# 파이썬 기초 문법

# 1. 자료형 (data type) – 숫자형 (1/2)

- 자료형이란 프로그래밍을 할 때 쓰이는 숫자, 문자열 등 자료 형태로 사용하는 모든 것을 의미
- 정수형(int type)과 실수형(float type)

1.1.정수형과_실수형_판단.py

```
x = 1 # 변수 x에 1이라는 값을 대입 (C, 자바 등과 다르게 변수 타입을 미리 선언할 필요가 없음)
y= 2.2 # 변수 y에 2.2라는 값을 대입
z = 3.0 # 변수 z에 3.0이라는 값을 대입

print(type(x), type(y), type(z)) # x, y, z의 자료형을 출력
```

출력 결과: <class 'int'> <class 'float'> <class 'float'>

- print(내용): 내용을 출력함
- type(변수): 해당 변수의 자료형을 반환

# 1. 자료형 (data type) – 숫자형 (2/2)

- 사칙연산

1.2.사칙연산.py

```
x = 2
y = 2.2
z = 3
a = x + y# a라는 변수에 x와 y의 합을 저장
b = x - z
c = x ** z # 제곱 연산자
d = 7/4 # 나눗셈 연산자
e = 7%3 # 나머지 연산자

print(a, b, c, d, e)
```

출력 결과:`4.2 -1 8 1.75 1`

# 1. 자료형 (data type) – 문자형 (1/2)

- 문자열 생성 방법

1. 큰따옴표로 양쪽 둘러싸기
(예) "Hello World"

2. 작은따옴표로 양쪽 둘러싸기
(예) 'Python is fun'

3. 큰따옴표 3개를 연속으로 써서 양쪽 둘러싸기
(예) """Life is too short, You need python"""

4. 작은따옴표 3개를 연속으로 써서 양쪽 둘러싸기
(예) '''Life is too short, You need python'''

- 문자열 인덱싱

a = "hello! world"

| h | e | l | l | o | ! |  | w | o | r | l | d | ! |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |

1.3.문자열_인덱싱.py

```
a = "hello!world"
print(a[0], a[1], a[2]) # 문자열 a의 0, 1, 2번째 요소를 출력
print(a[-1], a[-2], a[-3]) # 문자열 a의 -1, -2, -3번째 요소를 출력
```

출력 결과:
```
h e l
d l r
```

# 1. 자료형 (data type) –문자형 (2/2)

- 문자열 슬라이싱

1.4.문자열_슬라이싱.py

```
a = “hello!world”
print(a[0:4]) # 문자열 a의 0번째부터 4번째까지 요소전까지를 출력
print(a[3:]) # 문자열 a의 3번째부터 맨 끝까지 요소를 출력
print(a[:4]) # 문자열 a의 처음부터 4번째 요소전까지 출력
```

출력 결과:
```
hell
lo! world
hell
```

- 기계학습(특히, 텍스트 분석)에서 많이 사용되는 문자열 관련 함수

1.5.문자열_관련_함수.py

```
a = “hello! world”
x = a.count(‘l’) # 문자 개수 세기 함수: count
y = a.index(‘e’) # 문자 위치 반환 함수: index
z = a.find(‘e’) # 문자 위치 반환 함수: find
w = a.split(‘ ‘) # 문자열 나누기 함수: split
print(x, y, z, w)
```

출력 결과:
```
3 1 1 ['hello!', 'world']
```

# 1. 자료형 (data type) – 리스트형 (1/3)

- 리스트는 자료형을 저장하는 자료형이며, 기계학습에서 사용하는 데이터의 특성상 굉장히 자주 사용하게 될 자료형임

1.6.리스트의_생성.py

```
a = [ ] # 빈 리스트
b = [1, 2, 3] # 숫자를 요소로 가지는 리스트
c = ['Life', 'is', 'too', 'short’] # 문자열을 요소로 가지는 리스트
d = [1, 2, 'Life', 'is’] # 숫자와 문자열을 요소로 가지는 리스트
e = [1, 2, ['Life', 'is’]] # 숫자와, 문자열을 요소로 가지는 리스트를 요소로 가지는 리스트
```

