4주차

python에서 딥러닝까지

Python 기초반

주제 | 연산자(1)~리스트

Contents

Next

연산(1)

산술연산자 - 교재 35p

- 여러가지 연산자
  - 더하기: +
  - 빼기: -
  - 곱하기: *
  - 나누기
    - 실수 몫: /
    - 정수 몫: //
  - 나머지: %
  - 거듭제곱: **

- 연산자에는 우선순위가 있다.
- 애매하면 먼저 계산할 식을 ( )로 묶어주자

우선순위	연산자	설명
1	(값...), [값...], {키: 값...}, {값...}	튜플, 리스트, 딕셔너리, 세트 생성
2	x[인덱스], x[인덱스:인덱스], x(인수...), x.속성	리스트(튜플) 첨자, 슬라이싱, 함수 호출, 속성 참조
3	await x	await 표현식
4	**	거듭제곱
5	+x, -x, ~x	단항 덧셈(양의 부호), 단항 뺄셈(음의 부호), 비트 NOT
6	*, @, /, //, %	곱셈, 행렬 곱셈, 나눗셈, 버림 나눗셈, 나머지
7	+, -	덧셈, 뺄셈
8	<<, >>	비트 시프트
9	&	비트 AND
10	^	비트 XOR
11	|	비트 OR
12	in, not in, is, is not, <, <=, >, >=, !=, ==	포함 연산자, 객체 비교 연산자, 비교 연산자
13	not x	논리 NOT
14	and	논리 AND
15	or	논리 OR
16	if else	조건부 표현식
17	lambda	람다 표현식

문자열 연산

- 문자열 더하기
  - python에서는 문자열끼리 더할 수 있다.
  - [문법] <문자열1> + <문자열2> (ex. “안녕하세요.” + “반갑습니다“)
    결과: “문자열1문자열2“(ex. “안녕하세요.반갑습니다”)
- 문자열 곱하기
  - 나아가, python에서는 문자열을 곱할 수도 있다.
  - [문법] <문자열> * 정수 (ex. “정신나갈꺼같아” * 3)
    결과: <문자열을 정수만큼 반복>
    (ex. “정신나갈꺼 같아정신나갈꺼같아정신나갈꺼같아”)

복합할당 연산자

- 정의

  : 연산자와 할당연산자를 합쳐 놓은 것.

- 연산자계의 “단축키”라고 생각하면 됨.

연산자명	설명식	복합 대입 연산자	결과
`+=`	a = a + 1	a `+=` 1	4
`-=`	a = a - 1	a `-=` 1	2
`*=`	a = a * 2	a `*=` 2	6
`/=`	a = a / 2	a `/=` 2	1.5
`%=`	a = a % 2	a `%=` 2	1
`//=`	a = a // 2	a `//=` 2	1
`**=`	a = a ** 2	a `**=` 2	9

연산(1) 연습문제 & Mission

- 2nd_week_mission.py의 연산(1) 연습문제 및 Mission을 풀어보자
  - 산술 연산
  - 문자열 연산
  - 복합할당 연산

Next

연산(2)

비교연산(관계 연산) – 교재 74p 참고

- 두 개의 피연산자(변수나 상수)를 “비교”하는 데 사용하는 연산자
- 결과: 참(True) 또는 거짓(False)

연산	의미
x == y	x와 y가 같은가?
x != y	x와 y가 다른가?
x 〉 y	x가 y보다 큰가?
x 〈 y	x가 y보다 작은가?
x 〉= y	x가 y보다 크거나 같은가?
x 〈= y	x가 y보다 작거나 같은가?

논리 연산

- 논리 값을 판단해주는 연산자
- **Ture(참,1)//False(거짓,0)**

Operator	Description	Example
and	논리 AND 연산. 둘다 참일때만 참	(a and b) = False
or	논리 OR 연산. 둘 중 하나만 참이여도 참	(a or b) = True
not	논리 NOT 연산. 논리 상태를 반전	not(a and b) = True

a	b	AND	OR	XOR	NAND	NOR
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	1	0
1	1	1	1	0	0	0

멤버십 연산

- 포함관계를 나타내는 연산(결과 : True / False)
- in : 포함되어 있다.
- not in: 포함되어 있지 않다.

