Python Memo : 파이썬 관련 메모

Python code written while studying

- Python 코드 작성 중 자주 검색해보는 것들 또는 까먹지 말아야 할 것 메모했음.
- 전처리, 사용자 함수, 자주 사용하는 코드, 이론 등

이론

튜플

- 튜플은 리스트 [a,b, ...]와 비슷하다. 차이점은 튜플은 요소를 추가, 변경, 삭제하는 처리가 불가능하고 리스트는 가능하다는 것이다. 튜플이 메모리를 적게 사용하기 때문에 요소를 수정할 필요가 없으면 튜플을 사용하는 것이 효율적이다.

전처리

- 1.공백 제거한 뒤 변수명 일괄 변경

```
' '.join(k).split()

ex)
shares = pd.read_csv("OnlineNewsPopularity.csv")
k=shares.columns.values.tolist()
k=' '.join(k).split()
shares.columns=' '.join(shares.columns.values).split()
```

- 2.열 선택
  - a에 해당하는 열 가져오기

```
a=["a","b","c"]
shares[a] #a,b,c열 가져와짐.
```

  - a가 아닌 열 가져오기

```
shares[shares.columns.difference(a)]
```

  - 열 붙히기(cbind)

```
df_c = pd.concat([k1, k2], axis=1)
```

- 3. 반올림, 올림, 내림
  - 소수점을 n번째 까지만 표현하고 반올림을 하고싶을때, round 함수를 사용
  - default는 소수점 첫번째 자리에서 반올림해서 정수로 표현된다.
    - ex)

```
n = 7/15 -> n = 0.4666666666666667
>>> round(n,2)
0.47
>>> round(n,4)
0.4667
>>> round(n)
0
>>> type(round(n))
<class 'int'>
```

  - 물론 정수도 반올림 가능
    - ex)

```
>>> round(12345,-1)
12340
>>> round(12345,-2)
12300
```

  - math.ceil(i) : 올림, math.floor(i) : 내림, math.trunc(i) : 버림
    - ex)

```
>>> import math
>>> math.ceil(12.2)
13
>>> math.ceil(12.6)
13
>>> math.floor(12.2)
12
>>> math.floor(12.6)
12
>>> math.trunc(12.2)
12
>>> math.trunc(12.6)
12
```

- 파이썬 제곱(**), 나머지(%)
  - ex1 : `print(2 ** 10)` # 2의 10제곱 => 1024
  - ex2 : `print(10 % 2)` # 10을 2로 나누었을 때의 나머지 => 0

- 문자 숫자 type 변환
  - 문자열로 변환 : `str(test)` or `repr(Test)`
  - 숫자로 변환 : `int(test)`

함수

- del() : 변수 삭제
- len() : 문자열, 리스트 길이 반환.
- iloc() : DataFrame 함수로 특정 행의 데이터를 가져온다.
- 고유 값(unique 값) 출력
  - data -> `list(set(data))`
- int() : 정수로 형식 변환.

```
int(0.5) = 1       int(1.3)=1
```

Numpy Function

- np.random.rand() : Numpy의 random 모듈은 랜덤 값 생성을 위한 rand() 함수를 제공함. 0에서 1사이의 값을 생성하여 반환함. randint(low, high=None) 함수는 high를 넣지 않은 경우에는 0에서 low사이의 정수를 랜덤으로 생성하고 high를 넣은 경우 low에서 high 사이의 정수를 생성함.

- np.argmax() : argmax(array) 함수는 입력으로 들어온 array에서 가장 큰 값의 위치를 반환한다.

```
예를 들어 array = [3,5,7,0,-3] dlaus 가장 큰 값은 7이고 그 위치는 2이다.
np.argmax(array) => 2
```

- Numpy의 reshpe() 함수
  - ndarray를 reshape() 함수를 써서 다른 차원으로 변환할 수 있다. 예를 들어 1차원 배열인 [1,2,3,4,5,6]이 있을 때 이 배열의 shape(즉,모양)은 (6,)이다. 이를 (3,2)로 만들면 [[1,2],[3,4],[5,6]]이 된다. 이 때 유의할 점은 배열의 총 크기는 변하지 않아야 한다. 이 예에서는 변환한 후의 크기가 3x2=6 이므로 1차원 배열일 때와 같다.

