**목차**

1. 개발표준

* 개발스택
* 개발소프트웨어(IDE 포함)
* 파이썬 및 라이브러리 버전

1. 작업공간 (디렉토리) 설정
2. 코딩표준

* PEP 8 에 의거한 OnAD 코딩표준

1. 데이터베이스

* DB 테이블 설정 및 컬럼 설정
* DB aws설정

1. 형상관리

* 깃헙 (GitHub)

1. 품질관리

* 소나큐브

1. 개발 표준

* **1-1) 개발소프트웨어:**

|  |  |
| --- | --- |
| 항목 | 소프트웨어 |
| 개발언어 | Python3 (3.6.6), ipython |
| DBMS | MariaDB |
| IDE | Vscode, jupyter-notebook |
| OS | Windows7, ubuntu 버전 |
| 형태소분석기 | khaii |
| 형상관리 | GitHub |
| 스케쥴관리 | Trello |
| 품질관리 | Sonarqube |

* **1-2) 개발스택**

|  |
| --- |
| flask(nginx) |
| Pandas, khaii, keras |
| Beautifulsoup, Chatty, youtube/twitch API |
| Python3.6 |
| OS (ubuntu) |
| docker |

* **1-3) 라이브러리 버전 정리 :**
* 아나콘다 5.3.1
* 개발표준/Requirement.txt 파일
* asn1crypto         0.24.0     
  backcall           0.1.0      
  bleach             3.0.2      
  certifi            2018.1.18  
  chardet            3.0.4      
  cryptography       2.1.4      
  decorator          4.3.0      
  defusedxml         0.5.0      
  entrypoints        0.2.3      
  idna               2.6        
  ipykernel          5.1.0      
  ipython            7.2.0      
  ipython-genutils   0.2.0      
  ipywidgets         7.4.2      
  jedi               0.13.1     
  Jinja2             2.10       
  jsonschema         2.6.0      
  jupyter            1.0.0      
  jupyter-client     5.2.4      
  jupyter-console    6.0.0      
  jupyter-core       4.4.0      
  keyring            10.6.0     
  keyrings.alt       3.0        
  khaiii             0.1        
  MarkupSafe         1.1.0      
  mistune            0.8.4      
  nbconvert          5.4.0      
  nbformat           4.4.0      
  notebook           5.7.2      
  numpy              1.15.4     
  pandocfilters      1.4.2      
  parso              0.3.1      
  pexpect            4.6.0      
  pickleshare        0.7.5      
  Pillow             5.3.0      
  pip                18.1       
  prometheus-client  0.5.0      
  prompt-toolkit     2.0.7      
  ptyprocess         0.6.0      
  pycrypto           2.6.1      
  Pygments           2.3.0      
  pygobject          3.26.1     
  python-dateutil    2.7.5      
  pyxdg              0.25       
  pyzmq              17.1.2     
  qtconsole          4.4.3      
  requests           2.18.4     
  SecretStorage      2.3.1      
  Send2Trash         1.5.0      
  setuptools         39.0.1     
  six                1.11.0     
  ssh-import-id      5.7        
  terminado          0.8.1      
  testpath           0.4.2      
  torch              0.4.1      
  torchvision        0.2.1      
  tornado            5.1.1      
  tqdm               4.28.1     
  traitlets          4.3.2      
  urllib3            1.22       
  virtualenv         15.1.0     
  wcwidth            0.1.7      
  webencodings       0.5.1      
  wheel              0.30.0     
  widgetsnbextension 3.4.2
* + 추가

Matplotlib 3.0.1

Pandas 0.23.4

Flask 1.0.2

1. 작업공간 설정

Root - Onad

* **Data**
* **# 온애드/data**
* **# 수집한 데이터들을 담아놓는 공간**
* - twitch\_live\_chat
* > 트위치 채팅
* - twitch\_live\_stream\_list
* > 트위치 라이브 스트리밍 중 데이터
* - youtube\_live\_chat
* > 유투브라이브 채팅
* - youtube\_reple
* > 유투브 댓글들
* - more..
* **Docs**
* 문서모음
* **Lib**
* **# 온애드/lib**
* **## library화 한 onad 기능들**
* - onad\_analysis : 분석관련 기능모음
* > 예를들어, 시각화를 시켜주는 함수
* - onad\_contact\_db : 데이터베이스와 접촉하는 기능모음
* > 예를들어, db에서 데이터를 가져옴
* > 하지만 지금은 db를 사용하지 않기 때문에 크게 신경을 쓰지 않아도 됨.
* - onad\_get\_data : 크롤러 및 api사용과 같이 데이터를 수집하는 기능 모음
* > 하위 디렉토리
* >- /twitch\_crawl
* >- /twitch\_api
* >- /youtube\_crawl
* >- /youtube\_api
* - onad\_model\_fitting : ML 관련된 기능 모음
* > 예로, 가공된 학습데이터를 불러들여 훈련시키는 함수, 예측하는 함수 ( 더 세세히 나누고 추가할 필요성이 보임)
* - onad\_set\_data : 데이터를 가공 및 전처리하는 기능 모음
* > 예를 들어 pandas.DataFrame 으로 반환하는 함수
* **Onad\_web**
* **# onad/onad\_web**
* 구동파일
* - wsgi.py
* 하위 디렉토리
* service/model : DB 커넥션 및 DB 설정 관련
* > **\_\_init\_\_**.py : sqlalchemy 설정
* service/controller : 데이터를 처리하거나, 디비와 접촉하는 등의 대부분의 일들이 존재
* > asdf.py : 어떠한 일
* service/templates : view폴더로, html파일들
* service/static : js, css, 설정파일, 업로드파일, 감성분석모델(?)-들어가게 된다면
* > /js - js파일
* > /css - css파일
* > /storage - 업로드파일
* > /ml\_model
* **Onad\_run:**
* # 하위 디렉토리
* - run\_get\_data : 데이터 가져와 DB에 저장하는 실행파일 폴더
* - onad\_store\_analized : 데이터 정제 이후 정제된 데이터(편집점데이터, 감성분석모델 등)를 저장하는 실행파일 폴더
* - 추가할 필요가 있으면 회의
* **Ml\_model:**
* 학습된 모델을 dump 하여 저장하는 폴더
* Model 파일 형식:
* Senti\_01.model
* senti\_02.model
* 이후 추가적인 내용은 PEP8참조

