

과제 내용

[4주/1차시 학습내용]: 과제 내용, Numpy,
Arrays in Python

Pycharm 설치에 관한 동영상 참조

- 1. Pycharm 설치: 우분투에서 파이참 IDE 설치하기
 - 파이참 설치 영상: <https://youtu.be/CIQppYllGKE>
- 2. anaconda 설치: 우분투에서 아나콘다 인터프리터 설치하기 (아나콘다는 올인원 패키지 인터프리터임)
 - 아나콘다 설치 영상: <https://youtu.be/OeWBmxKzYOQ>
- 3. Pycharm 아나콘다연동: 우분투에서 Pycharm IDE와 아나콘다 인터프리터 연동하기
 - Pycharm과 아나콘다 연동 설명 영상: <https://youtu.be/bgFKPgjxxqs>
- 4. tensorflow 설치: 우분투에서 데이터 해석, 머신러닝을 위한 구글 텐서플로 플랫폼 설치하기
 - 텐서플로우 설치 영상: <https://youtu.be/5ahUc6zB3NI>
- 5. Pycharm 디버깅: 데이터시각화 예제를 통한 Pycharm 디버깅 소개 (콘솔 디버깅과 UI 디버깅의 차이 소개)
 - Pycharm 디버깅 영상: https://youtu.be/4dnSdcGT_mk

과제 내용

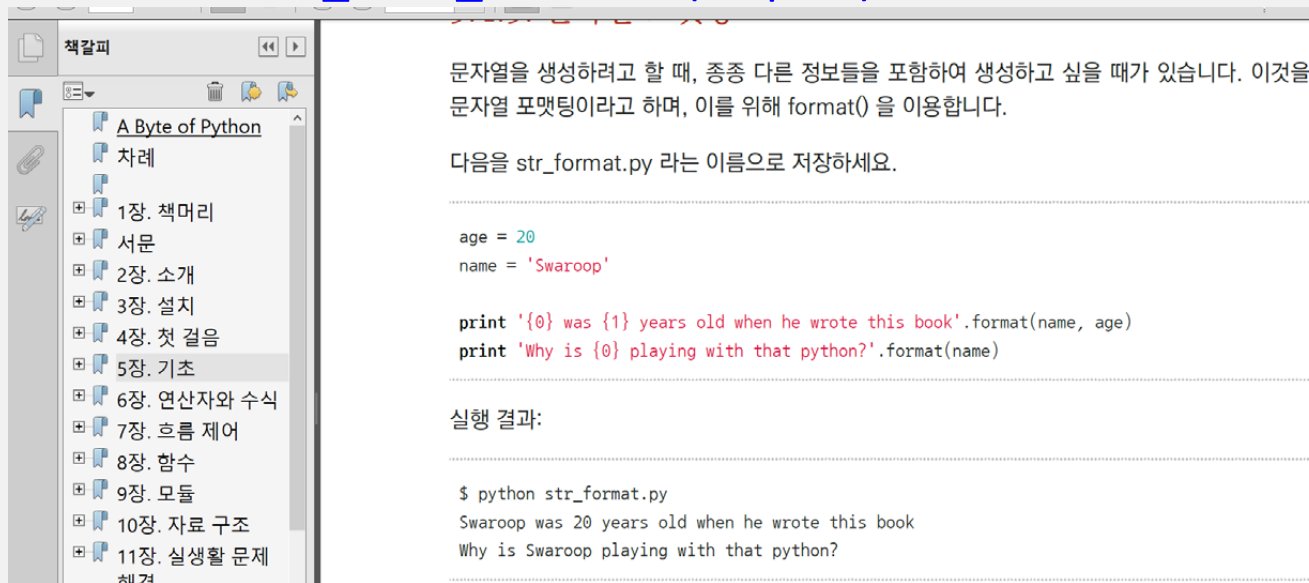
- Homework #1 : 데이터시각화 과제
- Homework #2 (선택 과제) : 기초 파이썬, numpy 과제
 - Homework #2-1 : 파이썬 과제
 - Homework #2-2 : numpy 과제
 - Homework #2-3 : numpy 과제
 - Homework #2-4 : 파이썬 과제
- Homework #3 (선택 과제) : 알고리즘 과제
- 과제 1번은 모두 제출하는 공통 과제
 - 제출일: 4월 4일까지 제출
- 과제 2번과 과제 3번은 두 과제 중 하나를 선택하는 선택형 과제
 - 과제 2번을 선택하면 과제 2-1에서 2-4번까지 제출
 - 과제 3번을 선택하면 과제 3번만 제출
 - 제출일: 5월 9일까지 제출

Homework #1 : 데이터시각화 과제

- 데이터시각화 과제 (데이터사이언스 스쿨)
- <https://datascienceschool.net/view-notebook/d0b1637803754bb083b5722c9f2209d0/>
- 제출일: 2019년 4월 4일 목요일 수업시간 전
- 제출 방법: A4 용지에 출력해서 제출 (1)
- 제출 내용:
 - 데이터 시각화에 관련된 코드를 직접 수행해 보고, 그 결과 그래프를 화면 결과와 소감, 소스 코드와 함께 문서작업하여 가상강의실에 제출 (2)
 - 파일 이름: 학번_이름_#1.doc (hwp 등)

Homework #2 과제

- **Homework #2-1**, Python Textbook 1 : byte_of_python
- Ch 5, ch 6, ch 7, ch 8, ch 9
 - 실행 결과 있는 예제 수행 한 후, 화면 결과와 소감을 소스 코드와 함께 문서작업하여 가상강의실에 5월 9일 제출 (1)
- Due Date: 5월 9일 수업 시작 전에 A4용지로 제출 (2)
 - 파일 이름: 학번_이름_#2-1.doc (hwp 등)



print 다음에는 () 괄호가 필요

5.4.5. 문자열 포매팅

문자열을 생성하려고 할 때 종종 다른 저널의 표기법을 새겨놓고 쓰는 때가 있습니다. 이것은 문자열 포매팅이라고 하며,

다음은 str_format.py 라는

```
age = 20
```

```
name = 'Swaroop'
```

```
print '{0} was {1} years
```

```
print 'Why is {0} playing
```