- Numpy의 arrange() 함수 `np.arrange(3) -> 배열 [0,1,2]를 반환. 시작값(start), 종료값(stop), 값 사이 간격(step) 지정 가능.`

- Numpy의 zeros() 함수
  - 인자로 배열의 형태인 shape을 받아서 0으로 구성된 NumPy 배열을 반환한다.

```
ex) np.zeros(3) -> [0,0,0] 을 반환,
   np.zeros((2,2)) -> [ [0,0], [0,0]] 을 반환.
```

  - 주의할 점은 다차원 배열의 경우 그 형태를 튜플로 넘겨줘야 한다는 점이다.

- Numpy의 full() 함수
  - 첫 번째 인자는 배열의 형태인 shape을 받는데 사용. 두 번째 인자는 입력된 값으로 채워진 NumPy 배열을 반환한다.

```
ex) np.full(3,1) -> [1,1,1]          np.full((2,2), 0.5) -> [
[0.5,0.5], [0.5,0.5]] 를 반환.
```

  - 주의할 점은 다차원 배열의 경우 그 형태를 튜플로 넘겨줘야 한다.
- Numpy의 타입 확인 및 변경
  - 타입 확인 : `data.dtype()`
  - 타입 변경 : `data = data.astype(np.int32)`
- Numpy 배열 합치기 & append로 배열 추가하기
  - `np.concatenate((data1, data2), axis = None)`
  - `np.concatenate((data1, data2.T), axis = 0)`
  - `np.concatenate((data1, data2), axis = 0)`
  - `np.append(a1, a2)`
- Numpy 배열 길이 확인 방법
  - `data.shape` or `len(data)`
- Numpy의 around() 함수
  - 반올림 함수
    - `np.round(data, 2)` or `np.around(data, 2)` 사용
- Numpy 배열에서 원소(value) 몇 개씩 존재하는지 count(R에서의 table)

```
# example 1
from collections import Counter
Counter(jaffe_Y)

# example 2
result = Counter(jaffe_Y)
print(result)
for key in result:
    print(key, result[key])

# example 3
import numpy as np
x = np.array([1,1,1,2,2,2,5,25,1,1])
y = np.bincount(x)
ii = np.nonzero(y)[0]
Zip(ii,y[ii]) # [(1, 5), (2, 3), (5, 1), (25, 1)]
```

```
# example 4 : 키 없이 카운트 한 값만 알아내고 싶을 때 사용
result = Counter(jaffe_Y).values()
print(result)
```

- Numpy 배열에서 어떤 값이 중복됐는지 확인하는 방법

```
# 1. 중복된 값만 추출
from collections import Counter
a = np.array([1, 2, 1, 3, 3, 3, 0])
[item for item, count in Counter(a).items() if count > 1]

# 2. 중복된 값의 인덱스 추출
u, c = np.unique(a, return_counts=True)
dup = u[c > 1]

# 3. 유니크한 값만 가져오기
np.unique(data)
```

  - 2D, 3D... array에서 중복된 값 제거하고 가져오는 방법

```
L = [np.array([1, 2, 3]), np.array([]), np.array([3, 2, 1]),
    np.array([1, 3, 2]), np.array([1, 2, 3]), np.array([1, 3, 2]),
    np.array([]), np.array([]), np.array([1, 2, 3]),np.array([7,7,7])]
L
`np.unique(L, axis=0)`
```

    - 참고 : https://docs.scipy.org/doc/numpy-1.14.0/reference/generated/numpy.unique.html
  - 응용 : A 변수에서 중복된 데이터 제외한 인덱스를 가져와 B 변수 select
    - `u, indices = np.unique(fer2013_X, axis=0, return_index = True)` -> `fer2013_label = fer2013_label[indices]`

- python에서 문자를 출력할 때 raw로 출력하기

  - `print('문자를 출력할 때 \t \n 같은 이스케이프 문자가 미출력될 때 사용한다.')`
  - `print(r'문자를 출력할 때 \t \n 같은 이스케이프 문자가 미출력될 때 사용한다.')` => raw 문자 출력

- data suffle(섞기)

  - `from sklearn.utils import shuffle`
  - x, y에 각각 셔플된 값 넣기 : `x, y = shuffle(img_data, Y, random_state = 202006)`
  - `y = shuffle(Y, random_state = 202006)`

- Matplotlib의 subplots() 함수

  - 여러 Axes로 구성된 Figure를 생성할 때 효과적이다. 인자로 들어가는 nrows는 행 개수,ncols는 열 개수를 의미한다. nrow가 2이고 ncol이 3이라면 2x3 Axes, 즉 6개의 Axes로 구성된 Figure가 생성

된다.

