#IF 인터페이스 파이썬 스터디

2 회차

템플릿 김태균
강의 백지오

Study Route Map

지난 시간 복습

input 함수를 이용해 사용자로부터 입력을 받을 수 있다.

```
number = int(input( “ give me a number"))
```

int() 함수를 이용하여 문자열 형태의 입력을 정수형으로 바꿔줄 수 있다!

print 구문을 아래와 같이 다양한 형태로 사용할 수 있다.

```
age = 21
print("저는", age, "살 입니다.")
>> 저는 21 살입니다.

print("Hello", end="")
print("World")
>> HelloWorld

n1 = 1
n2 = 2
print("n1=%d, n2=%d" %(n1,n2))
>> n1=1, n2=2
```

변수와 연산자

변수(Variable)이란 숫자나 문자와 같은 데이터를 저장해두기 위해 컴퓨터 메모리 상에 할당된 공간을 의미한다.

저장해야 하는 데이터의 종류(문자, 정수, 실수 등..)와 크기에 따라 데이터 타입이 달라지고, 대부분의 언어에서는 이를 프로그래머가 명시해줘야 한다.

```
int x = 3;          //C언어에서 정수 자료형 int 형의 변수를 선언하고 3을 초기값으로 입력하는 코드
```

그러나, 파이썬에서는 변수 데이터 타입을 명시해주지 않고 변수를 선언한다.
(다만, 이게 데이터 타입이 존재하지 않는다는 의미는 아님!)

```
>>> a = 3
>>> b = 2.1
>>> type(a)
<class 'int'>
>>> type(b)
<class 'float'>
```

숫자 자료형

파이썬에는 숫자를 저장하기 위한 다양한 자료형이 있다.

다루는 숫자	자료형
정수	int
실수	float
복소수	complex

변수에 어떤 값을 넣느냐에 따라 자동으로 알맞은 자료형이 할당된다.

숫자 자료형 - 연산자

파이썬의 숫자 자료형에는 아래와 같은 연산자를 사용할 수 있다.

연산자	기능
+, +=	덧셈
-, -=	뺄셈
*, *=	곱셈
/, /=	나눗셈
%	나머지
//	몫
**	거듭제곱

주의할 점은, C와는 달리 파이썬에서 나눗셈 연산자가 몫이 아닌 실수형(때로는 정수)의 정확한 값을 출력한다는 것이다.

다른 언어에서 나눗셈 연산자(/)와 같은 기능을 원한다면 몫 연산자(//)를 써야한다!

숫자 자료형 – 형변환

Int()나 float()를 활용해서 다른 자료형을 원하는 타입으로 변환할 수 있다.

이때, 실수를 정수로 변환할 때, 소수점 아래의 값은 버림(round)된다.

```
>>> float_num = 3.2
>>> int(float_num)
3
```

숫자 자료형 실습

파이썬의 숫자 자료형을 활용하여 문제를 풀어보자!

1) 사용자로부터 실수 2개를 입력받아 평균을 구하는 프로그램을 만들어보자!

2) 사용자로부터 정수 a, b를 입력받고, a를 b로 나눈 몫과 나머지를 출력하는 프로그램을 만들어보자!

문자열 자료형

파이썬에서 문자나 문자열은 str(String의 약자) 자료형에 저장된다.

```
>>> a = "Life is short,"
>>> b = 'You need python. '
>>> c = """Beautiful is better than ugly."""
>>> d = '''Explicit is better than implicit.'''
```

파이썬에서 문자열은 위 4가지 방법으로 만들 수 있다.

그 이유는 문자열 안에서 큰 따옴표(“)나 작은 따옴표(‘)를 사용할 때 편리성을 위함이다.

또한, 따옴표를 세 번 입력하는 방식은 주로 개행을 포함한 문자열을 저장할 때 사용한다.

```
>>> string = """I am Gio Paik.
and I am 21 years old."""
>>> print(string)
I am Gio Paik.
and I am 21 years old.
```

문자열 자료형 – 이스케이프 코드

문자열을 저장할 때, 프로그래밍 문법 상 사용할 수 없는 문자나 특수한 문자들을 저장하기 위해 이스케이프 코드를 사용한다.

주로 사용되는 이스케이프 코드는 아래와 같다.

₩n	개행 문자, 줄바꿈을 할 때 사용
₩t	문자열 사이에 탭(tab) 간격을 적용
₩₩	₩ 문자를 그대로 출력
₩’	작은 따옴표
₩”	큰 따옴표

```
>>> string = '\\n을 사용하면 줄이 바뀐다.\n이렇게.'
>>> print(string)
\n을 사용하면 줄이 바뀐다.
이렇게.
```

문자열 자료형 – 문자열의 연산자

문자열에 +와 * 연산자를 사용할 수 있다.