실행 결과:

```
$ python str_format.py
```

```
Swaroop was 20 years old
```

```
Why is Swaroop playing w
```

IPython

```
Type "copyright", "credits" or "license" for more information.
```

```
IPython 4.2.0 -- An enhanced Interactive Python.
```

```
--> Introduction and overview of IPython's features.
```

```
%quickref --> Quick reference.
```

```
help --> Python's own help system.
```

```
object? --> Details about 'object', use 'object??' for extra details.
```

```
In [1]: age=20
```

```
In [2]: name='Swaroop'
```

```
In [3]: print '{0} was {1} years old when he wrote this book'.format(name, age)
```

```
File "<ipython-input-3-46312128de35>", line 1
```

```
print '{0} was {1} years old when he wrote this book'.format(name, age)
```

```
SyntaxError: invalid syntax
```

```
In [4]: print "{0} was {1} years old when he wrote this book".format(name, age)
```

```
File "<ipython-input-4-ab09675c07b0>", line 1
```

```
print "{0} was {1} years old when he wrote this book".format(name, age)
```

```
SyntaxError: invalid syntax
```

```
In [5]: print ('{0} was {1} years old when he wrote this book'.format(name, age))
```

```
Swaroop was 20 years old when he wrote this book
```

```
In [6]:
```

print 다음에는 () 괄호가 필요

```
In [6]: print(name + ' is ' + str(age) + ' years old')
Swaroop is 20 years old
```

```
.....
# 소수점 이하 셋째 자리까지 부동 소숫점 숫자 표기 (0.333)
print '{0:.3f}'.format(1.0/3)
# 밑줄(_)로 11칸을 채우고 가운데 정렬(^)하기 (__hello__)
print '{0:_^11}'.format('hello')
# 사용자 지정 키워드를 이용해 (Swaroop wrote A Byte of Python) 표기
print '{name} wrote {book}'.format(name='Swaroop',
                                   book='A Byte of Python')
.....
```

실행 결과:

```
.....
0.333
__hello__
Swaroop wrote A Byte of Python
.....
```

Homework #2 과제

- **Homework #2-2**, Python Textbook 2: numpy
- Page 12 까지
 - 실행 결과 있는 예제 수행 한 후, 화면 결과와 소감을 소스 코드와 함께 문서작업하여 가상강의실에 5월 9일 제출 (1)
- Due Date: 5월 9일 수업 시작 전에 A4용지로 제출 (2)
 - 파일 이름: 학번_이름_#2-2.doc (hwp 등)

다음은 str_format.py 라는 이름으로 저장하세요.

```
age = 20
name = 'Swaroop'

print '{0}| was {1} years old when he wrote this book'.format(name, age)
print 'Why is {0} playing with that python?'.format(name)
```

실행 결과:

```
$ python str_format.py
Swaroop was 20 years old when he wrote this book
Why is Swaroop playing with that python?
```


Homework #2 과제

- **Homework #2-3**, Python Homework Textbook 2: numpy
- Page 24 까지
 - 실행 결과 있는 예제 수행 한 후, 화면 결과와 소감을 소스 코드와 함께 문서작업하여 가상강의실에 5월 9일 제출 (1)
- Due Date: 5월 9일 수업 시작 전에 A4용지로 제출 (2)
 - 파일 이름: 학번_이름_#2-3.doc (hwp 등)

다음은 str_format.py 라는 이름으로 저장하세요.

```
age = 20
name = 'Swaroop'

print '{0}| was {1} years old when he wrote this book'.format(name, age)
print 'Why is {0} playing with that python?'.format(name)
```

실행 결과:

```
$ python str_format.py
Swaroop was 20 years old when he wrote this book
Why is Swaroop playing with that python?
```

Homework #2 과제

- **Homework #2-4, Python**
Homework Textbook 1 :
- **byte_of_python**
 - Ch 10, ch 11, ch 12, ch 13, ch 14
 - 실행 결과 있는 예제 수행 한 후, 화면 결과와 소감을 소스 코드와 함께 문서작업하여 가상강의실에 **5월 9일** 제출 (1)
- **Due Date: 5월 9일 수업 시작 전에 A4용지로 제출 (2)**
 - 연장 없음
- **동시에 #2-1, #2-2, #2-3, #2-4를 제출함으로써, 과제 번호를 반드시 파일별로 구별할 것 (zip 사용 권장)**
- **또는 한 개의 파일 이름 안에서 각 과제 #2-1, #2-2, #2-3, #2-4를 정확히 구분해서 제출바람**
 - 파일 이름: 학번_이름_#2-4.doc (hwp 등)