- zip() 함수
  - 파이썬 내장 함수로 두 개의 배열에서 같은 인덱스의 요소를 순서대로 묶어준다.

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

- Matplotlib()의 axvline() 함수
  - x축 위치에서 세로로 선을 긋는 함수이다. 이선의 색깔은 color 인자로, 선의 투명도를 alpha로 정해줄 수 있다.
- Matplotlib()의 plot() 함수
  - x축의 데이터, y축의 데이터, 차트의 스타일을 인자로 받는다. x축 데이터와 y축 데이터의 크기가 같아야 한다. 스타일은 실선, 점선 등의 선 형태와 선의 색깔을 축약적으로 정의한다.
  - 표시할 다양한 모양과 색깔을 좇합할 수 있다.

```
ex1) axes[1].plot(x, num_stocks,'-k')에서 '-k' 는 검정색 실선을 의미함.
ex2) '-'은 선, '.'은 점, 'r'은 빨간색, 'b'는 파란색을 의미.
이를 조합한 '-r','-b' 등은 빨간 선, 파란 선을 의미.
```

- Matplotlib()의 fill_between() 함수
  - x축 배열과 두 개의 y축 배열을 입력으로 받는다. 두 개의 y축 배열의 같은 인덱스 위치의 값 사이에 색을 치란다. where 옵션으로 색을 칠할 조건을 추가할 수 있고 facecolor 옵션으로 칠할 색을 지정하고 alpha 옵션으로 투명도를 조정할 수 있다.
- Matplotlib()의 tight_layout() 함수
  - Fiqure의 크기에 알맞게 내부 차트들의 크기를 조정해 준다.
- os.path 모듈
  - os.path.join(path) 함수는 path로 경로를 설정한다.
  - os.path.isdir(path) 함수는 path가 존재하고 폴더인지 확인하는 함수이다.
  - os.path.isfile(path) 함수는 path가 존재하는 파일인지 확인하는 함수이다.
  - os.mkdirs(path) 함수는 path에 포함된 폴더들이 없을 경우에 생성해 주는 함수이다.

```
path가 '/a/b/c'이고 현재 '/a'라는 경로만 존재한다면 '/a'폴더 하위에 'b'폴더를 생성하고
'b'폴더 하위에 'c'폴더를 생성하여 최종적으로 '/a/b/c' 경로가 존재하도록 만든다.
```

  - 현재 위치 가져오기

```
import os
print(os.getcwd())
print(os.path.dirname(os.path.realpath(__file__)) )
```

- 현재 디렉토리에 있는 모든 파일(디렉토리) 리스트 가져오기

```
path = "./"
file_list = os.listdir(path)
```

  - ".py" 확장자를 가진 파일의 리스트 가져오기

```
# Use os.listdir()
path = "./"
file_list = os.listdir(path)
file_list_py = [file for file in file_list if
file.endswith(".py")]

print ("file_list_py: {}".format(file_list_py))
```

```
# Use glob
import glob
path = "./*"
file_list = glob.glob(path)
file_list_py = [file for file in file_list if
file.endswith(".py")]

print ("file_list_py: {}".format(file_list_py))
```

  - "os.listdir"과 "glob.glob"의 차이점

    - `os.listdir`을 사용한 경우는 해당 디렉토리의 파일명만 가져왔지만, `glob`로 파일 리스트를 가져올 경우에는 검색시 사용했던 경로명까지 전부 가져온다. 예를들어 현재 경로가 아닌 특정 경로의 파일을 찾는다면 다음과 같은 차이가 보일것이다.

