Python 설치 및 기초 강의

2022. 10. 22

권현섭 연구원(kert06@postech.ac.kr)

Contents

- 교육 환경 Setting(1H)

  - anconda3

  - jupyter notebook

- Python 프로그래밍 기초(3H)

  - 자료형(숫자, 문자열, 리스트, 튜플, 딕셔너리, 집합)

  - 제어문(if 문, while 문, for 문)

교육환경 Setting

- **환경 update**
  - command 창을 열고 다음과 같은 명령어 입력
    → **sudo apt-get upgrade**
    → **sudo apt-get update**

- **Anaconda 다운로드**
  → **wget https://repo.anaconda.com/archive/Anaconda3-2022.10-Linux-x86_64.sh**

교육환경 Setting

- **Anaconda 설치**
  - 해당 폴더로 진입하여 다음과 같은 명령어 실행

→ bash Anaconda3-2022.10-Linux-x86_64.sh

→ **설치시, 아래 화면 나오면 yes 입력**

```
Do you accept the license terms? [yes|no]
```

→ **아래의 화면에서는 엔터 입력 후 진행**

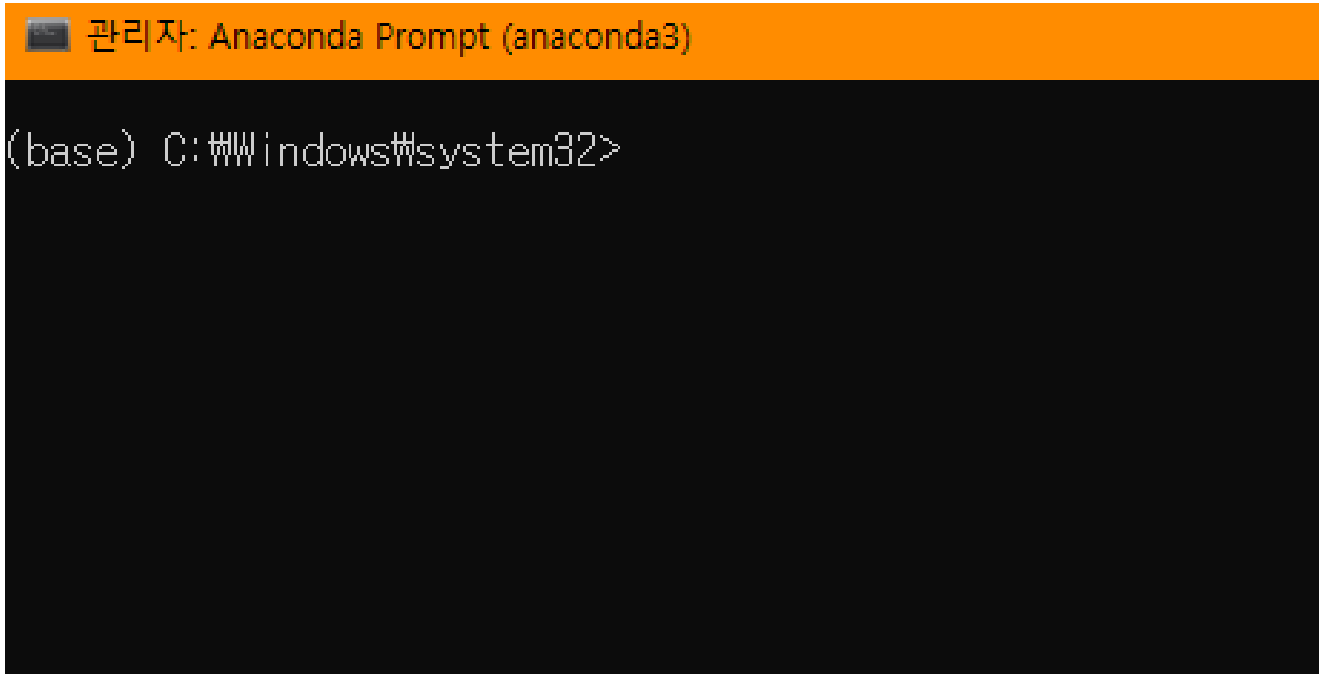
```
Anaconda3 will now be installed into this location:
/home/piai/anaconda3

  - Press ENTER to confirm the location
  - Press CTRL-C to abort the installation
  - Or specify a different location below

[/home/piai/anaconda3] >>>
```

→ **아래의 화면에서는 yes입력**

```
installation finished.
Do you wish the installer to initialize Anaconda3
by running conda init? [yes|no]
[no] >>>
```