예제 (ds_using_list.py 로 저장하세요):

```
# This is my shopping list
shoplist = ['apple', 'mango', 'carrot', 'banana']

print 'I have', len(shoplist), 'items to purchase.'

print 'These items are:',
for item in shoplist:
    print item,

print '\nI also have to buy rice.'
shoplist.append('rice')
print 'My shopping list is now', shoplist

print 'I will sort my list now'
shoplist.sort()
print 'Sorted shopping list is', shoplist

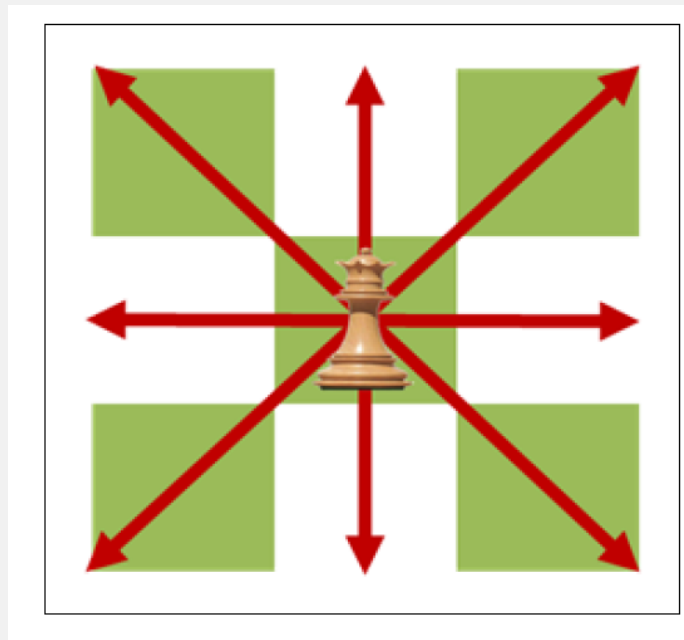
print 'The first item I will buy is', shoplist[0]
olditem = shoplist[0]
del shoplist[0]
print 'I bought the', olditem
print 'My shopping list is now', shoplist
```

실행 결과:

```
$ python ds_using_list.py
I have 4 items to purchase.
These items are: apple mango carrot banana
I also have to buy rice.
My shopping list is now ['apple', 'mango', 'carrot', 'banana', 'rice']
I will sort my list now
Sorted shopping list is ['apple', 'banana', 'carrot', 'mango', 'rice']
The first item I will buy is apple
```

Homework #3 : Queen Problem 알고리즘

- 이 문제는 $n \times n$ 체스 보드판에 n 개의 queen을 서로 공격하지 못하도록 배치하는 방법을 찾아내는 문제이다.
- 아래 그림은 n 이 4일 경우 queen을 서로 공격하지 못하게 배치한 한 예를 나타낸다.

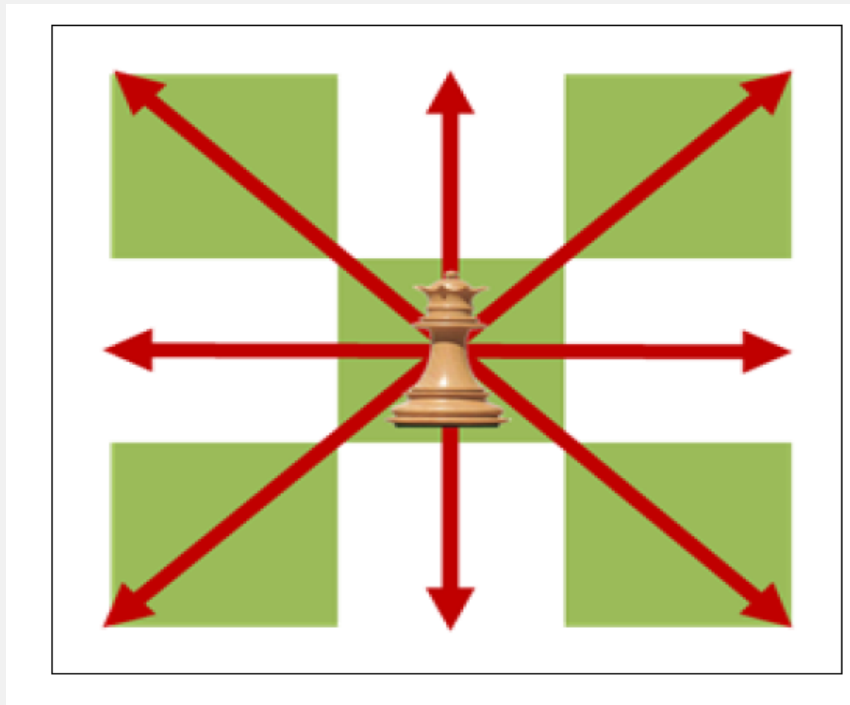


문제 내용

- 일반적으로 프로그래밍 콘테스트에서는 아래와 같이 문제가 제출된다.
- 체스판 크기 및 queen의 수를 나타내는 n 을 입력 받아서 서로 공격하지 못하도록 배치하는 총 방법의 수를 구하는 프로그램을 작성하시오
- 입력
 - 정수 n 이 입력으로 들어온다. ($3 \leq n \leq 9$)
- 출력
 - 서로 다른 총 경우의 수를 출력한다

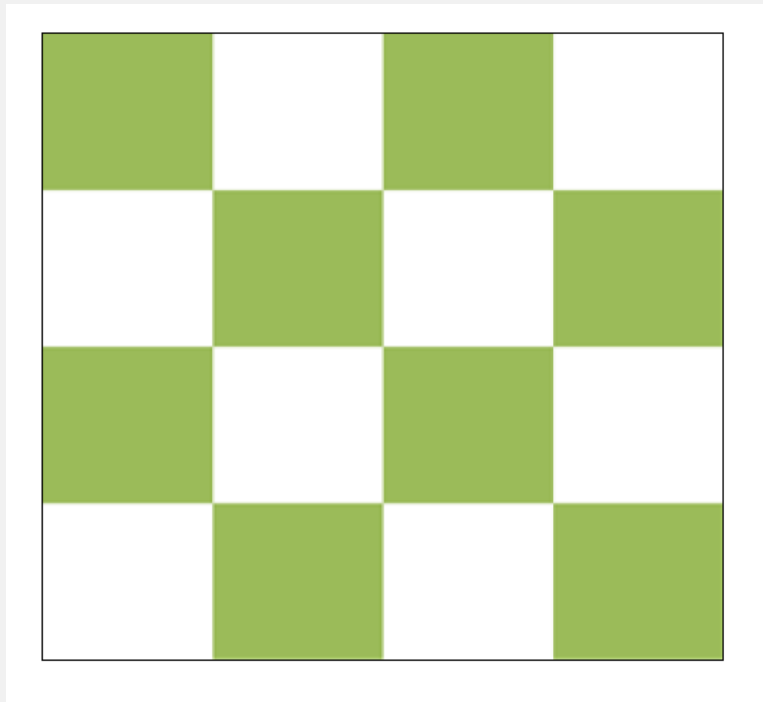
문제 풀이 간단 설명

- 일단 이 문제를 풀기 위해서 퀸이 공격할 수 있는 위치에 대한 생각을 해야 한다.
- 일단 퀸이 공격할 수 있는 루트는 다음과 같다. (8방향으로 체스판의 마지막 칸까지 모두 공격 가능하다.)