> **in 연산자** : 포함하는지 검사합니다.
> **not in 연산자** : 포함되어 있지 않은지 검사 합니다.

연산자(2) Mission

- 비교 연산
- 논리 연산
- 멤버십 연산

Next

입력과 자료형 변환

자료형 변환 왜 필요한가?

- Mission으로 알아보는 자료형 변환의 필요성
- Mission1: input()를 활용하여 변수 x,y에 숫자를 입력 받고, 두 숫자를 더하여 정수들의 합에 대한 결과를 출력해보자.

```
x를 입력하세요 >>30
y를 입력하세요 >>15
45
```

자료형 변환 왜 필요한가?

- 원하는 값들을 더해지는 것이 아닌 두 문자가 띄어쓰기 없이 순서대로 나열된 결과가 출력되었다.
- **Mission2: input()을 int(input())과 같은 형태로 자료형 변화를 시켜 다시 실행시켜보자**

```
x를 입력하세요 >>30
y를 입력하세요 >>15
45
```

자료형 변환 Mission3

- **Mission**

  : 출생년도를 입력받고 나이를 출력하는 프로그램을 작성해보자.

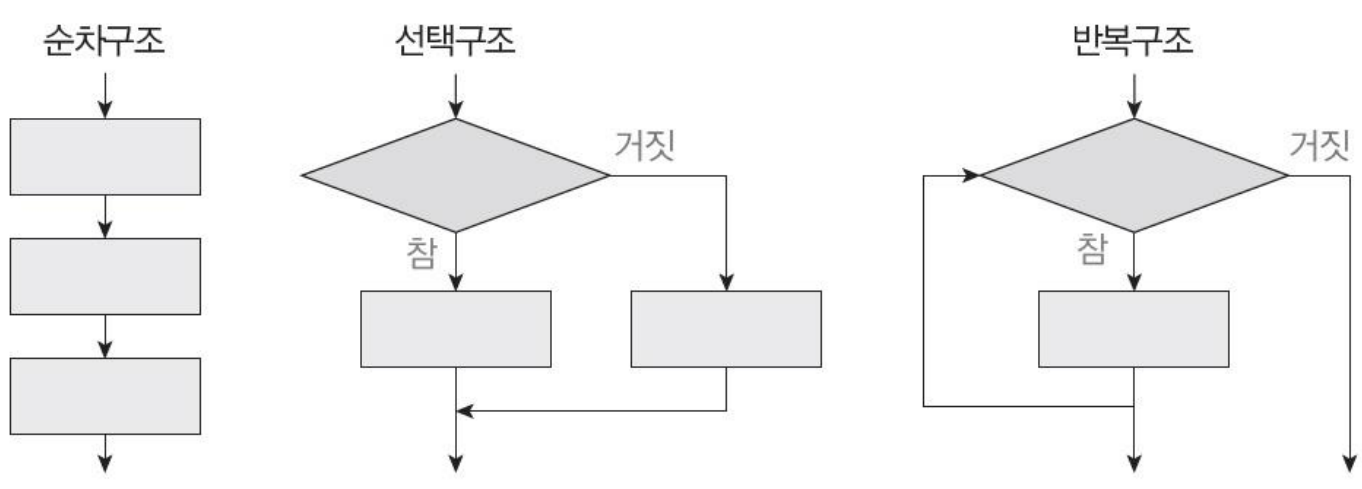
```
출생년도를 입력하세요 >> 1996
당신의 나이는 26입니다.
```

Next

조건문 (교재 77p 참고)