- 리스트의 인덱싱과 슬라이싱

1.7.리스트의_인덱싱_슬라이싱.py

```
a = [0, 1, 2, 3, 4, 5, 6]
print(a[1]) # a의 첫 번째 요소
print(a[3:6]) # a의 세 번째 요소부터 여섯 번째 이 전 값
b = [0, 1, 2, [3, 4, 5]] # 중첩 리스트
print(b[3][1]) # b의 세 번째 요소에서 첫 번째 요소
```

출력 결과:

```
1
[3, 4, 5]
4
```

# 1. 자료형 (data type) – 리스트형 (2/3)

- 리스트 연산자

1.8.리스트_연산자.py

```
[1, 2, 3] + [4, 5, 6] # [1, 2, 3, 4, 5, 6]
[1, 2, 3] * 3 # [1, 2, 3, 1, 2, 3, 1, 2, 3]
```

- 리스트의 수정

1.9.리스트의_수정.py

```
a = [1, 2, 3]
a[2] = 4
print(a)
del a[1] # a의 1번째 요소 삭제
print(a)
```

출력 결과:

```
[1, 2, 4]
[1, 4]
```

# 1. 자료형 (data type) – 리스트형 (3/3)

- 리스트 관련 함수

1.10.리스트_관련_함수.py

```
a = [1, 2, 3]
a.append(4) # .append(x): 요소 x를 맨 뒤에 추가
print(a) # [1, 2, 3, 4]
b = [1, 4, 3, 2]; c = [1, 4, 3, 2]
b.sort() # sort(): 정렬
print(b) # [1, 2, 3, 4], [2, 3, 4, 1]
b.sort(reverse = TRUE)
print(b)
print(a.index(3)) # index(): 위치 반환 함수
a.remove(3)
print(a)
print(a.count(1))
```

출력 결과:

```
[1, 2, 3, 4]
[1, 2, 3, 4]
[4, 3, 2, 1]
2
[1, 2, 4]
1
```

# 2. 제어문 –If 문 (1/4)

- if문의 기본 구조

```
if 조건문 1:
        수행할 문장 1-1
        …
elif 조건문 2:
        수행할 문장 2-1
        …
elif 조건문 3:
        수행할 문장 3-1
        …
else:
        수행할 문장 4-1
        …
```

- 조건문 1이 참이면 문장 1-1, 1-2, …를 수행
- 조건문 1이 참이 아니면 elif절을 확인
  - ➢ 조건문 2가 참이면 문장 2-1, 2-2, …를 수행
- if와 elif에 있는 모든 조건문이 참이 아니면 else절을 확인
- elif와 else는 없어도 작동하며, elif 절의 개수는 제약이 없음

- 조건문 뒤에는 콜론(:)이 반드시 붙어야 함
- 수행할 문장은 조건문에 대해 들여쓰기를 해야 함 (들여쓰기 수준은 반드시 동일해야 함)
  - ➢ 탭 vs 스페이스 2 ~ 4개 (개인적으로 탭을 선호)

# 2. 제어문 –If 문 (2/4)

- 조건문 작성
  - 자료형의 참: 0, [], {}, (), “”, False 외에 모든 값
  - 연산자

| 구분 | 연산자 | 설명 |
|---|---|---|
| 비교 연산자 | 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 |
| in 연산자 | x in 리스트 | x가 리스트에 포함된다 |
| | x not in 리스트 | X가 리스트에 포함되지 않는다 |

# 2. 제어문 –If 문 (3/4)

- 조건문 예시

2.1.조건문_예시.py

```
money = 4000
if money > 5000:
        print("모범 택시를 타라")
elif money > 3000:
        print("일반 택시를 타라")
elif money > 1000:
        print("전철을 타라")

else:
        print("걸어가라")
```

출력 결과: 일반 택시를 타라

- 조건문 퀴즈: 점수(score)가 75점인 학생의 등급(grade)을 출력하라. 단, 점수가 90점 이상이면 A, 80점 이상 90점 미만이면 B, 70점 이상 80점 미만이면 C, 70점 미만이면 F이다.