- Pandas의 replace() 함수

  - 특정 값을 바꿀 때 사용한다. ffill 메서드를 이용하면 특정 값을 이전의 값으로 변경하고, bfill 메서드를 이용하면 특정 값을 이후의 값으로 값으로 변경할 때 사용한다.

```
series=[1,3,0,5,7] 일 때,
series.replace(to_replace=0, method='ffill') => 결과 : [1,3,3,5,7]
series.replace(to_replace=0, method='bfill') => 결과 : [1,3,5,5,7]
```

- 자주 쓰는 Pandas 함수
  - 열이름 가져오는 가장 간단한 코드
    - `df.columns.values.tolist()` or `list(df.columns)`
  - 행과 열 개수 가져오기
    - 행 : `df.shape[0]`, `len(df)` , `len(df.index)`
    - 열 : `df.shape[1]`, `len(df.columns)`
  - 변수 타입 확인
    - `type()`
  - 특정 열 제외하고 가져오기
    - `colsum=[1,3,9]` or `colsum=arrange(0,40)`
    - 이후 `df.iloc[:, ~df.columns.isin(df.columns[colsnum])]`
  - 열 선택
    - `ind = ['a', 'b', 'c']` -> `data = data[ind]`
  - 변경하고 싶은 열 이름 변경
    - df = `df.rename(columns={'oldName1': 'newName1', 'oldName2': 'newName2'})`
    - rename the existing DataFrame (rather than creating a copy)
      - `df.rename(columns={'oldName1': 'newName1', 'oldName2': 'newName2'}, inplace=True)` or `df.rename({'oldName1': 'newName1', 'oldName2': 'newName2'}, inplace=True,axis=1)`
  - set 자료형을 이용하면 리스트의 차집합, 합집합, 교집합을 구할 수 있다.
    - ```
      s1 = set([1, 2, 3, 4, 5, 6])
      s2 = set([4, 5, 6, 7, 8, 9])
      ```
      - 합집합 : `s1 | s2` or `s1.union(s2)`
      - 차집합 : `s1 - s2` or `s1.difference(s2)`
      - 교집합 : `s1 & s2` or `s1.intersection(s2)`
    - 마지막에 다시 set 자료형을 리스트로 만들어 주기 -> `list('set자료형이름')`
  - 특정 집합, 요소에 해당(포함)하는 데이터만 추출
    - `all_data[all_data.image_id.isin(train_idx)]` : all_data 데이터에서 image_id 열 중에 train_idx 리스트에 포함된 요소만 추출.
  - data.frame(데이터 프레임)에서 조건에 해당하는 행만 추출
    - df에서 image_id가 P_002893.jpg인 행만 추출 : `df[df['image_id']=='P_002893.jpg']`
- 오름차순, 내림차순
  - `변수이름.sort()` or `sorted(변수이름,key=,reverse=)` 둘중 하나 사용
    - ```
      a = [1, 10, 5, 7, 6]
      a.sort() # = sort(a)
      -> [1, 5, 6, 7, 10]
      a = [1, 10, 5, 7, 6]
      a.sort(reverse=True) # = sort(a, reverse=True)
      -> [10, 7, 6, 5, 1]
      ```

- 연산자
  - `* , / , % , //` : 곱하기, 나누기, 나머지, 몫
  - `divmod()` 함수 사용하면 몫과 나머지를 튜플 형태로 리턴함.
    - `divmod(7,3)` => (2,1)

- 리스트에서 조건에 해당하는 값만 추출하기

```
# sample
originalList = [1, 2, 44, 52, 61, 7, 28, 92, 10

# 1. filter 함수
- 형식: filter(함수_이름, iterable)
- filter 함수의 인자로 들어가는 함수의 리턴 타입은 bool형이여야 한다. 람다를 쓰면 편하다.
- 리턴 타입: iterator (filter object)
- code :
filteredList = list(filter(lambda x: x%2==0, originalList))

# 2. Nested List Comprehension
filteredList = [x for x in originalList if x%2==0]
```

- image(이미지)에서 원하는 부분 cropping(잘라내기)

```
def im_trim (img): #함수로 만든다
for (x,y,w,h) in kkk:
    img_trim = img[y:y+h, x:x+w]
    cv2.imwrite('test_cropping_data/' + 'test_image.jpg',img_trim)
return img_trim
```

문자열 관련 처리 및 함수

- rjust() 함수
  - 문자열을 자리수에 맞게 오른쪽으로 정렬해 주는 함수이다.

```
ex1) "1".rjust(5) -> '    1' 이 된다.
    앞에 빈칸 4자리를 채워주고 1을 붙여서 5자리 문자열이 되는 것이다.
ex2) "1".rjust(5,'0') -> '00001'이 된다.
    두 번째 인자로 빈칸 대신 채워줄 문자를 지정해줄 수도 있다.
```