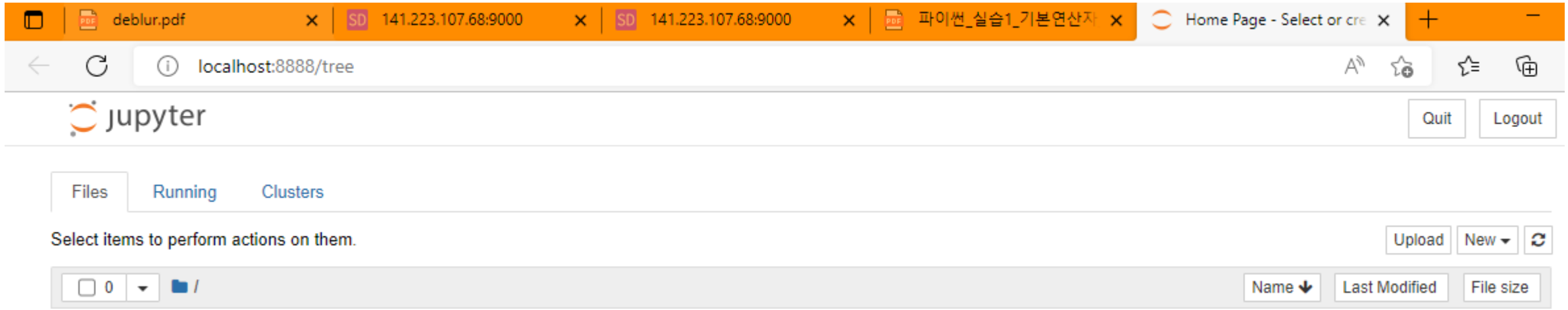
교육환경 Setting

- **Anaconda 설치**
  - → source ~/anaconda3/bin/activate
  - → conda init
  - → source ~/.bashrc
  - → conda config --set auto_activate_base True
  - → 터미널 재시작 후 아래와 같이 command 앞에 base가 있으면 성공



교육환경 Setting

- **Jupyter notebook 설치**
  - command 창에 다음과 같이 입력
  → conda install jupyter notebook
  → jupyter notebook



Python 프로그래밍 기초 - 자료형

- 변수란(variable)?
  1. 값을 저장하기 위한 메모리 공간
  2. 변수 이름 > 알파벳, 숫자, _ 로 구성 / 대소문자 구분

- 숫자(number)
  1. 정수형과 실수형

```
>>> a = 123
>>> a = -178
>>> a = 0
```

```
>>> a = 1.2
>>> a = -3.45
```

  2. 8진수와 16진수

```
>>> a = 0o177
```

```
>>> a = 0x8ff
>>> b = 0xABC
```

- 숫자(number)

3. 사칙연산

연산자	설명	예시
+	'더하기'를 한 값	2 + 1 = 3
-	'빼기'를 한 값	3 - 1 = 2
*	'곱하기'를 한 값	3 * 2 = 6
**	'거듭 제곱'과 동일	3 ** 3 = 27
/	'나누기'를 한 값	3 / 2 = 1.5
//	나누기를 한 뒤에, 소수점을 버린 값	3 // 2 = 1
%	나눈 뒤 '나머지' 값	5 % 3 = 2

- **숫자(number)**
  연습 문제 1: 홍길동 씨의 과목별 점수는 다음과 같다. 평균 점수를 구해보자.

과목	국어	수학	영어	과학	사회
점수	87	90	66	72	76

- 문자열 (string)

1. 문자는 작은따옴표(‘) 또는 큰따옴표로(“”)로 묶인 변수

```
"Hello World"
```

```
'Python is fun'
```

2. 문자열 연산

```
>>> head = "Python"
>>> tail = " is fun!"
>>> head + tail
'Python is fun!'
```

```
>>> a = "python"
>>> a * 2
'pythonpython'
```

3. 문자열 ‘in’ 키워드

- 자료에 특정 값이 있는 지 확인 / True or False

ex) a=‘Life’

‘i’ in word => True

‘n’ in word => False

- 문자열 (string)

4. 문자열 인덱싱(indexing)

```
>>> a = "Life is too short, You need Python"
```

```
Life is too short, You need Python
0         1         2         3
01234567890123456789012345678901234
```

```
>>> a = "Life is too short, You need Python"
>>> a[3]
'e'
```

