2019년 7월 1일 월요일
1과목 파이썬 개념

*문자형 3개
숫자형 4개 (int, float, bool, complex)

* list : 시퀀스타입, mutable
tuple : 시퀀스타입, immutable (불변)
set : no 시퀀스타입, mutable
▷ immutable (불변) : 고정된 값을 갖는 객체

* C++ : 다중상속 O vs JAVA : 다중상속X

* 파이썬 단점: 속도가 느리다
모바일 지원이 약하다.
장점: 우아함, 동적 타이핑

* R과 파이썬은 할 수 있는 범위가 다르다.
파이썬은 R에 비해 자동화시키기 쉬움

R: domain -> 소스코드가 짧아서 논문에 포함됨.
파이썬: general -> 논문구현 스터디 등 소스코드가 공개되어있지 않음(너무 길어서)

* 파이썬은
Dynamic language
Interpreted Language(컴파일x, 결과가 바로 나옴)

REPL : 실행하면 바로 나옴 (=Playground)
IDE
Text Editor

* 파이썬 키워드 갯수 : 35개

2과목 파이썬 실습

```
#!/usr/bin/env python
# coding: utf-8

# In[1]: import this
# 파이썬은 객체지향 패러다임이다

# In[3]: import keyword
# In[4]: dir(keyword)
# In[5]: keyword.kwlist
# 컨테이너의 개수는 len으로 알 수 있고 파이썬은 내가 사용할 수 있는 키워드 숫자가 적음으로 배우기 쉬움

# In[7]: a = 1
# 할당문
# 변수보다는 식별자 또는 이름이라고 한다.
# 왼쪽에는 식별자, 이름 오른쪽에는 표현식이 나온다.
# 식은 하나의 결과 값으로 축약할 수 있는 애를 식이라고 한다.
    ex. a = 1 + 2  (=3)
# 이름을 정하는 규칙 : coding style guide
[ 카넬 방식: 두 번째 단어를 이을 때 두번째 글자를 대문자로 클래스
                이름 만들 때 ex. moonBeauty
 파스칼 방식: 첫 번째 단어를 대문자로 ex. MoonBeauty
 underba: 식별자 만들 때 주로 사용 ex. moon_beauty]

# 파이썬은 실수가 없음 어떤 값을 바꾸게 해주는것이 없음.
# 식별자는 명명규칙과 pep8을 따라야한다.
  ▷ PEP8 : 파이썬 개선 제안서, 파이썬 코드를 어떻게 구상할지
           알려주는 스타일 가이드

# In[8]: 문근영 = 1
# 파이썬 3부터 유니코드 사용 가능 그렇다면 식별자 이름을 한글로 써도 될까? -> YES (언어상 문제는 없음, 그러나 명명규칙 오류)
# 식별자는 a~z까지 사용하라고 되어있음, 숫자x, 특수문자 _만 가능
#정리 식별자 규칙
1. 에러나는 것과(시작 특수문자, 숫자 등)
2. 에러는 안 나지만 따라야하는 가이드라인이 있음(한글)

# In[9]: b = 3 if a > 0 else 5
# 오른쪽도 결국 식이기 때문에 if가 있어도 할당 된다.

# In[10]: get_ipython().run_line_magic('whos', '')
# nameSpace보는 방법 %whos 여기에 없으면 error 난다.

# In[11]: id(a)
# 정리 NameError는 NameSpace에 없는 것을 호출해 내는 것.
# 그렇다면 메모리 공간에 어떻게 할당되는 것인가? : id(a)

# In[12]: hex(id(a))
# 메모리는 보통 16진수인데 위는 10진수로 나옴 그래서 : hex(id(a))
```

```
# In[13]: from sys import intern
          a = 150
          b = 150
# a=1000 , b=1000일 때 메모리 공간은 같을까?
  -> 다르다. 새로 할당되기 때문에
# 단, 정수 범위 일 때는 똑같은 메모리 범위를 가르친다. -5 ~ 256
  (* interning이라고 한다.)

# In[14]: hex(id(a))
          hex(id(b))
# In[15]: a == b
# == 값이 같은지 아닌지

# In[16]: a is b
# is 라는 연산자는 메모리 값이 같은지까지 확인한다.
# 즉 interning기법이 적용된 -5 ~ 256 범위는 is일 때 True!

# In[17]: type(a)
# 동적타입 (Dynamic type) 그러나 내부적으로 알아서 정해준다.
  명령어 : type()
# 기본적인 타입 13가지

# 정리 a = 1
1이 1) 메모리공간에 할당되고 2) 타입이 정해진다. 정해진 타입에 지어진 3) 이름(식별자)에 따라 프린팅 된다.
# 파이썬이 c로 만들어져서 포인터로 이루어져있기 때문에.

# In[19]: import sys
# 리터럴(Literal)방식: 객체지향방식, 리터럴 방식으로 나누어져있다.
# 리터럴 : 정수는 리터럴이 아무것도 붙지않을 때 리터럴이 정수이다.
  ex 1. -> float (타입에 대한 힌트를 줌 float 리터럴)
# 정수는 리터럴이 없고 float은 2. 과 it, infinity 등이 있다.
# 파이썬에는 int형이 하나 밖에 없음, 크기에 상관없다.
# ex. long, longlong을 썼을 때 해당 범위를 넘으면 오버플로우 생김.
  파이썬은 이런 선언이 없음으로 오버플로우가 발생하지 않음
# 파이썬에서는 이러한 방식 말고 다른 방식으로 오버플로우가 발생함
-> because. 자동으로 메모리크기를 확대시켜줌

# In[20]: sys.maxsize
#  maxsize보다 크면 오버플로우가 발생함.
# 여기에서는  sys.maxsize = 9223372036854775807

# In[21]: 9223372036854775807 + 1
# 그러나 파이썬에는 발생하지 않음
# 큰 수를 다룰 때 파이썬에서 ,를 찍을 수 있을까? -> NO
# 파이썬 3.6부터는 , 대신에 _를 씀으로써 큰 숫자를 쉽게 볼 수 있게 한다.
# 언더바(_ 구분자)의 위치는 아무대나 상관없음
  (한국, 미국 등에서 ,의 위치가 다 다르기 때문이다.)
```

```
# In[31]: a = -1
          a.__abs__()
#식: 하나의 값으로 축약할 수 있는가 100_000_000도 결국 하나의 값
임으로 식이기에 오른쪽에 위치할 수 있다.
#abs: 절대 값  함수

# In[32]: (-1).__abs__()
# -1.__abs__() 와 -1 .__abs__()의 차이 전자는 오류가나고 후자는
오류가 안남.
왜냐하면 -1. 으로 float으로 인식했기 때문에 그래서 다음과 같은 경우
에서는 오류가 나지 않기 위해서 (-1).__abs__()와 같이 사용한다.

# In[33]: dir(a)
# 디렉토리 명령어

# In[34]: sys.float_info
# 부동소수점에 대한 정보를 알 수 있다. sys.float_info
# 파이썬에는 무한대가 있다.  -> 부동소수로 표현한다.

# In[36]: x = float('infinity')
          x
# 무한대
# x의 결과 값 infi

# In[37]: t = float('nan')
          t
# 부동소수의 max 값인 1.7976931348623157e+308 에
   1.7976931348623157e+309을 더해보자 -> 결과: inf
# 분석할 때 무한대가 나오는 경우가 많음 : 파이썬 바인딩 사용 이유
# non은 float으로 지정해준다.

# In[42]: 1 == 1+1 #false
# In[49]: 1 == a+1 #false
# In[52]: a = 1.7976931348623157e+308
          a == a+1   #true ..왜지?

# In[54]: 0.1 + 0.1 + 0.1
# 결과가 0.3으로 나오지 않는다. 0.3000000000000004 이를 해결할
수 있는 방법은 분수와 소수점을 계산하는 패키지를 이용한다.
# decimal , fractions
# 컴퓨터는 유리수를 잘 다루지 못한다.

# In[55]: a = 3 + 2j
# 정리 Inf 개념과 non 개념이 float으로 들어간다.
# 파이썬에서 복소수(파이)를 j로 표현한다.

# In[57]: type(a)
# complex 타입 : 복소수타입

# In[58]: dir(a)
# In[62]: a.conjugate()
# conjugate 결과 (3-2j)
```

