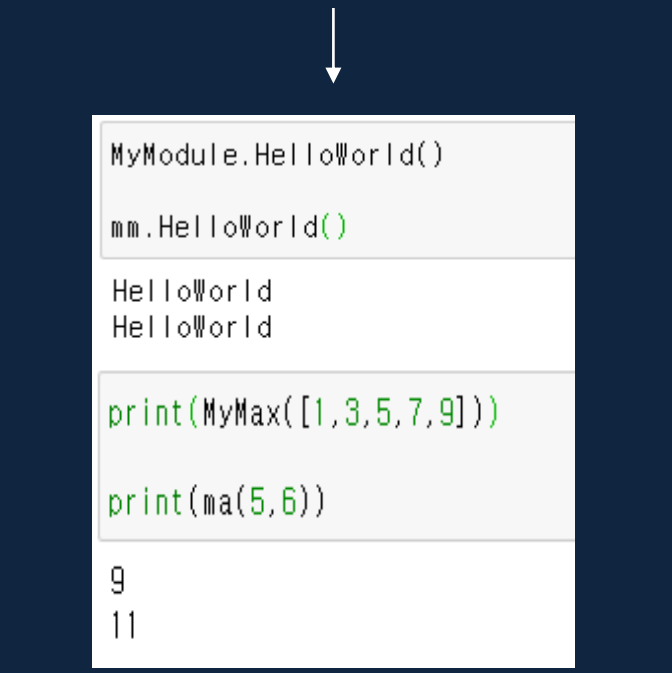
import Module
from Module import 변수명
from Module import 함수명
from Module import 클래스명
from Module import *

3. 모듈의 별명을 설정하는 방법

import Module as mm
from Module import 변수명 as 별명
from Module import 함수명 as 별명
from Module import 클래스명 as 별명



```
import MyModule
import MyModule as mm
from MyModule import MyMax
from MyModule import MyAdd as ma
```



```
MyModule.HelloWorld()

mm.HelloWorld()
```

```
HelloWorld
HelloWorld
```

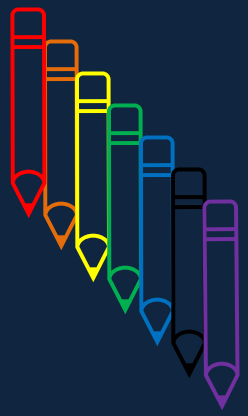
```
print(MyMax([1,3,5,7,9]))

print(ma(5,6))
```

```
9
11
```

5 클래스

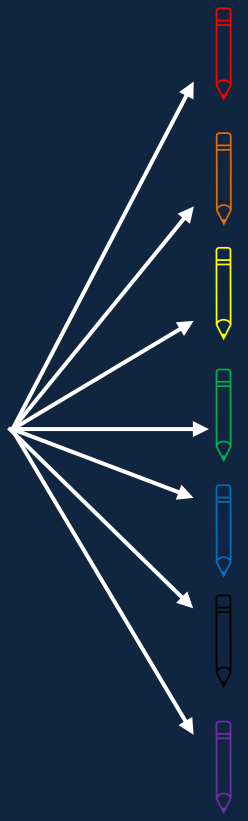
클래스 : 클래스는 인스턴스 생성을 위한 **설계도(틀)**입니다.
인스턴스는 클래스의 변수를 말합니다.

색연필



Pencil = 'black'

색연필을 각각을 함수로 만들어 보세요



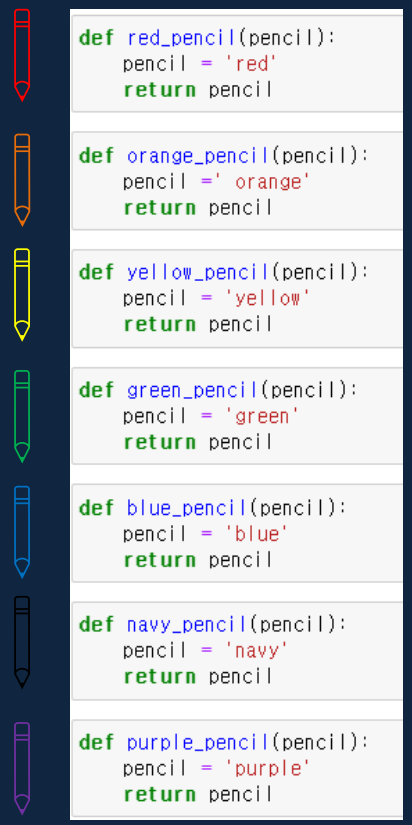
```
def red_pencil(pencil):
    pencil = 'red'
    return pencil

def orange_pencil(pencil):
    pencil =' orange'
    return pencil

def yellow_pencil(pencil):
    pencil = 'yellow'
    return pencil

def green_pencil(pencil):
    pencil = 'green'
    return pencil

def blue_pencil(pencil):
    pencil = 'blue'
    return pencil

def navy_pencil(pencil):
    pencil = 'navy'
    return pencil

def purple_pencil(pencil):
    pencil = 'purple'
    return pencil
```