- ljust() 함수
  - rjust()와 비슷한 함수

    - 이 함수는 기존 문자열 앞에 빈칸 또는 특정 문자를 채워준다. -> 이렇게 기존 문자 앞 또는 뒤에 어떠한 문자를 채워주는 것을 패딩(padding)이라 한다.
- 정규식
    - 문자열에서 숫자 추출하기

```
1. 정규식을 사용하기 위해서 re모듈을 사용함.
2. 숫자로 뽑아낼 문자열을 변수로 지정
3. "\d+ " : 정규식에서 숫자를 나타냄[\는 달러표시로 기입]
4. findall()함수를 사용해서 부합하는 모든 문자열을 리스트로 리턴함

import re

sentence = "730만원어치를 인터넷에서 170203에 구매했다"
num = re.findall("/d+", sentence)
print(num)
```

- 문자열에 값을 넣어주는 방법 `%d, %s, %f` and `format()`
    - `%d`는 정수, `%s`는 문자열, `%f`는 실수를 입력하는 자리이다.

```
ex1) ['Hello %s' % 'World!']는 ['Hello World']가 된다.
ex2) ['%d / %d = %.2f' % (10, 3, 10/3)] 이 [ '10 / 3 = 3.33' ]으로 된다.
```

    - 다른 방법으로 `format()` 함수가 있다. 이 함수에는 값을 넣을 자리를 {}로 표현한다.

```
ex1) ['Hello {}'.format('World!')] -> ['Hello World!']
ex2) ['Hello {tail}'.format(tail='World!')] -> ['Hello World!']
```

- 문자열 공백 제거
    - 1. 양쪽 끝의 공백 문자를 제거하는 경우 - str.strip:

```
sentence = ' hello  apple'
sentence.strip() #'hello  apple'
```

    - 2. 모든 스페이스(' ')를 제거하는 경우 - str.replace():

```
sentence = ' hello  apple'
sentence.replace(" ", "") #'helloapple'
```

    - 3. 모든 공백 문자를 제거하는 경우 - re모듈:

```
import re
pattern = re.compile(r'\s+')
sentence = re.sub(pattern, '', sentence)
# 또는
sentence = ' hello  apple\t\t\n\n\n\n'
''.join(sentence.split()) #'helloapple'
```

- 4. 중복된 공백 문자를 제거하는 경우(예_ 스페이스바가 2개) - str.split()과 str.join():

```
sentence = ' hello  apple'
" ".join(sentence.split()) # 'hello apple'
```

- 5. list안 특정 문자열을 포함하는 단어 찾기
  - sample

```
match_list = list()
for word in word_list:
    if search in word:
        match_list.append(word)
match_list
```

  - sample 2 : 리스트 내에 특정 문자있는지 확인하기
    - `'My_Data' in data_list` : data_list라는 list안에 'My_Data' 문자가 들어 있으면 TRUE, 없으면 False를 내뱉는다.
    - 바내는 `'My_Data' not in data_list` : in 대신 `not in`을 사용하면 된다.
  - in을 이용한 특정 문자열 검색

```
match_list = list()
for word in word_list:
    if search in word:
        match_list.append(word)
match_list
```

  - in을 이용한 comprehension 표현

```
[word for word in word_list if search in word]
```

  - 리스트에서 문자열 인덱스 찾기
    - `["hello","python","world","!"].index("world")`

- python list(리스트)에서 특수문자 및 제거
  - 모든 특수문자 제거 : `[re.sub('[^a-zA-Z0-9]+', '', _) for _ in list_name]`
  - `(`와 `)`만 제거 : `[re.sub('[()]', '', _) for _ in test]`

    - 특정 문자 제거 : `[re.sub('[()]+', '', _) for _ in list_name]`
- python list or text에서 split하기
    - `df.iloc[1,1].split(',')`

크롤링(Crawling)

- 셀레니움(Selenium)에서 화면 스크린샷(Screenshot) 하기

```
  # 현재 페이지 스크린샷
  screenshot_name = "my_screenshot_name.png"
  driver.save_screenshot(screenshot_name)

  # 전체 페이지 스크린샷(임시 코드)
  # 참고 페이지 : https://pypi.org/project/Selenium-Screenshot/
  pip install Selenium-Screenshot

  from Screenshot import Screenshot_Clipping
  from selenium import webdriver

  ob=Screenshot_Clipping.Screenshot()
  driver = webdriver.Chrome()
  url =
"https://github.com/sam4u3/Selenium_Screenshot/tree/master/test"
  driver.get(url)
  img_url=ob.full_Screenshot(driver, save_path=r'.',
image_name='Myimage.png')
      # img_url=ob.full_Screenshot(driver, save_path='c://...경로',
image_name='Myimage.png')
  print(img_url)
  driver.close()
```