In[67]: type(True)
true = 1 false =0 -> bool타입

In[68]: issubclass(bool,int)
bool타입은 int에 상속되었는가 -> True
즉, bool은 100% 숫자처럼 행동한다.

In[69]: True + True
bool타입을 100% 숫자처럼 사용함으로 다음의 결과는 2로 나온다.

In[70]: True is 1 #False
왜 -5~ 256인데 interning이 되지 않는가?
true false는 메모리에 따로 할당되어있다.

# In[71]: a = 1000
b = 1000
# In[72]: a is b
# false interning내의 값이 아님으로.

In[73]: id (a)
메모리 위치 값
In[74]: a = 1000
In[75]: id(a)
메모리 위치 값 바뀜 -> 재할당하면 기본적으로 메모리 값이 바뀐다. 같은 값을 할당해도 메모리 값은 바뀐다.
안바뀌는 경우도 있을까? interning 기법에 해당하는 값은 재할당해도 메모리 값이 바뀌지 않는다.
그러나 파이썬 3.7에서 바뀜 . interning범위를 넘어도 interning이 됨 -> 이건 고급과정 ^_^

In[79]: a = [2]
In[80]: id (a)
In[81]: a.append(3)
In[82]: a
In[83]: id(a)
배열에서 appned를 해도 메모리 값은 바뀌지 않는다. a.append(3) [2,3]이 되어도 메모리 값은 같음
#실제 첫 번째 취소 값 2 만 메모리 값으로 반영한다.
정리 파이썬 숫자형 4개, bool타입은 int상속 결국 숫자와 똑같다.