# 2. 제어문 –If 문 (4/4)

2.2.조건문_퀴즈_정답.py

```
score = 75
if score >= 90:
  grade =  "A"  elif
    score >= 80:
     grade =  "B"
elif score >= 70:
    grade =  "C"
else:
    grade =  "D"
print(grade)
```

출력 결과: C

# 2. 제어문 –for 문 (1/3)

- for문의 기본 구조

```
for 변수 in 리스트 (또는 튜플, 문자열):
        수행할 문장 1
        수행할 문장 2
        …
```

리스트에 포함된 모든 요소를 탐색할 때까지 문장을 반복 수행

- for문 예시

2.3.for문_예시.py

```
L = [1, 2, 3, 4, 5]
S = 'Hello'
for i in L:
        print(i)
for i in S:
        print(i)
```

출력 결과 :

# 2. 제어문 –for 문 (2/3)

- range 함수

2.4.for문_range_함수.py

```
for i in range(3):
        print(i)
for j in range(1, 3):
        print(j)
for k in range(1, 10, 3):
        print(k)
```

출력 결과:

```
0
1
2
1
2
1
4
7
```

2.5_nested_for문.py

```
for i in range(5):
        for j in  range(5):
                print(i, j)
```

출력 결과:

```
0 0
0 1
0 2
0 3
0 4
1 0
1 1
1 2
1 3
1 4
2 0
2 1
2 2
2 3
2 4
3 0
3 1
3 2
3 3
3 4
4 0
4 1
4 2
4 3
4 4
```

# 2. 제어문 –for 문 (3/3)

- for문 quiz: 다음과 같이 구구단을 출력하라 (Hint: print(i, “*”, j, “=“, i * j)를 사용하라)

```
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
```

Quiz 정답: 2.6_for문_퀴즈_정답.py

# 2. 제어문 –while 문 (1/3)

- while문의 기본 구조

```
while 조건문:
        수행할 문장 1
        수행할 문장 2
        …
```

조건문을 만족하면 while문 아래에 속하는 문장들을 반복 수행

무한 루프 주의

- while문 예시

2.7.while문_예시.py

```
treeHit = 0
while treeHit < 10:
        treeHit = treeHit + 1 # treeHit 변수를 1만큼 증가
        print("나무를 ", treeHit, "번 찍었습니다")
```

출력 결과:

```
나무를  1 번 찍었습니다
나무를  2 번 찍었습니다
나무를  3 번 찍었습니다
나무를  4 번 찍었습니다
나무를  5 번 찍었습니다
나무를  6 번 찍었습니다
나무를  7 번 찍었습니다
나무를  8 번 찍었습니다
나무를  9 번 찍었습니다
나무를  10 번 찍었습니다
```

# 2. 제어문 –while 문 (2/3)

- while문 예시 설명

| treeHit | 조건문 | 조건판단 | 수행하는 문장 |
|---|---|---|---|
| 0 | 0 < 10 | 참 | 나무를 1번 찍었습니다. |
| 1 | 1 < 10 | 참 | 나무를 2번 찍었습니다. |
| 2 | 2 < 10 | 참 | 나무를 3번 찍었습니다. |
| 3 | 3 < 10 | 참 | 나무를 4번 찍었습니다. |
| 4 | 4 < 10 | 참 | 나무를 5번 찍었습니다. |
| 5 | 5 < 10 | 참 | 나무를 6번 찍었습니다. |
| 6 | 6 < 10 | 참 | 나무를 7번 찍었습니다. |
| 7 | 7 < 10 | 참 | 나무를 8번 찍었습니다. |
| 8 | 8 < 10 | 참 | 나무를 9번 찍었습니다. |
| 9 | 9 < 10 | 참 | 나무를 10번 찍었습니다. |
| 10 | 10 < 10 | 거짓 | - |