ETC

- `__init__()`
    - 파이썬 클래스의 생성사 **init**()은 클래스의 객체가 생성될 때 자동으로 호출되는 함수임. 보통 이 함수에서 입력으로 받은 값들을 객체 내의 변수로 할당해 준다.
- 파이썬에서는 리스트에 곱하기를 하면 똑같은 리스트가 반복된다.

```
ex) [1,2,3]*3 => [1,2,3,1,2,3,1,2,3]
```

- `if else구문` 축소 가능

```
if x >= 0:
    x += 1
else:
    x -= 1

=> x += if x>= 0 else -1
```

- 파이참 단축키

```
1) 주석제거 ctrl + /
2) 실행 ctrl + shift + F10
3) 한줄 실행 Alt + Shift + E
4) 코드 축소 ctrl+shift+(-,+)
```

- pandas read.csv 인코딩
  - `data = pd.read_csv('0009.csv',encoding='CP949')`
- csv파일(df.to_csv)로 저장할 때 한글깨짐 현상 : encoding='utf8' -> encoding='utf-8-sig'
  - `pd.DataFrame(rows).to_csv("test2.csv", header = False, index = False, encoding='utf-8-sig')`
- csv 파일 저장 다른 코드 *

```
with open("test.csv", "wt", encoding = 'utf8') as f: # wt: write mode
writer = csv.writer(f)
writer.writerows(rows)
```

- 엑셀(xlsx) 파일 저장 코드 *

```
# 엑셀로 저장하기
df.to_excel("test.xlsx")
# 엑셀파일 읽기
df = pd.read_excel("test.xlsx")
```

- 폴더 없으면 생성하기
  - ```
    if not os.path.exists(model_object_file_path):
    os.makedirs(model_object_file_path)
    ```
- jupyter notebook에서 여러 output을 한 번에 출력하기
    ```

```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

- jupyter notebook에서는 한 cell에서 output을 마지막 하나만 출력할 수 있어 불편했다.
- 아래 코드로 패키지를 불러와 설정을 해주면 한 cell 내에서 여러 개의 결과값을 모두 출력할 수 있게 된다.
```

- 가상환경 관련 코드
  - 가상환경 생성
    - `conda create -n my_python_env`
  - 가상환경 생성 응용
    - `conda create --name YOUR_ENV_NAME python=3.6.5 tensorflow keras`
  - 버전 확인
    - `conda --version`
  - 가상환경 업데이트
    - `conda update`
  - 가상환경 활성화
    - `conda activate YOUR_ENV_NAME`
  - 가상환경 비활성화
    - `conda deactivate`
  - 가상환경 목록 확인
    - `conda env list`
  - 가상환경 삭제
    - `conda remove --name YOUR_ENV_NAME --all`
  - 가상환경 추출
    - `conda env export --name YOUR_ENV_NAME > environment.yml`
  - 추출한 가상환경으로 새로운 가상환경 생성
    - `conda env create -f ./environment.yml`

- 파이썬에서 GPU 정보와 사용 유무 확인하기

```
import torch
print(torch.cuda.is_available())
print(torch.cuda.device_count())
print(torch.cuda.get_device_name(torch.cuda.current_device()))
```

```
# confirm TensorFlow sees the GPU
from tensorflow.python.client import device_lib
assert 'GPU' in str(device_lib.list_local_devices())

# confirm Keras sees the GPU
from keras import backend
assert len(backend.tensorflow_backend._get_available_gpus()) > 0
```