5. 문자열 슬라이싱

```
>>> a = "Life is too short, You need Python"
>>> a[0:4]
'Life'
```

```
>>> a = "20010331Rainy"
>>> year = a[:4]
>>> day = a[4:8]
>>> weather = a[8:]
>>> year
'2001'
>>> day
'0331'
>>> weather
'Rainy'
```

문자열을 나누는 방법

- 문자열(string)

6. 문자열 포매팅(format)

```
>>> "I eat %d apples." % 3
'I eat 3 apples.'
```

```
>>> "I eat %s apples." % "five"
'I eat five apples.'
```

```
>>> number = 3
>>> "I eat %d apples." % number
'I eat 3 apples.'
```

```
>>> number = 10
>>> day = "three"
>>> "I ate %d apples. so I was sick for %s days." % (number, day)
'I ate 10 apples. so I was sick for three days.'
```

- 문자열(string)

6. 문자열 포매팅(format)

- 자료형에 따라 다른 문자열 코드를 확인

코드	설명
%s	문자열(String)
%c	문자 1개(character)
%d	정수(Integer)
%f	부동소수(floating-point)
%o	8진수
%x	16진수
%%	Literal % (문자 `%` 자체)

- format 함수 사용

```
>>> number = 10
>>> day = "three"
>>> "I ate {0} apples. so I was sick for {1} days.".format(number, day)
'I ate 10 apples. so I was sick for three days.'
```

- 문자열(string)

7. 문자열 관련함수

- 문자 개수 세기(count)

```
>>> a = "hobby"
>>> a.count('b')
2
```

- 위치 알려주기(index)

```
>>> a = "Life is too short"
>>> a.index('t')
8
```

- 문자열 삽입(join)

```
>>> ",".join('abcd')
'a,b,c,d'
```

- 소→대문자(upper)

```
>>> a = "hi"
>>> a.upper()
'HI'
```

- 대→소문자(lower)

```
>>> a = "HI"
>>> a.lower()
'hi'
```

- 문자열 길이(len)

```
>>> a = "Life is too short"
>>> len(a)
17
```

- 문자열 나누기(split)

```
>>> a = "Life is too short"
>>> a.split()
['Life', 'is', 'too', 'short']
>>> b = "a:b:c:d"
>>> b.split(':')
['a', 'b', 'c', 'd']
```

- **문자열(string)**

연습문제 1 : 주민등록번호 881120-1068234를 연월일(YYMMDD) 부분과 뒷자리번호를 나누어 출력해 보자.(슬라이싱)

연습문제 2 : 주민등록번호 뒷자리의 맨 첫 번째 숫자는 성별을 나타낸다. 주민등록 번호에서 성별을 나타내는 숫자를 출력해보자.(인덱싱)

- **리스트 (list)**

1. 리스트(list)는 여러 변수를 하나로 묶어서 관리할 수 있는 자료형

```
리스트명 = [요소1, 요소2, 요소3, ...]
```

```
>>> a = []
>>> b = [1, 2, 3]
>>> c = ['Life', 'is', 'too', 'short']
>>> d = [1, 2, 'Life', 'is']
>>> e = [1, 2, ['Life', 'is']]
```

2. 리스트 연산하기

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> a + b
[1, 2, 3, 4, 5, 6]
```

```
>>> a = [1, 2, 3]
>>> a * 3
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

```
>>> a = [1, 2, 3]
>>> len(a)
3
```

- 리스트 (list)

3. 리스트의 인텍싱(

```
>>> a = [1, 2, 3]
>>> a
[1, 2, 3]
```

```
>>> a[0]
1
```

```
>>> a[0] + a[2]
4
```

*** 이중 리스트**

```
>>> a = [1, 2, 3, ['a', 'b', 'c']]
```

```
>>> a[-1][1]
'b'
>>> a[-1][2]
'c'
```

4. 리스트의 슬라이싱

```
>>> a = [1, 2, 3, 4, 5]
>>> a[0:2]
[1, 2]
```

- 리스트 (list)

5. 리스트의 수정과 삭제

```
>>> a = [1, 2, 3]
>>> a[2] = 4
>>> a
[1, 2, 4]
```

```
>>> a = [1, 2, 3]
>>> del a[1]
>>> a
[1, 3]
```

```
>>> a = [1, 2, 3, 4, 5]
>>> del a[2:]
>>> a
[1, 2]
```

6. 리스트 관련 함수들

- 요소 추가하기(append)

```
>>> a = [1, 2, 3]
>>> a.append(4)
>>> a
[1, 2, 3, 4]
```

- 정렬하기(sort)

```
>>> a = [1, 4, 3, 2]
>>> a.sort()
>>> a
[1, 2, 3, 4]
```

- 리스트(list)