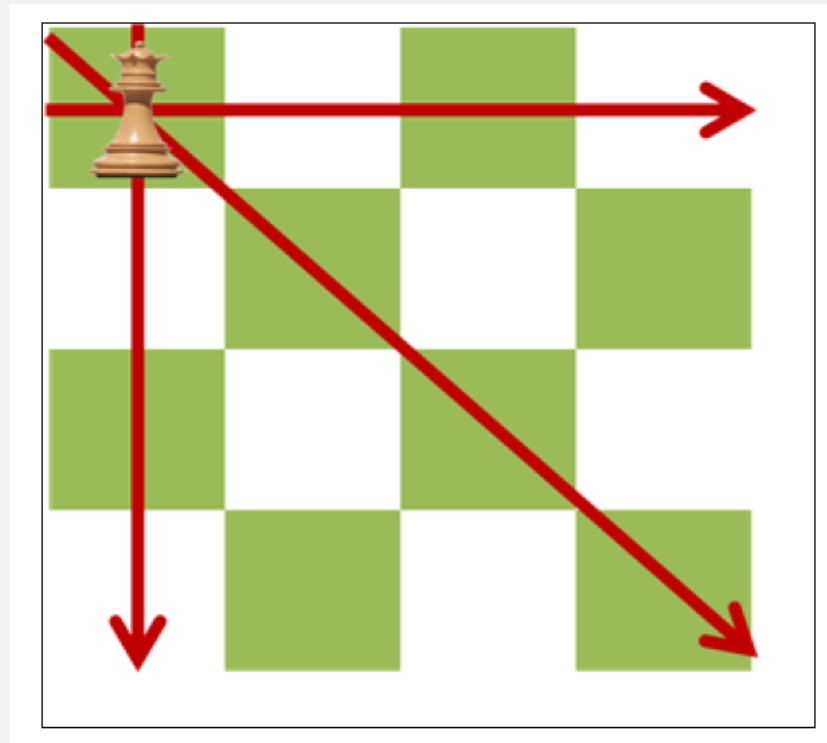
문제 풀이 간단 설명

- 이 문제를 해결하기 위하여 확실한 것은 한 행에 하나 이상의 퀸을 놓을 수 없다는 것이다.
- 4×4 의 체스판을 살펴보자.



문제 풀이 간단 설명

- 위 체스판에서 1행 1열에 하나의 퀸을 배치하면 공격범위는 아래 화살표와 같으며 화살표가 지나가는 칸에는 퀸을 놓을 수 없다.



문제 풀이 간단 설명

- 아래 방법으로 깊이우선 탐색하며 해를 구할 때 마다 카운트 하면 원하는 해를 구할 수 있다.
- 1. 첫 번째 행, 첫 번째 열에 퀸을 놓는다.
- 2. 다음 행에서 가능한 가장 왼쪽 열에 퀸을 놓는다.
- 3. n 번째 열에 더 이상 퀸을 놓을 수 없다면 백트래킹한다.
- 4. 마지막 행에 퀸을 놓으면 하나의 해를 구한 것이다.
- 5. 모든 경우를 조사할 때까지 백트래킹해가며 해들을 구한다.

문제 풀이 간단 설명

- 체스게임에서 Queen은 직선과 대각선을 자유롭게 움직일 수 있는 유닛이다.
- $8 * 8$ 체스판이 있다고 가정하고 이곳에 총 8개의 Queen을 놓아야 한다.
- 단 Queen간에 충돌이 없어야 한다. (각각의 Queen이 좌우, 대각선으로 움직일 때 다른 Queen과 충돌이 없어야 한다.)
- 다음과 같은 예를 들 수 있다. (X는 Queen의 위치를 의미한다.)

```
X 0 0 0 0 0 0 0
0 0 0 0 0 0 X 0
0 0 0 0 X 0 0 0
0 0 0 0 0 0 0 X
0 X 0 0 0 0 0 0
0 0 0 X 0 0 0 0
0 0 0 0 0 X 0 0
0 0 X 0 0 0 0 0
```

```
0 0 0 0 0 0 X 0
0 0 0 X 0 0 0 0
0 X 0 0 0 0 0 0
0 0 0 0 X 0 0 0
0 0 0 0 0 0 0 X
X 0 0 0 0 0 0 0
0 0 X 0 0 0 0 0
0 0 0 0 0 X 0 0
```

#3 과제 구현 방법

- 파이썬 3.5 버전으로 작성된 파이썬 소스 코드를 디버깅하면서 4 * 4 풀이 과정을 설명한다.
 - <https://github.com/SCKIMOSU/Numerical-Analysis/blob/master/queen.py>
 - 4 * 4 풀이 과정에서 깊이 우선 탐색과 백트래킹에 대한 내용을 주로 기술하는 주석 작업을 한다
- 너비 우선탐색이란 무엇인지 인터넷으로 찾아보고 너비우선탐색 코드를 주어진 깊이 우선 탐색 코드 안에서 완성한다.
- 제출 방법: A4 용지에 출력해서 5월 9일 제출 (1)
- 제출 내용:
 - 1. 깊이 우선 탐색의 백트래킹 과정
 - 2. 너비 우선 탐색에 대한 코드와 풀이과정(4 * 4)
 - 깊이우선탐색에 대한 주석작업만으로는 과제 점수가 없다
 - 주어진 깊이 우선탐색 코드 안에서 너비 우선 코드를 작성 바람.
 - 소감, 소스 코드와 함께 문서작업하여 가상강의실에 5월 9일 제출 (2)
 - 파일 이름: 학번_이름_#3.doc (hwp 등)