5 클래스

클래스 : 클래스는 인스턴스 생성을 위한 **설계도(틀)**입니다.
인스턴스는 클래스의 변수를 말합니다.

```
myPen = color_pencil('black')
myPen.pencil
```

```
'black'
```



```
def red_pencil(pencil):
    pencil = 'red'
    return pencil

def orange_pencil(pencil):
    pencil =' orange'
    return pencil

def yellow_pencil(pencil):
    pencil = 'yellow'
    return pencil

def green_pencil(pencil):
    pencil = 'green'
    return pencil

def blue_pencil(pencil):
    pencil = 'blue'
    return pencil

def navy_pencil(pencil):
    pencil = 'navy'
    return pencil

def purple_pencil(pencil):
    pencil = 'purple'
    return pencil
```

색연필의 설계도

```
class color_pencil:

    def __init__(self, color):
        self.pencil = color
```

```
color_pencil('red').pencil
```

```
'red'
```

```
color_pencil('orange').pencil
```

```
'orange'
```

```
color_pencil('green').pencil
```

```
'green'
```

```
color_pencil('blue').pencil
```

```
'blue'
```

```
color_pencil('purple').pencil
```

```
'purple'
```

5 클래스

클래스 : 클래스는 인스턴스 생성을 위한 **설계도(틀)**입니다.
인스턴스는 클래스의 변수를 말합니다.

색연필의 설계도

```
class color_pencil:

    def __init__(self, color):
        self.pencil = color
```

```
myPen = color_pencil('black')
myPen.pencil
```

```
'black'
```

설계도 기능 추가

```
class color_pencil:

    def __init__(self, color):
        self.pencil = color

    def change_color(self, new_color):
        self.pencil = new_color
```

```
myPen = color_pencil('black')
print(myPen.pencil)

myPen.change_color('blue')
print(myPen.pencil)

myPen.change_color('yellow')
print(myPen.pencil)
```

```
black
blue
yellow
```

클래스 구조

```
class color_pencil:

    def __init__(self, color):
        self.pencil = color

    def change_color(self, new_color):
        self.pencil = new_color
```

1. 변수, 속성 -> pencil
2. 생성자 -> __init__ 함수
3. 메소드 -> change_color
4. self -> 클래스의 인스턴스를 나타내는 변수

5 클래스

클래스 : Car

1.변수, 속성 -> color, speed, count

2.생성자 -> __init__(기본값 설정)

3.메소드 -> upSpeed, downSpeed

```python
class Car:
    count = 0  클래스 변수

    def __init__(self, color='silver', speed=0):
        self.color = color
        self.speed = speed
        Car.count += 1

    def upSpeed(self, value):
        self.speed += value
        if self.speed > 150:
            self.speed = 150

    def downSpeed(self, value):
        self.speed -= value
        if self.speed <= 0:
            self.speed = 0
```

```python
myCar1 = Car()
print(myCar1.color)
print(myCar1.speed)
myCar1.upSpeed(30)
print(myCar1.speed)
print(Car.count)
```

```
silver
0
30
1
```

```python
myCar2 = Car('white', speed=10)
print(myCar2.color)
print(myCar2.speed)
myCar2.downSpeed(20)
print(myCar2.speed)
print(Car.count)
```

```
white
10
0
2
```

5 클래스

클래스 : Car

```
class Car:
    count = 0

    def __init__(self, color='silver', speed=0):
        self.color = color
        self.speed = speed
        Car.count += 1

    def upSpeed(self, value):
        self.speed += value
        if self.speed > 150:
            self.speed = 150

    def downSpeed(self, value):
        self.speed -= value
        if self.speed <= 0:
            self.speed = 0
```

```
class Benz:
    count = 0

    def __init__(self, color='black', speed=0):
        self.color = color
        self.speed = speed
        self.AI = 'carAlphago'
        self.RiskFactor = 0
        Car.count += 1

    def upSpeed(self, value):
        self.speed += value
        if self.speed > 150:
            self.speed = 150

    def downSpeed(self, value):
        self.speed -= value
        if self.speed <= 0:
            self.speed = 0

    def aiControl(self, RiskFactor=0):
        self.RiskFactor += RiskFactor
        if self.RiskFactor > 0.33:
            print('조심하세요 !!')
        self.RiskFactor = 0
```