6. 리스트 관련 함수들

- 리스트 뒤집기(reverse)

```
>>> a = ['a', 'c', 'b']
>>> a.reverse()
>>> a
['b', 'c', 'a']
```

- x의 개수 세기 (count)

```
>>> a = [1,2,3,1]
>>> a.count(1)
2
```

- 위치 반환(index)

```
>>> a = [1,2,3]
>>> a.index(3)
2
>>> a.index(1)
0
```

- 요소 삽입(insert)

```
>>> a = [1, 2, 3]
>>> a.insert(0, 4)
>>> a
[4, 1, 2, 3]
```

- 요소제거(remove)

```
>>> a = [1, 2, 3, 1, 2, 3]
>>> a.remove(3)
>>> a
[1, 2, 1, 2, 3]
```

- 리스트(list)
  연습 문제 1 : [1,3,5,4,2] 리스트를 [6,5,4,3,2,1]로 만드시오.

- **튜플(tuple)**

  1. 튜플은 리스트와 거의 비슷하며, 다른 점은 다음과 같다.
     - 리스트는 []로 둘러싸지만 튜플은 ()로 둘러싼다.
     - 프로그램이 실행 되는 동안리스트는 값의 생성, 수정, 삭제가 가능하지만 튜플은 그 값을 바꿀 수 없다.

```
>>> t1 = ()
>>> t2 = (1,)
>>> t3 = (1, 2, 3)
>>> t4 = 1, 2, 3
>>> t5 = ('a', 'b', ('ab', 'cd'))
```

- 딕셔너리(dictionary)

1. 딕셔너리는 key와Value로 대응 관계로 구성되어있고, 순서가 없는 자료형

```
{Key1:Value1, Key2:Value2, Key3:Value3, ...}
```

```
>>> dic = {'name':'pey', 'phone':'0119993323', 'birth': '1118'}
```

key	value
name	pey
phone	01199993323
birth	1118

2. 딕셔너리 추가

```
>>> a = {1: 'a'}
>>> a[2] = 'b'
>>> a
{1: 'a', 2: 'b'}
```

```
>>> a['name'] = 'pey'
>>> a
{1: 'a', 2: 'b', 'name': 'pey'}
```

- 딕셔너리(dictionary)

3. 딕셔너리 삭제

```
>>> del a[1]
>>> a
{2: 'b', 'name': 'pey', 3: [1, 2, 3]}
```

4. 딕셔너리 활용

```
>>> dic = {'name':'pey', 'phone':'0119993323', 'birth': '1118'}
>>> dic['name']
'pey'
>>> dic['phone']
'0119993323'
>>> dic['birth']
'1118'
```

- 딕셔너리(dictionary)

5. 딕셔너리 관련 함수

- Key 리스트 만들기(keys)

```
>>> a = {'name': 'pey', 'phone': '0119993323', 'birth': '1118'}
>>> a.keys()
dict_keys(['name', 'phone', 'birth'])
```

- Value 리스트 만들기(values)

```
>>> a.values()
dict_values(['pey', '0119993323', '1118'])
```

- Key, Value 쌍 얻기(items)

```
>>> a.items()
dict_items([('name', 'pey'), ('phone', '0119993323'), ('birth', '1118')])
```

- 딕셔너리(dictionary)

  5. 딕셔너리 관련 함수

     - Key로 Value얻기(get)

```
>>> a = {'name':'pey', 'phone':'0119993323', 'birth': '1118'}
>>> a.get('name')
'pey'
>>> a.get('phone')
'0119993323'
```

연습문제 1: a={‘국어’:90, ‘수학’:80, ‘영어’:70} 인 딕셔너리에서 수학에 해당하는 값을 추출해보시오.

- **집합(set)**

1. 집합 자료형은 집합에 관련된 것을 쉽게 처리하기 위해 만든 자료형

```
>>> s1 = set([1,2,3])
>>> s1
{1, 2, 3}
```

```
>>> s2 = set("Hello")
>>> s2
{'e', 'H', 'l', 'o'}
```

2. 집합 자료형의 특징

- 순서가 없다.
- 중복을 허용하지 않는다.