[4주/1차시 학습내용]: Numpy, Arrays in Python

들여쓰기(Indentation)

- 파이썬에서 공백은 중요한 역할을 합니다.
 - 사실, 한 행의 앞에 붙어있는 공백이 정말 중요합니다.
- 이것을 _들여쓰기_라 부릅니다.
- 한 논리적 명령행의 앞에 붙어있는 공백 (빈 칸 혹은 탭)은 논리적 명령행의 들여쓰기 단계를 의미하며, 이것은 한 명령의 범위를 구분하는 데 사용됩니다.

들여쓰기(Indentation)

- 이것은 같은 들여쓰기 단계에 있는 명령들은 반드시 같은 들여쓰기를 사용해야 함을 의미합니다.
- 이러한 같은 들여쓰기를 사용하고 있는 명령들의 집합을 **블록 (block)** 이라고 부릅니다.

들여쓰기(Indentation)

```
i = 5
# 다음 행에서 오류가 발생합니다! 행 앞에 잘못된 공백이 한 칸 있습니다.
print 'Value is ', i
print 'I repeat, the value is ', i
```

위 예제를 실행하면 다음과 같이 오류가 발생합니다.

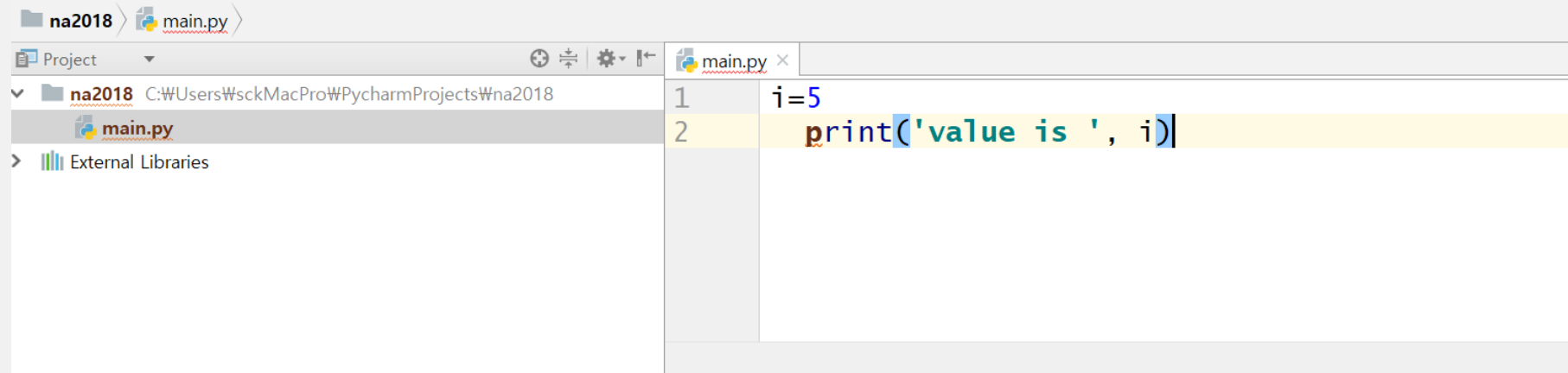
```
File "whitespace.py", line 5
    print 'Value is ', i
    ^
```

IndentationError: unexpected indent

두 번째 행 앞에 공백이 한 칸 있다는 점을 확인
들여쓰기 하는 법: 들여쓰기를 할 때에는 공백 4개를 이용

들여쓰기(Indentation)

File Edit View Navigate Code Refactor Run Tools VCS Window Help



Run main

```
C:\Anaconda3\python.exe C:/Users/sckMacPro/PycharmProjects/na2018/main.py
File "C:/Users/sckMacPro/PycharmProjects/na2018/main.py", line 2
    print('value is ', i)
    ^
IndentationError: unexpected indent

Process finished with exit code 1
```

흐름 제어와 들여 쓰기

```
number = 23
g = input('Give me a number: ')
guess=int(g)

print("Entered Value is ",guess)

if guess == number:
    # New block starts here
    print('Congratulations, you guessed it.')
    print('but you do not win any prizes!')
    # New block ends here

elif guess < number:
    # Another block
    print('No, it is a little higher than that')

else:
    # Another block
    print('No, it is a little lower than that')

print('Done')
# This last statement is always executed,
# after the if statement is executed.
```

Give me a number: 34
Entered Value is 34
No, it is a little lower than that
Done

Process finished with exit code 0

Function (함수)