# 2. 제어문 –while 문 (3/3)

- while문 Quiz: 구구단을 작성하라 (Quiz 정답: 2.8_while문_퀴즈_정답.py)

# 3. 모듈 (1/2)

- 모듈 불러오기
  - import 모듈명
  - import 모듈명 as 약어
  - from 모듈명 import 함수
  - from 모듈명 import *

3.1.모듈_불러오기.py

```
import math
x =  math.sqrt(4)
import math as m
 y= m.sqrt(4)
from math import sqrt
z =  sqrt(4)
from math import *
w =  sqrt(4)
```

# 3. 모듈 (2/2)

- 기계학습에 자주 사용되는 다섯가지 모듈

# 4. 파일 읽고 쓰기 (1/4)

- 파일 쓰기

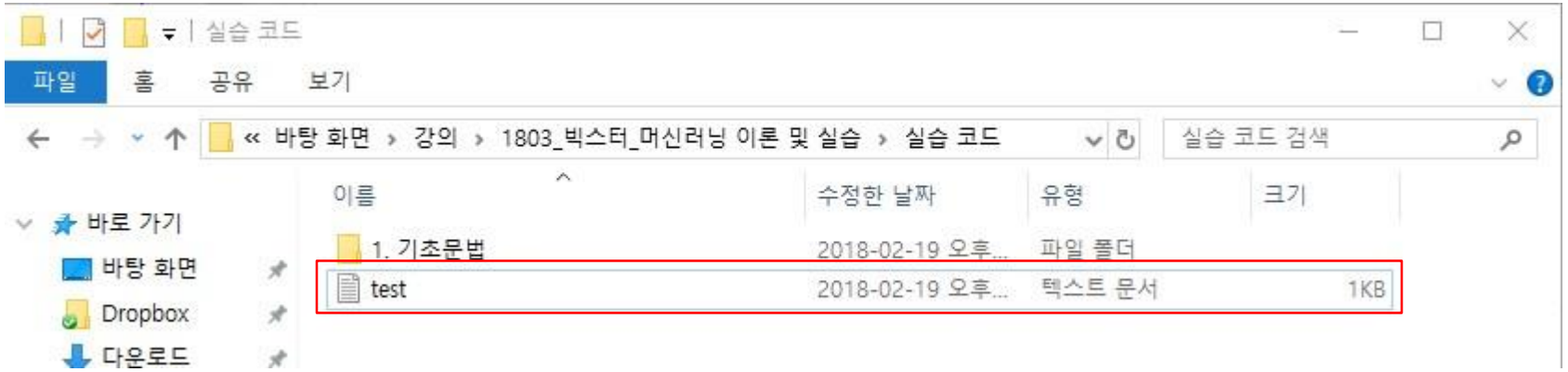
4.1.파일_쓰기.py

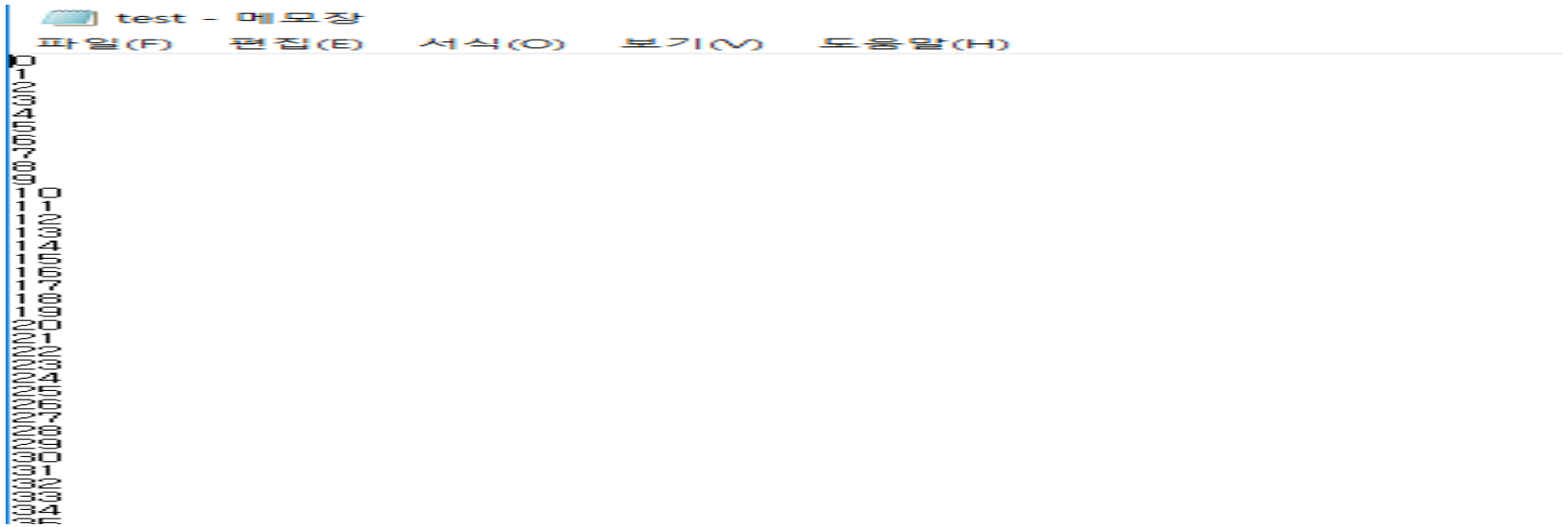
```
f = open("test.txt", "w")       # w: writing file, r: reading file
for i in range(100):
        f.write(str(i)) # str(x); x를 문자열로 변경하는 함수. 문자열만 파일에 입력 가능
        f.write('\n')
f.close()
```

- 이미 있는 파일명으로 라이팅을 하면, 그 파일에 자동으로 덮어써지므로 주의해야 함

# 4. 파일 읽고 쓰기 (2/4)

- 파일 쓰기



# 4. 파일 읽고 쓰기 (3/4)

- 파일 읽기

4.2.파일_읽기.py

```
f =  open("test.txt","r") # w: writing file, r: reading file
for i in range(100):
        print(f.readline())

f =  open("test.txt", "r")
full_text =  f.read()
print(full_text)
```

# 4. 파일 읽고 쓰기 (4/4)

- 파일 읽기: pandas.read_csv