1. 코딩표준
2. 네이밍 룰

상수: ASDF\_GH

변수: asdf\_asdf

클래스 명: ClassName

함수 명: function\_name

* 반환값이 True/False 인 함수: is\_function\_name
* 크롤링 함수: crawl\_function\_name
* 시각화 함수: view\_function\_name
* api요청하여 데이터 가져오는 함수: get\_function\_name
* 데이터 정제하는 함수: filter\_function\_name
* 분석하는 함수: anal\_function\_name
* 이외 임의로 정하나, 맨 앞에 의미가 명확한 단어를 쓰도록 합시다.

길어도 좋으니 의미가 명확하도록 하는 것이 좋다고 생각합니다.

1. 파이썬 문법
2. 들여쓰기: 띄워쓰기4번 – vscode와 jupyter notebook 사용시 자동적으로 세팅됨
3. 구분자(,) 뒤에는 띄어쓰기 한번
4. 함수의 인자가 길어지는 경우 아래로 펼쳐 쓴다.

def long\_function\_name(variable1,

variable2, variable3, variable4,

variable5):

print(variable1)

1. If 문의 조건이 길어지는 경우 and 또는 or 를 기점으로 아래로 펼쳐 쓴다.

# 추가적인 들여쓰기를 사용하지 않음

if (this\_is\_one\_thing and

that\_is\_another\_thing):

do\_something()

1. 리스트등의 컨테이너 타입의 목록이 긴 경우 – 아래로 펼쳐 쓴다.

my\_list = [

1, 2, 3,

4, 5, 6,

]

result = some\_function\_that\_takes\_arguments(

'a', 'b', 'c',

'd', 'e',

1. 클래스 안의 메소드는 한 줄로 구분한다. 일반 함수의 경우 두 줄로 구분한다.

# 메소드끼리는 1줄, 함수끼리는 두줄

# 메소드의 경우

class ClassName():

def \_\_init\_\_(self):

return None

def func1(self):

return None

# 일반함수의 경우

def asdf():

return None

def func2():

return None

1. 인덱싱과 슬라이싱의 경우 불필요한 띄우기 사용하지 않는다.

# 인덱싱과 슬라이싱의 경우

yes:

a[:5]

a[1:5]

a[5][4][key]

no :

a[ :5 ]

a[1: 5]

a[ 5 ] [ 4 ] [ key ]

1. 딕셔너리자료형의 경우 ‘:’의 뒤에는 한칸을 띄운다

# 딕셔너리의 경우 : 이후에 한번의 띄어쓰기

yes\_dic = {

key: value,

key1: value1

}

no\_dic = {

key:value,

key1:value1

}

1. 함수의 인자를 넣을 땐 “=” 의 양옆에 띄우지 않는다.

# 함수의 인자는 띄우지 않는다.

yes:

create\_new\_dict(key=key1, value=value1)

no:

create\_new\_dict(key = key2, value = value2)

1. 주석처리의 경우

# 주석처리

asdf(asdfasdf) # 짧은줄에 짧은 주석은 그 줄의 끝에서 두번띄우고 쓴다

# 긴 주석의 경우 그 줄의 위에 쓴다. 마지막에 자신의 날짜와 이름을 적으십시오.

# 두줄 이상의 주석이라도 설명하고자 하는 줄 위에 쓴다. - 1217,강화수

var1 = 12345

# 함수또는 메소드의 설명은

"""

3따옴표 를 활용하여 설명한다. 이 함수는 뭐시기뭐시기 함수입니다.

input: 들어오는 인자 목록과 그 설명

- 들어가는 인자1: 이인자는 뭐시기입니다, 뭐뭐 하는거니 무슨 값을 넣으세요

- 들어가는 인자2: 설명

output: 나오는 출력의 목록 리턴값

"""