프로그램의 3가지 제어구조

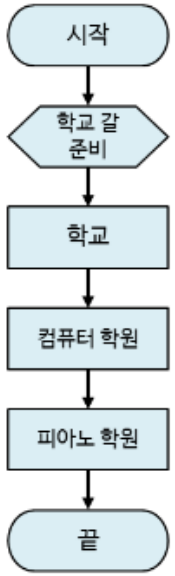
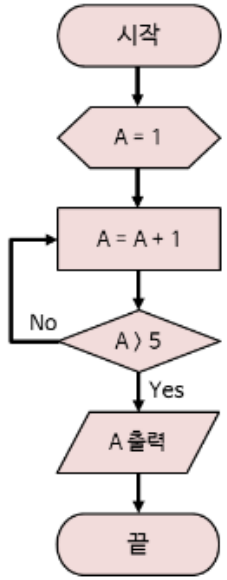
- 순차구조(sequence): 명령어들이 순차적으로 실행되는 구조
- 선택구조(selection): 둘 혹은 다수의 명령어 흐름을 선택하여 실행하는 구조
- 반복구조(iteration): 동일한 명령어들이 반복되며 실행되는 구조
- 위와 같은 제어 구조들은 외우는 것이 아닌 “익숙해 져야 한다”



순서도(flowchart)

- 어떠한 일을 처리하는 과정을 순서대로 간단한 기호와 도형으로 도식화한 것

기호	명칭	설명
시작/종료	단말	순서도의 시작과 끝을 나타냄.
	흐름선	순서도 기호 간의 연결 및 작업의 흐름을 표시함.
준비	준비	작업 단계 시작 전 해야 할 작업을 명시함.
처리	처리	처리해야 할 작업을 명시함.
입출력	입출력	데이터의 입출력 시 사용함.
판단	의사 결정	비교 및 판단에 의한 논리적 분기를 나타냄.
표시	표시	화면으로 결과를 출력함.

1. 여러분의 일과를 순서도로 나타내시오.	2. 변수 A를 1부터 시작하여 1씩 증가시켜 6이 되면 출력하시오.
	

의사코드(pseudo code)

- 컴퓨터 프로그램이나 알고리즘이 수행해야할 내용을 우리가 사용하는 언어로 간략히 서술해 놓은 것
- 주석으로 중간중간 순서대로 표시

```
import turtle

# 변수들 선언
width = 500
height = 500
lineLenght = 100
diffAngle = 90

# 스크린과 거북이 선언
win = turtle.Screen()
win.setup(width, height)
t = turtle.Turtle('turtle')
```

```
# 그림그리기
t.forward(lineLenght)
t.right(diffAngle)
t.forward(lineLenght)
t.right(diffAngle)
t.forward(lineLenght)
t.right(diffAngle)
t.forward(lineLenght)
t.right(diffAngle)

turtle.mainloop()
```

조건문

- 조건문이란?

  : 조건에 따라 명령이 달라지는 제어문

- 제어문 및 함수의 주요 특징

  : 제어문에 포함된 문장은 “띄어쓰기”로 구분한다.

```
if a == 4:
    # 조건이 참이므로 실행됩니다.
    print('한국 사람이 싫어하는 숫자 입니다.')
    print('그래서 4층을 "F"층으로 표기하기도 합니다.')
    print()
```

if 조건문

- if 조건문(Conditional)
  - if: '만약 ~라면(if)', 첫번째 조건
  - elif: '그렇지 않고(else) 만약 ~라면(if)'
  - else: '그렇지 않다면(else)'
  - 조건이 True(1)가 되는지를 판단
- 주의사항
  - 조건 뒤":" (콜론) 반드시 기입
  - 조건 충족 시, 실행할 명령들은 들여쓰기로 구분

```
if 조건A:
    조건A가 참이면 수행할 코드
    ...
    if 조건A2:
        조건A와 조건A2가 모 참이면 수행할 코드
        ...
elif 조건B:
    조건A가 거짓이고, 조건B가 참이면 수행할 코드
    ...
elif 조건C:
    이전 조건들(조건A, 조건B)가 거짓이고, 조건C가 참이면 수행할 코드
    ...
else:
    이전 조건들이 모두 거짓이면 수행할 코드
    ...
```

조건문 연습문제

- 교재 80p “덧셈 문제를 맞히는 프로그램 "
- if-else로 판단하는 프로그램을 작성해 봅시다.
- 조건
  - input을 활용하여 “12+23=“을 출력하고 사용자로부터 답을 입력 받는다.
  - x가 12+23의 결과와 일치하는 경우: “천재!”를 출력
  - x가 12+23과 일치하지 않는 경우: “바보?”를 출력