# In[86]: a = u'문근영'
type(a)
# type(a) -> str
# 파이썬에서는 바이트를 숫자형으로 쓰지만 문자형으로 많이쓴다.

In[87]: a = b'abde'
In[88]: b = bytes(67)
In[93]: b
bytes는 immutable이다.

```
# In[94]: a ='''
          문근영
          '''
# 문자열은 입력마크를 하나, 두개, 세 개 쓸 때 가 있음
# 세 개 쓸 때 여러 줄 문자일 때 사용한다.
  결과 : \n\n문근영\n\n
# 만약에 \n가 거슬린다하면 print를 사용한다.
# ''' ''' 세개는 Docstring에서 자주 사용한다.

# In[95]: print(a)

# In[96]: a = '문근영'
          b = '문근영'
# In[97]: a is b # False
# In[98]: a = 'moon'
          b = 'moon'
# In[99]: a is b # True
# 왜 true와 false로 나뉠까: 문장도 interning이 있기 때문이다.
  공백이면 x, a~z일때
# == equlitity , is identification


# In[132]: a = 'abc'
# 문자 관점에서 보면 3개임,
  문자열은 데이터를 한꺼번에 담는애임 = container
# 문자열 중 순서가 중요한것 : Homogeneous한 sequence type
# sequence type은 인덱싱과 슬라이싱을 지원한다.
# 인덱싱 초과시 인덱스error 발생
▷호모지니어스는 '동질의' 이라는 의미이며
  헤테로지니어스는 '서로 다른 종류들로 이루어진'


# In[136]: a[1:5]
# 인덱스 슬라이싱 a[1:3]
# 인덱스 슬라이싱은 범위가 벗어나도 에러가 나지 않는다.

# In[138]: a = '문근영 예뻐요'
# In[139]: a[::-1]
# 역순 출력

# In[140]: a[::2]
# a[::2] = 두글자 마다

# In[141]: a[slice(1,3)]
#이러한 기법을 이용해 목차를 뽑아낼 수 있다.

# In[142]: a = [1,'a']
# 순서가 중요하지 않지 않고 헤테로지니어스: list

# In[143]: range(100)[50]
# 호모지니어스 : str, bytes, bytesarr
# range() 숫자만 나옴, 호모지니어스 시퀀스 중요 , 숫자형 중 INT형
```

각각의 원소를 구할 때 len 사용, len은 항상 true 다 ? -> NO

#append, clear 와 같은게 있으면 mutable하다.
#list와 짝 인에 tuple , tuple은 immutable
순서가 없는데 헤테로지니어스한 것 : set (집합)

#In[148]: a = {3,1,2}
adir(a)
# 집합은 순서가 없음 결과: {1,2,3}
# 순서가 없음으로 index, slicing 모두 불가능하다.
#mutable

In[149]: b = frozenset({1,2,3})
list(mutable) vs tuple(immutable)
set(mutable) vs forzenset (immutable)

#In[150]: b
파이썬에서 상수는 없음 그러나 immutable인 수는 있다.
파이썬에서는 모든 것을 재 할당 할 수 있다. 이러한 변경이슈를 막기 위해서 immutable이 중요한 역할을 한다.

3과목 특강
각종 오픈소스의 기술내용이 아닌 개발 매커니즘 (오픈소스 개발방식)

* git은 history 관리
* Hello world 찍을 때, Git이 Needs인 세상 == 오픈소스 프로젝트

* Openhub : openSource 관리 사이트

gcc test.c
gcc -pg test.c (각각의 함수에 따라 print 됨)

uftrace a.out (uftrace가 fork exe을 통해서 실행된다. 시간 출력, 트레이싱 방법)
uftrace &fibonacci fibonacci 5 (트레이싱 방법)

* uftrace : pull request(commit 제출, 다른 사람에게 제출할 때) / Review
contribute nb(?) : Coding style
commit으로 코드를 관리하면 line by line으로 관리함

* 좋은 표현 방식 before / after
* 크니까 잘게 나눠라 ? -> 몇 개로 나눠야하는가, 기준은? -> 정답이 없음, review와 discussion이 가능한 단위

* 마크다운 문법 : 들여쓰기 등 안 깨지게 가지고 오는 방법

*오픈소스 프로젝트 uftrace : issue/Disscusion 과정 예시
Label에 크리티컬(빨간색)을 달 수 있음, 버그(노란색) 등
버전에 대해 명시할 필요성과 현상에 대해 이야기해줄 수 있음

commit id, message를 통해서 역을 추적 할 수도 오류를 수정했는지 알 수 있다.