```
# confirm PyTorch sees the GPU
from torch import cuda
assert cuda.is_available()
assert cuda.device_count() > 0
print(cuda.get_device_name(cuda.current_device()))
```

Linux Command

- 사용 tool MobaXterm
  - `shell __file__` or `python "__file__"`
- `htop` : 서버 사용량 확인
- 커멘드로 파일 또는 폴더 지우기
  - `rm -r -f /home/desktop/folder/file`
- `cd` : 경로 이동
- sh 파일 실행
  - `./train.sh 20190622 20190702 12`
- 파일 & 폴더 복사
  - 폴더 전체를 복사 : foler_1을 foler_2로 복사
    - `cp -R /home/desktop/folder_1 /home/desktop/folder_2`
  - 폴더 하위 파일들을 복사 : foler_1의 하위 폴더들을 folder_2에 복사
    - `cp -R /home/desktop/folder/file/. /home/desktop/folder_2`
- Process 종료
  - kill -9 35221(해당 프로세스 name) or `ctrl + Shift + C`

리눅스 GIT Command

- git clone(복사) : cd로 원하는 폴더 이동 후 Command 입력
  - `git clone http://----.git(주소 복사) commit/` 주소의 git을 commit 폴더로 복사
- Commit
  - 1.global setting
    - ex) (유저네임 , email ,비밀번호) `git config --global user.email "koos.gu@--.com" git config --global user.name "koos.gu"`
  - 2.커밋 폴더로 경로 이동 `cd` 사용
    - 최상위로 이동 : `cd ..`
  - 3.log 확인
    - `git log` or `git log -1`
    - `git status`
  - 4.Add
    - `git add scripts/SFA_XGboost.R scripts/Xgboost.R`
    - 여러 파일 한번에 add(추가)하기
      - 1. `git add -i`
      - 2. `add unstracked(4번)` 진입
      - 3. 열개 파일 있으면 1,2,3,4,5,6,7,8,9,10 입력
      - 4. ctrl(컨트롤)+d or ctrl(컨트롤)+c 로 나가기

        - 정상적으로 커밋 후 푸쉬
    - 5.Commit
        - `git commit` : [detail NOTE] 입력 후 커밋 내용 작성
    - 6.Push
        - `git push origin master`
    - 7.branch
        - branch 가져올 때
            - `git clone -b stage.oper(branch이름) http://----.git 생성폴더이름(안넣어도됨)`
        - GIT에서 branch 이후 git clone
            - `git clone http://-----.git -b operation(branch이름) oper(생성폴더이름)/`
            - oper폴더로 이동 : `cd oper/`
            - 목록 확인 : `ls`
            - 폴더가 어떤 branch를 향하고 있는지 확인 : `git branch`
            - 전체 branch 확인 : `git branch -a`
            - add, rm, cp -R 등 작업 후 커밋, 푸쉬
                - `git add scripts/---.R scripts/--.R`
                - `git commit -a` (변경사항 전체 커밋), `git commit 변경파일이름1, 변경파일이름2,...` (부분 커밋)
                - `git push origin operation(브런치 이름)`
    - 8.ls
        - 목록보기
        - 숨김 파일 및 폴더 모두 확인 : `ls -ahl`
    - 9.파일 이름 및 위치 변경
        - `mv preprocess.R preprocess2.R/`
    - cd
        - 최상위 경로 이동 : `cd /`
        - 이전 경로 이동 : `cd -`
- 이 페이지 참고하면 좋음 : http://rogerdudler.github.io/git-guide/index.ko.html

파이썬 코드로 R 스크립트 실행하기

- code

```
import subprocess

user_home = os.environ['user_home']
script_path = os.path.join(user_home,'scripts','default_model.R')

os.system(script_path)
```

Deep learning 관련 함수

- label categorical 범주형 변환
  - `from keras.utils import np_utils`
  - `Y = np_utils.to_categorical(label, num_classes)`
- train test dataset split
  - `from sklearn.model_selection import train_test_split`
  - `X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.15, random_state=2)`

알아두면 좋은 상식

- conda와 pip의 차이점
  - `pip` 는 패키지 관리 도구이다. 모듈을 설치하거나 모듈간 디펜던시를 관리하거나 할 때 사용한다.
  - `conda` 라는 도구는 virtualenv 와 같이 가상 환경을 제공하는 도구이다.
    - 즉, conda를 사용해서 별도의 가상 환경을 만들어 격리(독립공간)시키고 그 격리된 공간에서 pip를 사용해서 패키지들을 설치한다. (물론 conda 도 anaconda.org에서 관리하는 패키지들을 설치할 수 있다.)
    - conda 같은 경우 virtualenv + pip 같은 느낌이지만 설치할 수 있는 패키지가 anaconda.org 에서 관리하는 패키지로 한정된다는 제한이 있다.
  - 참고 사이트 : https://hashcode.co.kr/questions/3873/conda%EC%99%80-pip%EC%9D%98-%EC%B0%A8%EC%9D%B4%EA%B0%80-%EB%AC%B4%EC%97%87%EC%9D%B8%EA%B0%80%EC%9A%94
- 쥬피터 노트북(Jupyter notebook) 테마 변경하기
  - 1.anaconda prompt : `pip install jupyterthemes` 입력
  - 2.테마 입력
  - 여러 테마 예제
    - `jt -t oceans16 -f hack -fs 12 -T -N`
    - `jt -t grade3 -f roboto -fs 12 altp -tfs 12 -nfs 12 -cellw 80% -T -N`
    - `jt -t onedork -f bitstream -T -N -fs 12 -nf opensans -nfs 10` *

