강원대학교
AI 소프트웨어학과

인공지능
- 입출력 및 변수 형태 -

KNU
강원대학교

자료형 : 숫자 자료형 (예: 1, 0, -10)

문자 자료형 (예 : 작은 따옴표 안의 문자, 큰 따옴표 안의 문자)

숫자 자료형 (예 : “3”, “5.1”)

불린 자료형 (예 : True, False)

: → 콜론

“ ”, ‘ ’ → 큰따옴표, 작은따옴표

, →쉼표

01 입출력

```
a=1
b=2
c=3
d=4
```

ex) a = 1

변수 a 에 1을 저장

```
a+b, a-b, a*c
```

```
(3, -1, 3)
```

```
print(a+b)
print(a+b, a-b, a*c)
```

```
3
3 -1 3
```

% : 나머지 연산

ex) 5%2 = 1

5/2 했을 때, 몫 2, 나머지 1

```
print(a*b)
print(d/b)
print(d%c)
```

```
2
2.0
1
```

01 입출력

```
In [97]: a=input()
         10
```

```
In [98]: type(a)
Out[98]: str
```

```
In [99]: a=int(input())
         10
```

```
In [100]: type(a)
Out[100]: int
```

```
In [101]: a=float(input())
          10
```

```
In [102]: type(a)
Out[102]: float
```

Input을 이용해 변수를 저장할 수 있음

이때, 변수들의 type을 정확하게 지정해 주는 것이 중요함

01 입출력

코드	설명
`\n`	문자열 안에서 줄을 바꿀 때 사용
`\t`	문자열 사이에 탭 간격을 줄 때 사용
`\\`	문자 `\`를 그대로 표현할 때 사용
`\'`	작은따옴표(`'`)를 그대로 표현할 때 사용
`\"`	큰따옴표(`"`)를 그대로 표현할 때 사용
`\r`	캐리지 리턴(줄 바꿈 문자, 현재 커서를 가장 앞으로 이동)
`\f`	폼 피드(줄 바꿈 문자, 현재 커서를 다음 줄로 이동)
`\a`	벨 소리(출력할 때 PC 스피커에서 '삑' 소리가 난다)
`\b`	백 스페이스
`\000`	널 문자

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

이중에서 활용빈도가 높은 것은 `\n`, `\t`, `\\`, `\'`, `\"`이다. 나머지는 프로그램에서 잘 사용하지 않는다.

01 입출력

```
a=123
b=-123
c=0
```

정수형 : 소수점으로 표현하지 않는 수. Python에서 int로 정수를

```
type(a)
```

```
int
```

```
a=1.2
b=-2.1
```

실수형 : 소수점으로 표현해야 하는 수 Python에서 float로 실수를 표현

```
type(b)
```

type(a) : a의 자료형을 반환함. 예시 : int, float, str, object 등

```
float
```

```
a=1.24e-2
```

en(n은 숫자) : 10의 n승을 의미함

예시: 1.24e-2 : 1.24*10^-2 = 0.0124

```
a
```

```
0.0124
```

```
b=1.24E2
```

```
b
```

```
124.0
```

```
a+b #더하기 연산
```

```
124.0124
```

```
a-b #빼기 연산
```

```
-123.9876
```

```
a*c #곱하기 연산
```

```
0.0
```

```
a**c #제곱 연산
```

```
1.0
```

```
7%3 #나머지 반환
```

```
1
```

```
3%7 #나머지 반환
```

```
3
```

```
7/4 #나누기
```

```
1.75
```

```
7//4 #몫을 반환
```

```
1
```

Boolean 타입

- True, False의 값을 갖는 자료형
- None, 공백, 0인 경우에 False이고 이외의 값은 True
- 비교연산자, 논리연산자의 결과값으로 반환됨

```
[18] 10<15
True

[19] 10<=5
False

[20] 10>61
False
```

```
[18] print(bool(1))
True

[19] print(bool(0))
False

[20] print(())
()

[21] print(bool(""))
False

[22] print(bool("안녕"))
True

[23] 1 == 1
True

[24] 1 == 0
False

[30] 1!=1 # 같지 않다
```