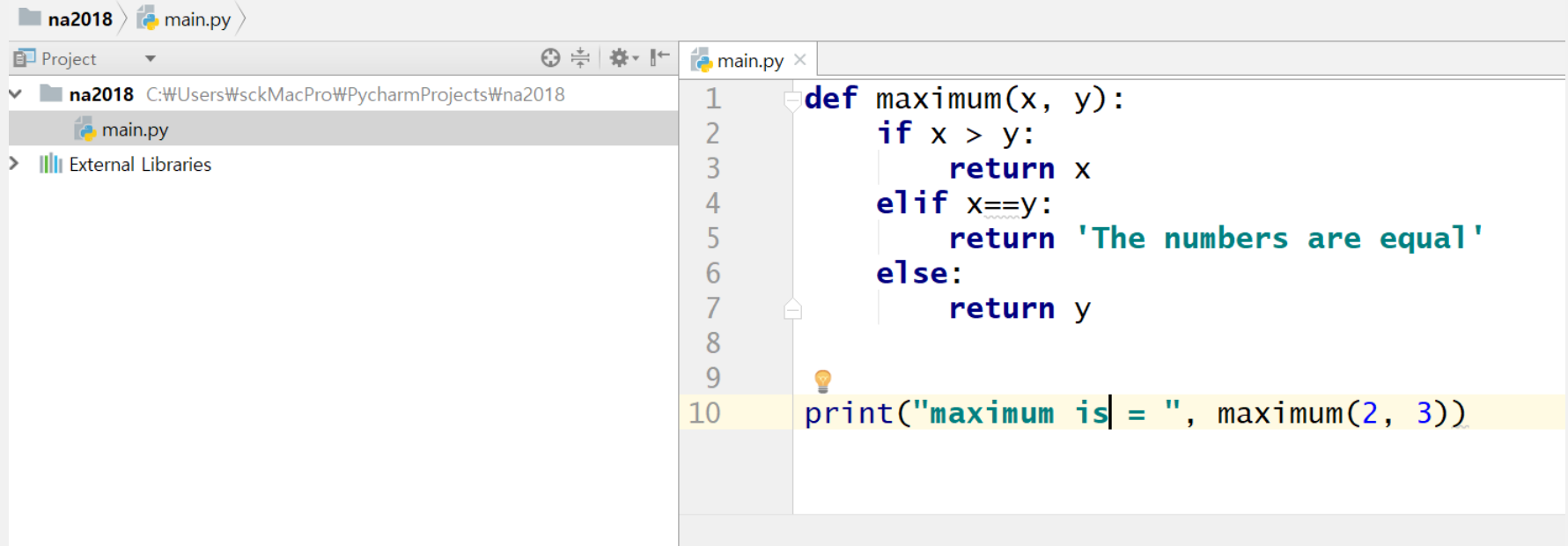
- 재사용 가능한 프로그램의 조각
 - Reusable unit of program
- 특정 블록의 명령어 덩어리를 묶어 이름을 짓고, 그 이름을 프로그램 어디에서건 사용함
- 그 블록에 포함된 명령어들을 몇 번이고 다시 실행할 수 있게 하는 것
- 이를 보고 함수를 ‘호출한다’라고 함
 - 사실 우리는 이미 앞에서 size이나 range 와 같은 많은 내장 함수들을 사용해 왔음.

Function (함수)

- 함수는 def 키워드를 통해 정의됨
 - Use 'def' keyword
- def 뒤에는 함수의 식별자 이름을 입력하고, 괄호로 감싸여진 함수에서 사용될 인자(arguments)의 목록을 입력
- 마지막으로 콜론을 입력하면 함수의 정의가 끝남.
 - `def function_name(arguments):`
- 새로운 블록이 시작되는 다음 줄부터는 이 함수에서 사용될 명령어들을 입력함.

함수와 Return

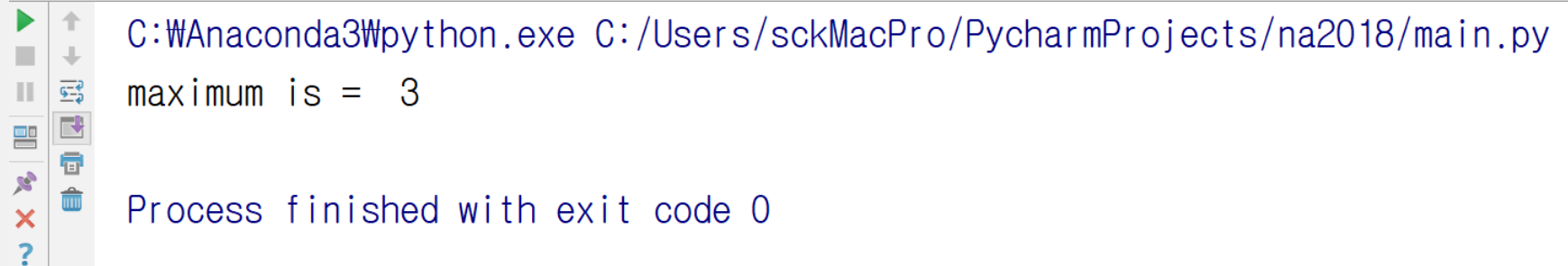
File Edit View Navigate Code Refactor Run Tools VCS Window Help



The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The left sidebar shows the project structure for 'na2018' with a file 'main.py'. The main editor window displays the following Python code:

```
1 def maximum(x, y):
2     if x > y:
3         return x
4     elif x==y:
5         return 'The numbers are equal'
6     else:
7         return y
8
9
10 print("maximum is | = ", maximum(2, 3))
```

Run main



The Run console shows the execution of the script. The command executed is `C:\WAnaconda3\python.exe C:/Users/sckMacPro/PycharmProjects/na2018/main.py`. The output is `maximum is = 3`. The process finished with exit code 0.

Module

- 함수를 통해 여러분의 프로그램 안에서 코드를 재사용하는 방법에 대해서 배워 보았습니다.
- 그러면 여러 함수들을 한꺼번에 불러들여 재사용하는 방법은 없을까요? 네, 이럴 때 모듈을 이용합니다.
- 모듈을 작성하는 데에는 여러 가지 방법이 있습니다만, 가장 간단한 방법은 .py 확장자를 가진 파일을 하나 만들고 그 안에 함수들과 변수들을 정의해 두는 것입니다.

Import Module

- 모든 파이썬 프로그램은 곧 모듈이기 때문입니다.
- 즉 .py 확장자를 가진 이름으로 저장되기만 하면 됩니다.
- 다른 프로그램에서 import 명령을 통해 모듈을 불러와 사용할 수 있습니다.