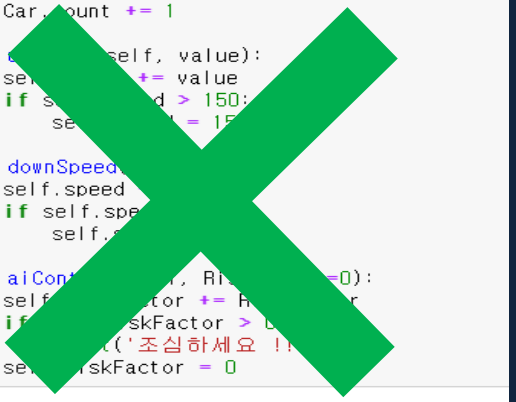
```
myCar = Benz()
print(myCar.color)
print(myCar.speed)
print(myCar.AI)
myCar.aiControl(0.5)
```

```
black
0
carAlphago
조심하세요 !!
```

5 클래스

클래스 : Car

```
class Car:
    count = 0

    def __init__(self, color='silver', speed=0):
        self.color = color
        self.speed = speed
        Car.count += 1

    def upSpeed(self, value):
        self.speed += value
        if self.speed > 150:
            self.speed = 150

    def downSpeed(self, value):
        self.speed -= value
        if self.speed <= 0:
            self.speed = 0
```




```
myCar = Benz()
print(myCar.color)
print(myCar.speed)
print(myCar.AI)
myCar.aiControl(0.5)
```

```
black
0
carAlphago
조심하세요 !!
```

5 클래스

클래스 : 상속

```
class Car:
    count = 0

    def __init__(self, color='silver', speed=0):
        self.color = color
        self.speed = speed
        Car.count += 1

    def upSpeed(self, value):
        self.speed += value
        if self.speed > 150:
            self.speed = 150

    def downSpeed(self, value):
        self.speed -= value
        if self.speed <= 0:
            self.speed = 0
```

super()로
기반 클래스의
__init__ 메서드 호출

```
class Benz(Car):

    def __init__(self, color='black'):
        super().__init__()
        self.color = color
        self.AI = 'carAlphago'
        self.RiskFactor = 0

    def aiControl(self, RiskFactor=0):
        self.RiskFactor += RiskFactor
        if self.RiskFactor > 0.33:
            print('조심하세요 !!')
        self.RiskFactor = 0
```

```
myCar = Benz()
print(myCar.color)
print(myCar.speed)
myCar.upSpeed(50)
print(myCar.speed)
```

```
black
0
50
```

오버라이드

```
class Benz(Car):

    def __init__(self, color='black'):
        super().__init__()
        self.color = color
        self.AI = 'carAlphago'
        self.RiskFactor = 0

    def upSpeed(self, value):
        self.speed += value
        if self.speed > 180:
            self.speed = 180

    def aiControl(self, RiskFactor=0):
        self.RiskFactor += RiskFactor
        if self.RiskFactor > 0.33:
            print('조심하세요 !!')
        self.RiskFactor = 0
```

Break Time

Riew Time

6 예외처리

에러핸들링

우리는 코딩을 하면서 수 많은 에러와 마주치게 됩니다.

SyntaxError, NameError, FileNotFoundError,
TypeError, IndexError, ValueError,
KeyError, AttributeError, ...

5시간 짜리 웹스크래핑을 돌려놓고 다녀왔는데… 나간지 5분만에 에러가 !!!!

예외처리 try – except

try: try문에는 어떤 일을 할 것인지를 넣어줍니다.

except: except문에는 만약 오류가 났을 경우
어떻게 할 것인지를 알려줍니다.

```
# 오류가 나는 것이 정상
2021 / 0
```

```
---------------------------------------
-----
ZeroDivisionError
<ipython-input-40-161f08d394cb> in <
----> 1 2021 / 0

ZeroDivisionError: division by zero
```

```
# 오류가 났음에도 코드가 돌아간 모습.
try:
    2021 / 0
except:
    print('오류발생!')
```

```
오류발생!
```

THANK YOU