조건이 맞을 시에는 True 반환

조건이 맞지 않을 시에는 False 반환

a = b : a에 b를 대입한다(변수선언)

a==b : a와 b는 같다(조건식)

a != b : a와 b는 같지 않다

Boolean 타입

- True, False의 값을 갖는 자료형
- None, 공백, 0인 경우에 False이고 이외의 값은 True
- 비교연산자, 논리연산자의 결과값으로 반환됨

```
[31]  not 1 == 1 # 같지 않다
```

```
False
```

not a == b : 두 숫자가 같지 않을 때

```
1 == 1 and 1 == 2
```

```
False
```

and : 두 조건이 모두 만족

or : 두 조건중 하나만 만족

```
[33]  1 == 1 or 1 == 2
```

```
True
```

01 입출력

```
"Life is too short, You need Python"
```

```
'Life is too short, You need Python'
```

1

```
"a"
```

```
'a'
```

2

```
"123"
```

```
'123'
```

```
food = "Python's favorite food is perl"
```

3

```
type(food)
```

```
str
```

```
python='python'
```

```
a="python"
b=2
```

4

```
a*b
```

5

```
'pythonpython'
```

```
print("="*50)
print("My Program")
print("="*50)
```

```
==================================================
My Program
==================================================
```

1 str(문자형) : Python에서 문자형을 표현할 때 큰따옴표 혹은 작은따옴표 안에 문자를 넣음

str형 * int형 : str형인 문자를 int형 숫자만큼 반복 5

2 따옴표 안에 숫자를 넣으면 숫자(int, float)로 인식 안함.
문자로 인식

3 type(a) : a의 자료형을 반환함.
문자의 자료형 : str

02 슬라이싱

문자열은 슬라이싱을 사용할 수 있음

슬라이싱 : 연속적인 객체들에(예: 리스트, 튜플, 문자열) 범위를 지정해 선택해서 객체들을 가져오는 방법 및 표기법을 의미함

A="안녕하세요 지금은 인공지능 수업 시간입니다."

0 1 2 3 4 5 6 7 8 9 10 … … … … … … … … … … … … 23

문자열은 슬라이싱을 사용할 수 있음

슬라이싱 : 연속적인 객체들에(예: 리스트, 튜플, 문자열) 범위를 지정해 선택해서 객체들을 가져오는 방법 및 표기법을 의미함

A="안녕하세요 지금은 인공지능 수업 시간입니다."

A[시작범위:직전범위]

A[0:4] = 안녕하세

문자열은 슬라이싱을 사용할 수 있음

슬라이싱 : 연속적인 객체들에(예: 리스트, 튜플, 문자열) 범위를 지정해 선택해서 객체들을 가져오는 방법 및 표기법을 의미함

A="안녕하세요 지금은 인공지능 수업 시간입니다."

A[-1] = . → 맨뒤는 0으로 할 수 없으므로 -1 부터

A[-5:] = 간입니다.

02 슬라이싱

```
print("안녕하세요₩n반갑습니다")
```

₩n : 줄바꿈

```
안녕하세요
반갑습니다
```

Int 형 : %d, str형 : %s, float형 : %f

```
print("이번시험의 성적으로 %d점을 맞았습니다."%3)
```

ex) "~%d~" %3 : %d 자리에 3을 넣어 줌

```
이번시험의 성적으로 3점을 맞았습니다.
```

```
print("이번시험의 성적으로 %0.3f점을 맞았습니다."%3.333333)
```

%0.3f : 소수점 3번째까지만 보여줌

```
이번시험의 성적으로 3.333점을 맞았습니다.
```

```
print(("이번시험의 성적으로 {}점을 맞았습니다.").format(100))
```

```
이번시험의 성적으로 100점을 맞았습니다.
```

```
print(("이번시험에서 수학은 {}점 영어는 {}점 국어는 {}점을 맞았습니다.").format(100,60,60))
```

```
이번시험에서 수학은 100점 영어는 60점 국어는 60점을 맞았습니다.
```

('~{}~{}~{}').format(a,b,c)

{}의 개수와 format뒤 ()안의 매개변수 개수가 같아야 함

맨 앞 {}부터 차례로 format뒤의 값을 넣어줌

format : 문자열을 포매팅 하는 방법으로 문자열 중간중간 특정 변수의 값을 넣어주기 위해 사용되는 것

format(중괄호 사이에 들어갈 값)

A="I am a "
B="boy"

A+B➔I am a boy

A="I am a {}"
B="boy"

A.format(B)

format은 print에서도 활용되고 단순 문자열에서도 사용됨

format(중괄호 사이에 들어갈 값)

print("{}, {}").format(첫번째 값, 두번째 값)

"I am a {}".format(첫번째 값)

format은 print에서도 활용되고 단순 문자열에서도 사용됨

```
print("{:.2f}, {:.3f}".format(3.1415, 3.1415)
```