Module

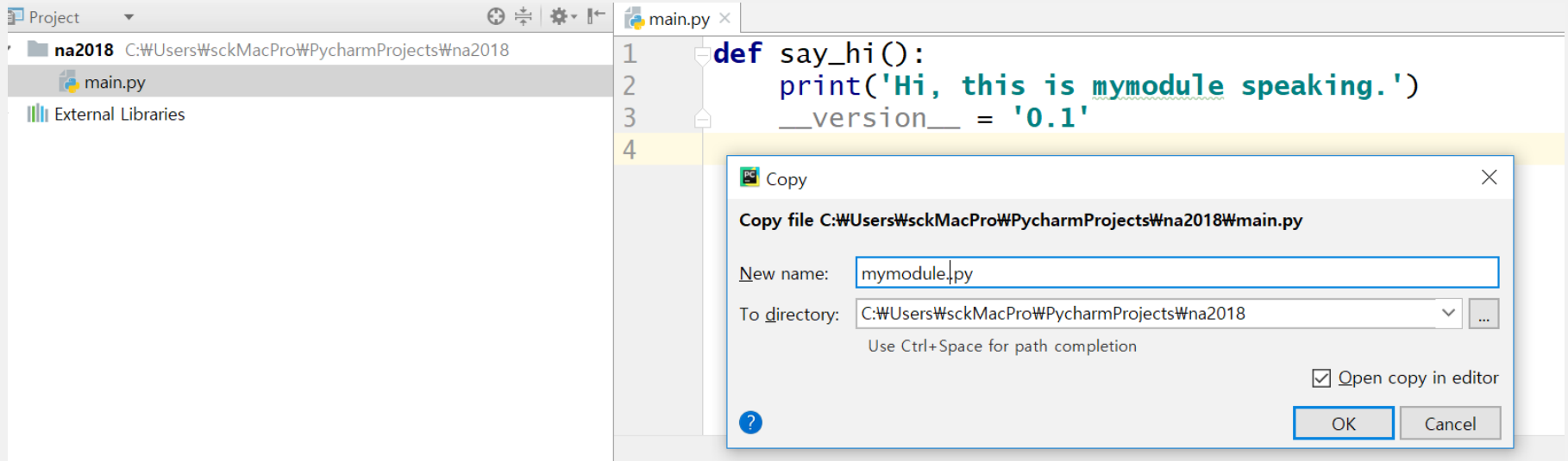
```
def say_hi():  
    print('Hi, this is mymodule speaking.')
```

```
__version__ = '0.1'
```

C:\Users\SCK_RetinaMAC\Anaconda3\python.exe C

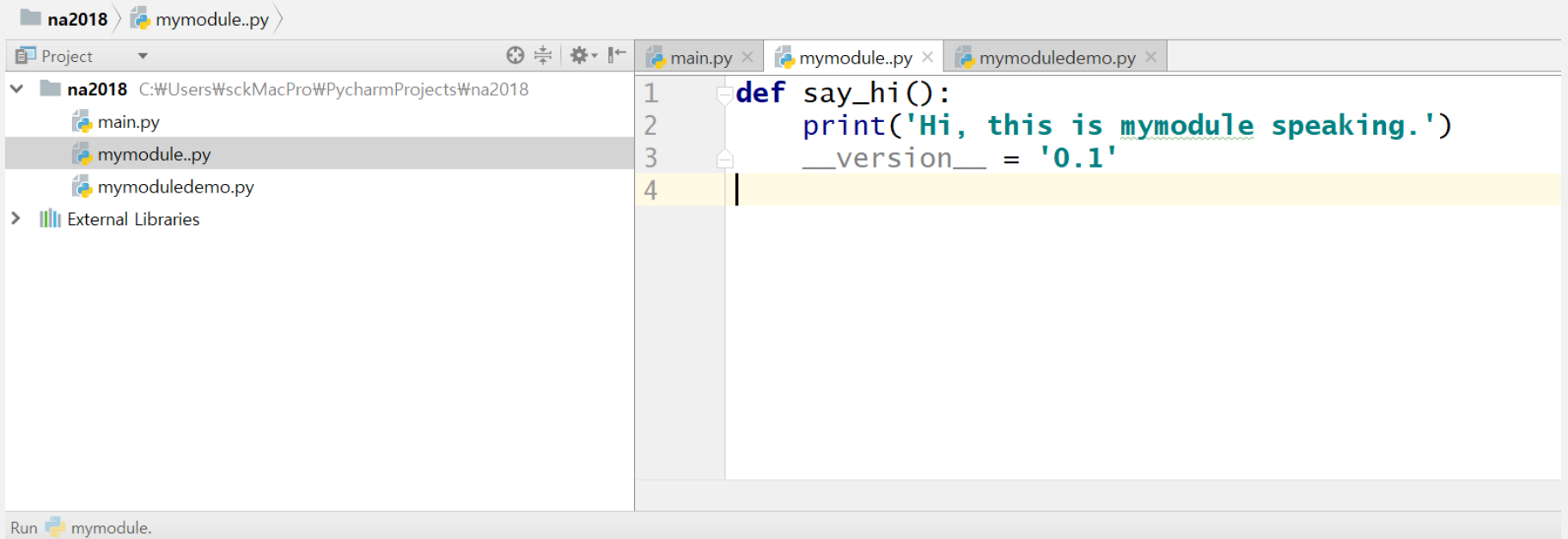
Process finished with exit code 0

- Save as 'mymodule.py'
 - `print()`: 들여쓰기
 - `__version__`: 들여쓰기 하지 않음
- Run 'mymodule.py'