+ 연산자를 활용하여 문자열을 합칠 수 있다.

```
>>> a = 'Lift is short, '
>>> b = "You need Python."
>>> a+b
'Lift is short, You need Python.'
```

* 연산자를 활용하여 문자열을 여러 번 반복시킬 수 있다.

```
>>> a = '='
>>> a * 10
'=========='
```

문자열 자료형 – 인덱싱과 슬라이싱

문자열은 기본적으로 문자의 시퀀스이다.

len() 함수를 통해 문자열의 길이를 알 수 있다.

```
s = 'Life is short, You need python.'
>>> len(s)
31
```

또한, 문자열 상에서의 위치를 통해 특정 순서의 문자를 추출할 수 있다.

```
>>> s[0]
'L'
```

콜론(:)을 활용해서 특정 구간의 문자열을 추출할 수도 있다.

```
>>> s[0:4]
'Life'
```

이때, 앞의 0을 생략하여도 맨 앞에서부터 추출한다. 콜론 뒤의 숫자(4)를 생략하면, 맨 뒤까지 추출한다.

문자열 자료형 실습

이제 문자열 문제를 풀어보자.

1) 사용자로부터 문자열을 두 개 입력받아 합쳐서 출력해보자.
2) “python”이라는 문자열에서 슬라이싱을 활용해 tho만 출력해보자.

Bool 자료형

bool 자료형은 가장 단순한 1bit 자료형으로, 참 혹은 거짓의 값을 갖는다. (원래 이름은 Boolean이다.)

참은 True, 거짓은 False로 표현되며, 이때 False는 0 혹은 빈 값(‘’, “”, [] 등..), True는 이외의 모든 값과 같다.

```
>>> t = 0
>>> bool(t)
False
>>> t = 1
>>> bool(t)
True
```

bool 자료형은 앞으로 배울 반복문이나 조건문 등, 다양한 분야에 활용된다.

어렵지 않으니 꼭 기억해두자!

리스트, 튜플 자료형

지금까지 하나의 데이터를 갖는 자료형을 배웠다.
(수학으로 치면 스칼라 값을 갖는 변수)

그러나 살다보면 데이터 여러 개를 저장해야 할 일이 생긴다.
(백지오가 좋아하는 영화의 리스트는? 태양계에 있는 행성들의 지름은?)

즉, 이제 스칼라 값이 아닌 고차원의 데이터를 저장해야 한다! 이를 위해 있는 것이 리스트와 튜플이다.

리스트 자료형

먼저, 리스트를 배워보자.

리스트(list)는 이름 그대로, 여러 값을 저장하는 자료형이다. 아래와 같이 선언하여 사용한다.

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

위는 보는 그대로, 정수(int) 변수 4개를 저장하는 리스트이다. 대괄호([])를 이용하여 선언하고, 위처럼 리스트 자체에 접근하거나, 인덱스를 활용하여 리스트 내부의 값에 접근할 수도 있다.

```
>>> myList[0]
1
>>> myList[2]
3
```

유의해야 할 점은, 프로그래밍에서 인덱싱은 0부터 시작한다는 것이다.
즉, 리스트의 첫번째 인자에 접근하려면 0, 두번째 인자에 접근하려면 1로 접근해야 한다.

리스트 자료형 – 값 수정

리스트를 선언한 이후, 리스트에 값을 추가하거나 삭제 또는 수정할 수 있다.

특정 인덱스의 값을 수정하기 위해선 아래와 같이 간단히 수정할 수 있다.

```
>>> myList[0] = 7
>>> myList
[7, 2, 3, 4]
```

리스트 뒤에 특정 값을 추가하려면 append 함수를 사용하면 된다.

```
>>> myList.append(8)
>>> myList
[7, 2, 3, 4, 8]
```

이 외에도 다양한 리스트 자료형 관련 함수가 있는데, 이는 다음 슬라이드의 표를 참조하자!

리스트 자료형 - 함수

함수	하는 일
append(x)	리스트의 맨 뒤에 x를 추가한다.
sort()	리스트를 정렬
reverse()	리스트를 역순으로 변환
index(x)	리스트 안에서 x값의 인덱스를 반환
insert(a,b)	리스트의 a번째 위치에 b값을 삽입
remove(x)	리스트에서 가장 앞에 있는 x값 삭제
pop()	리스트 맨 마지막 값을 반환하고 그 값은 리스트에서 삭제
count(x)	리스트에 담긴 x의 개수 새기
extend(list)	리스트에 뒤에 다른 리스트 list를 이어 붙임

리스트 자료형 - 특징

리스트는 다른 프로그래밍 언어들의 배열 등과 비슷하지만, 큰 차별점이 있다.
바로, 한 리스트에 여러가지 자료형의 값을 담을 수 있다는 것이다. 심지어 리스트를 담은 리스트도 가능하다!

```
>>> various = [123, 3.14, 'string', True, [1,2,'hi']]
>>> various
[123, 3.14, 'string', True, [1, 2, 'hi' ]]
```

위 various 리스트 안에는 정수, 실수, 문자열, bool, 리스트까지 우리가 배운 모든 자료형의 데이터가 들어있다!

리스트안에 리스트를 넣는 방법을 통해 우리는 고차원 데이터도 저장할 수 있다!

```
>>> matrix = [[1,2,3],
              [4,5,6]]
>>> matrix
[[1, 2, 3], [4, 5, 6]]
```

위는 리스트안의 리스트를 통해 행렬을 구현한 모습이다.