3.14, 3.141

Replace : 특정 문자열을 찾아서 다른 문자열로 대체 가능함

변수.replace(“기존의 문자”, “변환하고 싶은 문자”)

```
url = "https://www.youtube.com/"

str = url.replace("https://", "")
```

Replace : 특정 문자열을 찾아서 다른 문자열로 대체 가능함

```
Index_str=str[:str.index("직전까지 문자")]

a="독고영재, 50"    b="김철수, 30"

a[:a.index(",")]    b[:b.index(",")]
```

리스트 : 하나로 변수로 표현했던 숫자나 문자열을 담는 주머니

리스트명 = [요소1, 요소2, 요소3, ...]

a="독고영재, 50"

b="김철수, 30"

리스트명 = ["독고영재, 50", "김철수, 30",...]

리스트 : 하나로 변수로 표현했던 숫자나 문자열을 담는 주머니

리스트명 = [요소1, 요소2, 요소3, ...]

리스트는 문자와 같이 각각의 공간을 가진다.

A="안녕하세요 지금은 인공지능 수업 시간입니다."

0 1 2 3 4 5 6 7 8 9 10 23

A = [10, 20, 30, ...]

0 1 2 ...

리스트 : 하나로 변수로 표현했던 숫자나 문자열을 담는 주머니

A = [10, 20, 30, 40, 50, 60, 70, 80, 90]

A[시작범위:직전범위]

A[0:4] = 안녕하세

A[시작범위:직전범위]

A[0:4] = [10, 20, 30, 40]

리스트 : 하나로 변수로 표현했던 숫자나 문자열을 담는 주머니

A = [10, 20, 30, 40, 50, 60, 70, 80, 90]

A[0] = [10]

A[5] = [60]

리스트 : 하나로 변수로 표현했던 숫자나 문자열을 담는 주머니

A = ["독고영재, 50", “김철수, 30“,…]

A=“안녕하세요 지금은 인공지능 수업 시간입니다.”

A[시작범위:직전범위]

A[0] = 독고영재, 50　　A[0][0:4] = 독고영재

리스트 : 하나로 변수로 표현했던 숫자나 문자열을 담는 주머니

a = ["독고영재, 50", “김철수, 30“,…]

a[:a.index(",")] b[:b.index(",")]

어떻게 바꿀까?

리스트는 다양한 type을 가질 수 있음

```
a=["hi", 10, "80","김철수", 50, 50.6]
```

type(a[1]) → int

type(a[3]) → str

type(a[5]) → float

리스트를 추가하는 방법

```
a=["hi", 10, "80","김철수", 50, 50.6]

a.append(추가하고 싶은 새로운 값)

a.append(1)

a=["hi", 10, "80","김철수", 50, 50.6, 1]
```

리스트의 값을 삭제하는 방법

```
a=["hi", 10, "80","김철수", 50, 50.6]

a.pop(제거하고 싶은 값의 위치)

a.pop(1)

a=["hi", "80","김철수", 50, 50.6, 1]
```

리스트의 값을 삭제하는 방법

```
a=["hi", 10, "80","김철수", 50, 50.6]

a.index(50)

a.pop(4)

a=["hi", 10, "80","김철수", 50.6]
```

리스트의 값을 삭제하는 방법

```
a=["hi", 10, "80","김철수", 50, 50.6]

b=a.index(“80”)

a.pop(b)

a=["hi", 10,"김철수", 50, 50.6]
```

딕셔너리 : 하나의 키 값을 정해줘 키 값 안에 변수를 넣어주는 방법

A={키값: 변수값, 키값: 변수값}

A[키값] = 변수값
A[키값] = 변수값

딕셔너리 : 하나의 키 값을 정해줘 키 값 안에 변수를 넣어주는 방법

A={1: “김철수”, 20: “박영희”}

A[1] = 김철수
A[20] = 박영희
A.get(1) = 김철수
A.get(3) = None

딕셔너리의 키값을 설정하는것에 있어 꼭 수치값이 아니라도 괜찮음

A={"1-A": "김철수", "20-B": "박영희"}

A["1-A"] = 김철수
A["20-B"] = 박영희
A.get("1-A") = 김철수
A.get("20-B") = 박영희

딕셔러리의 추가 삭제

A={"1-A": "김철수", "20-B": "박영희"}

추가 : A["2-A"] = "영희"

삭제 : del A["2-A"]