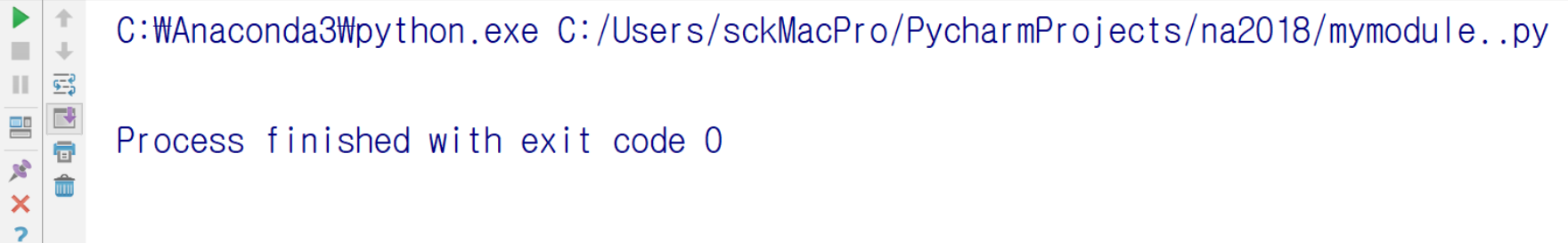


Run 'mymodule.py'

File Edit View Navigate Code Refactor Run Tools VCS Window Help



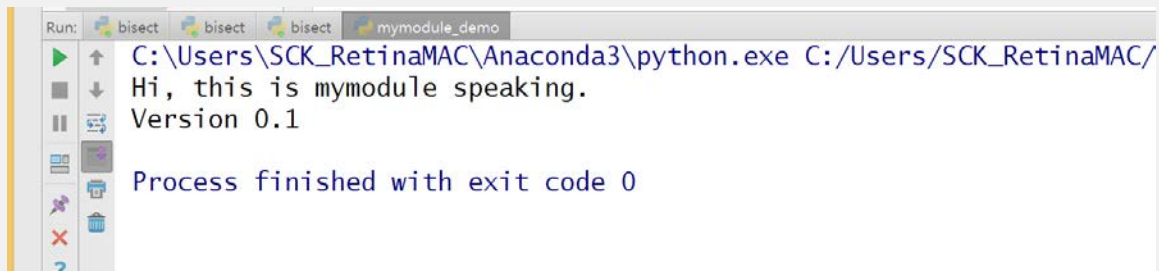
Run mymodule..



Use Module

```
import mymodule  
  
mymodule.say_hi()  
print('Version', mymodule.__version__)
```

- Save as
‘mymoduledemo.py’
- Run
‘mymoduledemo.py’



Ch 2. Python Fundamental (Python Textbook 2: numpy, Arrays in Python)

Prof. Sang-Chul Kim

What is Numerical Python (Numpy)?

- The central feature of NumPy is the array object class.
- Arrays are similar to lists in Python,
 - except that every element of an array must be of the same type,
 - typically a numeric type like float or int.
- Arrays make operations with large amounts of numeric data very fast and are generally much more efficient than lists.

np.array([1,4,5,8], float)

- Here, the function array takes two arguments: the list to be converted into the array and the type of each member of the list.

```
import numpy as np
a=np.array([1,4,5,8], float)

type(a)
```

```
a
Out[10]: array([ 1.,  4.,  5.,  8.])

type(a)
Out[11]: numpy.ndarray
```

np.array([1,4,5,8], float)

```
a[:2]
array([ 1.,  4.])
a
array([ 1.,  4.,  5.,  8.])
a[3]
8.0
a[0]
1.0
a[0]=5.
a
Out[18]: array([ 5.,  4.,  5.,  8.])
```

Two Dimensional Array

```
a = np.array([[1, 2, 3], [4, 5, 6]], float)
```

```
a
```

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

```
a[0,0]
```

```
1.0
```

```
a[0,1]
```

```
2.0
```

Two Dimensional Array

- When standard mathematical operations are used with arrays, they are applied on an element-by-element basis.

```
a = np.array([1,2,3], float)
a
array([ 1.,  2.,  3.])
b = np.array([5,2,6], float)
b
array([ 5.,  2.,  6.])
a+b
array([ 6.,  4.,  9.])

a-b
array([-4.,  0., -3.])
a*b
array([ 5.,  4., 18.])
b/a
array([ 5.,  1.,  2.])
a%b
array([ 1.,  0.,  3.])
b**a
array([ 5.,  4., 216.])
```

Two Dimensional Array

- For two-dimensional arrays, multiplication remains elementwise and does *not* correspond to matrix multiplication.

```
a = np.array([[1,2], [3,4]], float)
```

```
a
```

```
array([[ 1.,  2.],  
       [ 3.,  4.]])
```

```
b = np.array([[2,0], [1,3]], float)
```

```
b
```

```
array([[ 2.,  0.],  
       [ 1.,  3.]])
```

```
a*b
```

```
array([[ 2.,  0.],  
       [ 3., 12.]])
```

Array iteration

- Array iteration

```
array([1, 4, 5])  
for x in a:  
    print(x)
```

```
1  
4  
5
```


Minimum and maximum values

- find the minimum and maximum element values

```
a = np.array([2, 1, 9], float)
a.min()
1.0
a.max()
9.0
```

Minimum and maximum values

- The `argmin` and `argmax` functions return the array indices of the minimum and maximum values:

```
a = np.array([2, 1, 9], float)
a.argmin()
1
a.argmax()
2
```

Unique elements

- Unique elements can be extracted from an array:

```
a = np.array([1, 1, 4, 5, 5, 5, 7], float)
np.unique(a)
array([ 1., 4., 5., 7.])
```

Comparison

- Arrays can be compared to single values using broadcasting:

```
a = np.array([1, 3, 0], float)
a>2
array([False,  True, False], dtype=bool)
```

Nonzero()

- Nonzero()

```
a = np.array([[0, 1], [3, 0]], float)
a
array([[ 0.,  1.],
       [ 3.,  0.]])
In[572]: a.nonzero()
(array([0, 1], dtype=int64), array([1, 0], dtype=int64))
```

NaN ("not a number") or finite

- NaN ("not a number") or finite

```
a = np.array([1, np.NaN, np.Inf], float)

array([ 1., nan, inf])
np.isnan(a)
array([False,  True, False], dtype=bool)
np.isfinite(a)
array([ True, False, False], dtype=bool)
```