튜플 자료형

이번에는 튜플 자료형을 알아보자.

튜플 또한 리스트와 크게 다르지 않다.

여러 개의 다양한 자료형의 변수를 담을 수 있고, 인덱스를 통해 개별 값에 접근하거나 전체 값을 불러올 수도 있다.

그러나 가장 큰 차이점은, 튜플은 한 번 선언되면 내부 값의 수정이 불가능하다는 것이다!

```
>>> myTuple = (1,2,3)
>>> myTuple
(1, 2, 3)
```

튜플은 리스트와 달리 소괄호를 통해 만들고 구분한다.
그러나 인덱싱을 통해 내부 값에 접근할 때는 여전히 대괄호를 사용하므로 주의하자.

```
>>> myTuple[2]
3
```

딕셔너리 자료형

딕셔너리 자료형에 대해 알아보자.

딕셔너리 자료형은 dict로 표기하며 사전 자료형이라고도 부른다.

딕셔너리 자료형의 특징은 키값의 존재이다.

리스트와 튜플에서, 각 값은 값이 리스트 내에서 위치한 순서(인덱스)로 구분했다.

딕셔너리는 이와달리 키값을 이용하여 값을 구분한다.

딕셔너리 내부의 각 값은 키와 벨류(value)의 쌍인 아이템(item)으로 이루어진다. 선언 방식은 아래와 같다.
(보면 알겠지만 중괄호({})를 사용한다.)

```
>> myDict = {'name':'Gio','age':21,'isMale':True}
```

값들이 콜론(:)을 기준으로 쌍을 이루고 있다. 각각 왼쪽이 키, 오른쪽이 벨류 값이다.

딕셔너리 자료형

키와 벨류의 분리를 통해 개발자는 더욱 쉽게 원하는 값을 찾고 접근할 수 있다.

```
>>> myDict['name']
'Gio'
```

이는 다른 개발자와 협업하거나, 보기 좋은 코드를 만드는 클린코딩에 있어 크게 도움이 된다.

위에서 설명하지는 않았지만, 키값으로 str 외에도 다양한 자료형을 쓸 수 있으니 참고하자.

집합 자료형

딕셔너리를 제외하고 리스트와 튜플, 문자의 집합인 문자열(str) 자료형은 순서가 중요한 자료형으로, 시퀀스라고 부른다.

수학에서, 벡터 (2,3)과 (3,2)가 다름을 생각하면 될 것이다.

문자열 “abc”와 “bac”, “cab”는 모두 다른 문자열이다.

또한, “aa”와 “aaa”는 다르다.

그러나 중복과 순서가 중요하지 않은 자료형이 있다. 집합(set) 자료형이다.

집합 자료형은 리스트나 튜플에 set() 함수를 활용해서 선언한다.

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

중복으로 들어간 l이 없어지고, 각 알파벳이 들어간 집합이 생성되었다.

집합 자료형

집합 자료형은 간단하게 중복을 제거할 때 활용하거나, 수학적으로 집합을 사용할 필요가 있을 때 사용할 수 있다

아래처럼 연산자를 활용하여 집합 연산을 할 수 있다.

```
>>> s1 = set([2,4,6,8])
>>> s2 = set([3,6,9,12])
>>> s1 & s2
{6}
```

연산자	용도
&	교집합
|	합집합
-	차집합

강의자 정보

백지오
컴퓨터공학과 19학번, 인터페이스 32기 학술차장
https://skyil.tistory.com/
https://github.com/skyil7
https://linkedin.com/in/giopaik

수고하셨습니다 :)

자료형을 배우느라 수고하셨습니다! 자료형은 프로그래밍 언어에서 가장 어려운 부분이기도 하지만,
자료형을 안다는 것은 그 언어를 안다고 할 수 있을 정도로 중요하고 핵심적인 부분이기도 합니다!

실제로 프로그래밍 경험자가 다른 프로그래밍 언어를 배울 때 가장 많은 시간을 투자하게 되는 부분이 보통 자료형입니다.

같은 자료형(문자열이나 정수)을 다루더라도 언어의 설계이념이나 작동 방식 등에 따라 자료형의 구현 방식에 차이가 있기 때문입니다.

실제로 파이썬은 모든 자료형을 class 라는 객체 형태로 다루며, 사용자가 매직 메소드라는 도구를 활용하여 손쉽게 새로운 자료형을 구현할 수도 있게 되어있습니다.
(이는 추후에 기회가 된다면 파이썬 심화과정에서 다루도록 하겠습니다!)

자료형을 배우는 것이 비록 고됬을 수도 있지만, 우리가 살고 있는 세상을 이진수로 구성된 컴퓨터에게 이해 가능하게 하는 핵심적인 부분이니 만큼 잘 짚고 넘어가셨으면 좋겠습니다.
(우리가 사는 세상을 숫자만으로 표현할 수 있습니다. 이러한 패러다임은 멀티미디어와 인공지능에 큰 영향을 주고 있습니다.)

다시 한번 어려운 자료형들을 배우느라 수고하셨고 앞으로도 잘 부탁드립니다!
- 강의자 백지오