→ set의 특징을 이용하여 자료형의 중복을 제거하기 위한 필터 역할로도 쓰인다.

```
>>> s1 = set([1,2,3])
>>> l1 = list(s1)
>>> l1
[1, 2, 3]
>>> l1[0]
1
>>> t1 = tuple(s1)
>>> t1
(1, 2, 3)
>>> t1[0]
1
```

- **집합(set)**
  연습문제 1: a=[1,1,1,2,2,3,3,4,5,5,5] 인 리스트에서 중복숫자를 제거해보자.

- **제어문**
  - 프로그래밍에서 조건을 판단하여 해당 조건에 맞는 상황을 수행하는데 사용

```
>>> money = True
>>> if money:
...     print("택시를 타고 가라")
... else:
...     print("걸어 가라")
...
택시를 타고 가라
```

- **If 문**
  - 조건문 이란 참과 거짓을 판단하는 문장을 말한다.

```
>>> money = True
>>> if money:
```

- **if 문**
  1. **비교 연산자**

비교연산자	설명
x < y	x가 y보다 작다
x > y	x가 y보다 크다
x == y	x와 y가 같다
x != y	x와 y가 같지 않다
x >= y	x가 y보다 크거나 같다
x <= y	x가 y보다 작거나 같다

```
>>> money = 2000
>>> if money >= 3000:
...     print("택시를 타고 가라")
... else:
...     print("걸어가라")
...
걸어가라
>>>
```

- **if 문**

2. 연산자

연산자	설명
x or y	x와 y 둘중에 하나만 참이어도 참이다
x and y	x와 y 모두 참이어야 참이다
not x	x가 거짓이면 참이다

```
>>> money = 2000
>>> card = True
>>> if money >= 3000 or card:
...     print("택시를 타고 가라")
... else:
...     print("걸어가라")
...
택시를 타고 가라
```

3. in / not in 연산

in	not in
x in 리스트	x not in 리스트
x in 튜플	x not in 튜플
x in 문자열	x not in 문자열

```
>>> pocket = ['paper', 'cellphone', 'money']
>>> if 'money' in pocket:
...     print("택시를 타고 가라")
... else:
...     print("걸어가라")
...
택시를 타고 가라
```

- **if 문**

4. elif문

- 조건을 판단해야 하는 부분이 두개 이상일 때 사용한다.

```
>>> pocket = ['paper', 'handphone']
>>> card = True
>>> if 'money' in pocket:
...     print("택시를 타고가라")
... else:
...     if card:
...         print("택시를 타고가라")
...     else:
...         print("걸어가라")
...
택시를 타고가라
```

- **while 문(반복문)**
  1. 조건문이 참인 동안에 while 문 아래의 문장이 반복해서 수행된다.

```
>>> treeHit = 0
>>> while treeHit < 10:
...     treeHit = treeHit +1
...     print("나무를 %d번 찍었습니다." % treeHit)
...     if treeHit == 10:
...         print("나무 넘어갑니다.")
...
나무를 1번 찍었습니다.
나무를 2번 찍었습니다.
나무를 3번 찍었습니다.
나무를 4번 찍었습니다.
나무를 5번 찍었습니다.
나무를 6번 찍었습니다.
나무를 7번 찍었습니다.
나무를 8번 찍었습니다.
나무를 9번 찍었습니다.
나무를 10번 찍었습니다.
나무 넘어갑니다.
```

- **while 문**

  2. break문

  - while문이 참인 동안 반복적으로 수행 될 때 그만 수행하게 하고싶을때 사용

```
>>> coffee = 10
>>> money = 300
>>> while money:
...     print("돈을 받았으니 커피를 줍니다.")
...     coffee = coffee -1
...     print("남은 커피의 양은 %d개입니다." % coffee)
...     if coffee == 0:
...         print("커피가 다 떨어졌습니다. 판매를 중지합니다.")
...         break
```

- **while 문**

  3. continue 문

  - while문 안의 문장이 수행될 때, break가 아닌 맨 처음으로 돌아가게 할 때 사용한다.

```
>>> a = 0
>>> while a < 10:
...     a = a + 1
...     if a % 2 == 0: continue
...     print(a)
...
1
3
5
7
9
```

- **for 문**

1. for문은 while문과 비슷하지만, 문장 구조를 쉽게 확인할 수 있어 유용하다.

```
>>> test_list = ['one', 'two', 'three']
>>> for i in test_list:
...     print(i)
...
one
two
three
```

2. range 함수

```
>>> a = range(1, 11)
>>> a
range(1, 11)
```

```
>>> add = 0
>>> for i in range(1, 11):
...     add = add + i
...
>>> print(add)
55
```