1. Import 방식

# 무조건 lib 아래에서 시작한다

from lib.get\_data import function\_name, ClassName

1. 데이터 베이스
2. DB 설정 및 테이블 설정

* DB 설정

|  |  |
| --- | --- |
| 호스트 | pythondb.cfvnihegi7no.ap-northeast-2.rds.amazonaws.com |
| 포트 | 3306 |
| 데이터베이스 명 | onad |
| DB user | root |
| DB password | \*\*\*\*\*\*\*\* |
| Charset | utf8mb4\_general\_ci |
| Log flag | False |

* 테이블 설정

|  |  |
| --- | --- |
| 개체 명 | 속성 |
| 회원(광고주)  정보 | 회원 번호(PRIMARY), 아이디, 비밀번호, 담당자 명, 담당자전화번호, 담당자이메일, 소속 기업 명, 소속 기업 코드 |
| 기업  정보 | 기업 코드(PRIMARY), 기업체 명, 국가 명, 사업자등록번호, 대표자 명, 업종, 주요 상품 군, 회사홈페이지주소 |
| 회원(크리에이터)  정보 | 회원 번호(PRIMARY), 방송 플랫폼, 크리에이터(채널)아이디, 아이디, 비밀번호, 이름, 닉네임, 전화번호, 이메일, 방송국홈페이지/feed, 권한요청여부, 방송 송출 프로그램 |
| 라이브  채팅 | 채팅 번호(PRIMARY), 날짜 및 시간, 시청자 아이디, 채팅 내용, 방송id, 크리에이터(채널)아이디, 크리에이터 이름, 방송 제목, 시청자 수, 방송카테고리 |
| 하이라이트 포인트(편집점) | 방송 플랫폼, 크리에이터(채널)아이디, 크리에이터 닉네임, 방송 날짜, 최다채팅구간 |
| 유튜버 정보 | 채널 아이디(PRIMARY), 채널 명, 구독자 수, 채널 조회수, 채널 이미지URL, 채널내 영상 수 |
| 유튜브  영상 | 영상 번호(PRIMARY), 영상ID, 영상 게시자, 영상 제목, 영상 조회수, 영상 좋아요, 영상 싫어요, 댓글 개수, 영상 썸네일url, 영상카테고리 |
| 댓글 정보 | 댓글 번호(PRIMARY), 작성자, 댓글 게시 날짜, 댓글 내용,  영상 번호, |
| 모델 버전 관리 | 버전 정보(primary), 날짜, 모델 파일 명, f-1 score, 정확도, 코멘트 |

1. 형상관리

* 형상관리는 깃허브 사용
* https://github.com/hwasurr/OnAD\_project
* branch 룰
  + master : 최종본
  + develop : 작업본
  + **대부분의 경우 develop브랜치에 commit 하기 바람**
  + master branch에 커밋하는 경우 :  
    해당기능을 최종 완성시 또는에만(토의 이후)  
    또는, document 수정
* 기본 용어
  + **branch** : master의 가지 (다른 작업을 한 이후 master에 합치는 식)
  + **commit** : 지금 상태를 스냅샷 찍어 기록을 남긴다고 생각하면됨  
    다 남기 때문에 예전으로 돌아갈 수 있음  
    하지만 이는 로컬pc에서만 저장됨 github에 올리기 위해서는 push
  + **push** : 커밋한 상태인 로컬pc의 상태로 github를 최신화시키는 것
  + **pull** : 가장 최신의 상태를 로컬pc로 가져오는 것
* vscode 에서 git사용:
  + git이 없다면 설치
  + 이후 cmd 창에서

git config --global user.name "이름" git config --global user.email "이메일"

* + 이후 현재 레파지토리의 clone or download 클릭
  + url 복사
  + vscode 열고, ctrl + shift + p
  + 명령창에 git:clone 입력 또는 클릭
  + 폴더 선택창이 뜰텐데, 빈 폴더 하나 생성하여 확인클릭
  + 작업한 이후 vscode의 왼쪽 탭 3번째 가지모양 클릭
  + 체크모양의 Commit 단추 누르면 commit이 됨 (**코멘트입력** : 고친내용 or 추가한내용\_181218\_강화수)
* commit 룰
  + commit은 뭔가가 바뀌면 바로바로 하는게 좋을 것 같아보임.(바뀜의 단위는 자신의 기준)
  + **하교 전 저녁에는 push, 출첵 후 아침에는 pull 하는게 바람직할 듯**
  + push, pull 모두 ''' 단추를 누르면 할 수 있음. **(push 는 commit 이후!!)**
  + develop 브랜치로 변경하려면 vscode 에서 왼쪽아래 줄기표시 누르면 바꿀 수 있음.
  + **develop 브랜치로 변경 필수!!** 좌측아래 줄기표시 develop로 변경한 이후 commit
  + develop에 commit할 경우 push to -> origin -> develop 선택 (develop가 안뜬다면 화수에게 말하세요)
  + 문서를 교체한 후 commit 하기 위해서는 문서내용 바꾼 이후 저장한 이후 vscode onad폴더 들어오면  
    commit 할 수 있음

1. 품질관리

* 소나큐브(sonarqube)