```
- f bitstream : 코드 폰트를 bitstream으로 적용
- - fs 12 : 글자 크기 12
- -T -N : 툴바와 제목을 보임 // default는 안보이는 상태라서 코드를 더 많이 볼 수 있습니다.
- -nf : 주피터노트북 폰트를 opensans로 적용
- -nfs 10 : 노트북 글자 크기 10
```

    - `jt -t onedork -T -f oxygen`
      - 이 테마 적용 시 파이플롯 옵션 추가 `fig = plt.figure()` / `fig.patch.set_facecolor('xkcd:white')`
    - `jt -t onedork -fs 95 -tfs 11 -nfs 115 -cellw 70% -T`
      - 이 테마 적용 시 output 잘려보이는 부분 css옵션 추가
      - ~/.jupyter/custom/custom.css 에서

```
div.output_area {
display: -webkit-box;
padding-left: 20px;
}
```

    - `jt -t gruvboxd -fs 11 -tfs 12 -dfs 9 -nfs 11 -ofs 11 -f office`
    - `jt -t chesterish`
    - default로 되돌리기 : `jt -r`
    - 테마 리스트 확인 : `jt -l`
  - etc 기능

```
-h 명령어 documentation 출력
-l 적용할 수 있는 테마 이름출력
-t theme 해당 이름의 테마 적용
-f fontname 폰트의 이름 적용.  고정폭 글꼴만 가능하다보니 선택에 제한있음.
hack consola 등등
-fs 폰트사이즈
-T 툴바 보이기
-N 이름 & 로고 보이기
```

    - 내 현재 테마
      - `jt -t onedork -T -N -kl -f roboto -fs 11 -tfs 11 -nfs 13 -tfs 13 -ofs 10 -cellw 80% -lineh 170 -cursc r -cursw 6`
      - 참고 : https://chancoding.tistory.com/48

- 크롬 캡쳐 프로그램없이 전체 화면 캡쳐하는 방법
  - f12(개발자 도구) -> ctrl + shift + p -> capture full size screenshot 입력
- 우분투 방화벽 설정 : Ubuntu ufw setting
  - 설정 환경 : ubuntu 16.04
  - 1. 방화벽 활성화
    - sudo ufw enable # UFW 활성화
    - sudo ufw disable # UFW 비활성화
    - sudo ufw status verbose #UFW 상태 확인
    - sudo ufw rest # 방화벽 초기화
  - 2. 특정 IP만 허용
    - sudo ufw allow form xxx.xxx.xxx.xxx to any port 22 proto tcp # IP영역대 22번 포트 접근 가능

이것 사용함.

    - sudo ufw allow from xxx.xxx.xxx.xxx[IP]
  - 3. 방화벽 rule 삭제

        - sudo ufw delete 2 #[sudo ufw status 에서 확인 후 원하는 숫자(rule) 입력]
    - 4. Ubuntu 서버 부팅시 ufw 자동 활성화 설정
        - sudo update-rc.d -f ufw defaults
    - 5. [방화벽 설정 사항]
        - 방화벽 로그 기록 활성화
        - 재부팅 시 방화벽 자동 활성화
        - 특정 IP만 접속 가능하도록 설정
    - 6. [ALLOW IP]
        - xxx.xxx.xxx.xx1
        - xxx.xxx.xxx.xx2
        - xxx.xxx.xxx.xx3
    - 7. [필수]
        - 재부팅 할 때 방화벽 활성화 되어 있는지 확인할 것 명령어는 'sudo ufw status verbose'