4.3.pandas_파일_읽기.py

```
import pandas as pd
df =  pd.read_csv("preprocessed_redwine_quality.csv", encoding = "cp949")
df
```

```
In [61]: df
Out[61]:
      고정산도  휘발성산도  구연산  잔류설탕    염화물  자유이산화황  총이산화황      밀도    pH  황산염  \
0      7.4  0.700  0.00   1.9  0.076    11.0    34.0  0.99780  3.51  0.56
1      7.8  0.880  0.00   2.6  0.098    25.0    67.0  0.99680  3.20  0.68
2      7.8  0.760  0.04   2.3  0.092    15.0    54.0  0.99700  3.26  0.65
3     11.2  0.280  0.56   1.9  0.075    17.0    60.0  0.99800  3.16  0.58
4      7.4  0.700  0.00   1.9  0.076    11.0    34.0  0.99780  3.51  0.56
5      7.4  0.660  0.00   1.8  0.075    13.0    40.0  0.99780  3.51  0.56
6      7.9  0.600  0.06   1.6  0.069    15.0    59.0  0.99640  3.30  0.46
7      7.3  0.650  0.00   1.2  0.065    15.0    21.0  0.99460  3.39  0.47
8      7.8  0.580  0.02   2.0  0.073     9.0    18.0  0.99680  3.36  0.57
9      7.5  0.500  0.36   6.1  0.071    17.0   102.0  0.99780  3.35  0.80
```

- 데이터 분석에는 csv 파일을 사용하는 것이 좋으며, pandas 외에도 데이터를 분석할 수 있는 패키지가 많지만 가장 손쉽게 다룰 수 있는 모듈이 pandas임