조건문 Mission1

- **Mission**

  : 구독자 수를 입력 받고 수익창출이 되는지 여부를 판단하는 프로그램을 작성해 보자

```
구독자 수를 입력하세요 >> 995
수익창출을 할 수 없습니다.
```

```
구독자 수를 입력하세요 >> 2365
수익창출을 할 수 있는 계정입니다.
```

조건문 Mission2

- **Mission**
  : 현재 가진 금액을 통해 먹을 수 있는 음식을 출력하는 프로그램을 작성해 보자
  - 20000원 이상: 오늘 저녁은 치킨이닭!
  - 10000원 이상: 이제는 고오급 음식인 떡볶이를 먹으러 가자!
  - 2000원 이상: 그래도 굶지는 않는다! 편의점 삼각김밥!
  - 2000원 미만: 돈이 없다고 굶어야 하는 것은 아니다... 친구에게 “한입만”을 외쳐보자

```
현재 가진 금액을 입력하세요 >> 30000
오늘 저녁은 치킨이닭!!
```

조건문 Mission3

- **Mission**
  : 국어, 영어, 수학 점수를 입력 받고 합계와 평균을 출력한 뒤
  - 평균이 60점이 넘을 경우: 보충 대상자가 아닙니다
  - 평균이 60점을 넘지 않을 경우: 보충 대상자입니다.

  를 출력하는 프로그램을 작성해 보자

```
국어>>> 95
수학>>> 100
영어>>> 85
합계:  280, 평균: 93.33
보충 대상자가 아닙니다. 즐거운 방학보내세요 :D
```

Next

리스트 자료형

리스트 자료형

- 리스트 자료형이란?

  : 데이터가 묶여 일렬로 들어있는 것과 같은 군집형 자료형 중 하나

- 리스트를 사용하는 이유
  - 여러 개의 데이터를 한 번에 저장하기 위해서
  - (여러 개를 한 번에 관리하므로써) 데이터르 효율적으로 관리하고 제어하기 위해서
  - 예시) 아파트 호 수를 방송

리스트 자료형

- 리스트 선언
  - 리스트이름 = [데이터1, 데이터2, …]
  - 빈 리스트도 생성 가능: 리스트 이름 = [ ]
- 리스트 제어1: 데이터 추가하기
  - 리스트이름.append()

※ append: ‘추가’라는 의미

리스트 자료형

- 리스트 제어2: 인덱싱
  - 순서 번호를 가지고 원하는 데이터만 가져오기. 이 때, 인덱스 번호는 0부터 시작
  - 리스트이름[인덱스 번호]
- 리스트 제어3: 데이터 수정하기
  - 리스트이름[인덱스 번호] = 대신 넣을 데이터

리스트 자료형

- 리스트 제어4: 데이터 삭제하기
  - del 리스트이름[삭제할 인덱스 번호]
- 리스트 제어5: 리스트 슬라이싱
  - 리스트이름[시작인덱스:끝인덱스+1]
  - 리스트이름[:끝인덱스+1]
  - 리스트이름[시작인덱스:]
  - 리스트이름[:]

리스트 자료형

- 리스트 제어6: 리스트 길이 구하기
  - len(리스트이름)
- 리스트 제어7: 리스트 정렬하기
  - 오름차순 정렬: 리스트이름.sort()
  - 내림차순 정렬: 리스트이름.sort(reverse=True)

리스트 Mission1

- **Mission**
  : RGB 색상(red, green, blue)을 리스트에 저장하고 turtle 모듈을 활용하여 색상이 서로 다른 직선을 그려보자
- **굵기: 30 // 선길이: 200**

```
color = ['red', 'green', 'blue']
win = turtle.Screen()
win.setup(600, 600)
t = turtle.Turtle('turtle')
t.pensize(30)
```

```
t.penup()
t.setpos(0, 30)
t.pencolor(color[1])
t.pendown()
t.fd(200)
```

리스트 Mission2

- 리스트 Mission2라고 적힌 문제를 풀어보자