Python

변수와 연산자

Spring 2025

AI융합학과
Seongbok Baik
sbbaik@dju.kr

00 Text Book

교재명	으뜸 파이썬
저자	박동규, 강영민
출판사	생능출판사
발행년	2024.06.14

학습목표

- print() 함수의 사용법을 자세히 알아본다.
- 변수의 개념과 사용법에 대해 이해한다.
- 변수의 자료형에 대해 이해한다.
- 식별자란 무엇인지 알아보고, 식별자 이름을 짓는 규칙에 대해 알아본다.
- 키워드란 무엇인가 알아본다.
- 연산자의 개념과 종류에 대해 알아본다.
- 문자열을 사용할 수 있다.
- 주석문의 개념과 사용법에 대해 알아본다.

2.1 파이썬의 출력 함수 print()

- 1장의 내용을 복습해 보자.
- print() 함수를 통해서 파이썬 코드가 수행한 내용을 화면에 출력해 볼 수 있다.
- 대화식 실행 모드와 스크립트 파일 실행 모드가 있다.

2.1 파이썬의 출력 함수 print()

- 대화식 실행모드
  - 파이썬 명령어를 대화창에 입력하면 바로 인터프리터의 반응(피드백)을 받을 수 있다.
  - 간단한 코드를 테스트할 적에는 주로 대화식 실행모드를 사용

```
>>> print('Hello Python!!')
Hello Python!!
```

- 스크립트 실행모드
  - .py라는 확장자를 가지는 스크립트를 만들어서 파이썬 인터프리터에 한꺼번에 전달하여 실행시키는 모드이다.
  - 복잡한 로직이 있는 코드는 스크립트 파일을 만들어서 실행

2.1 파이썬의 출력 함수 print()

- 다음 코드를 입력해 봅시다.

```
>>> print('Hello Python!!')
    ...
    SyntaxError: invalid syntax
```

오류 발생
SyntaxError : 구문 오류
invalid syntax : 유효하지 않은 구문
번역기는 여러분에게 친절하게 오류를 표시해 주고 알려줍니다.

```
>>> print('My age is', 20)  # 문자열과 숫자를 쉼표로 구분하여 출력
    My age is 20
>>> print('오늘의 걸음 수', 8000, '걸음')
    오늘의 걸음 수 8000 걸음
```

제대로 된 출력 방식
문자열과 숫자는 쉼표로 구분해 줍시다.

```
>>> print('Hello ' * 2)   # 문자열 'Hello '가 두 번 반복 출력됨
    Hello Hello
>>> print('Hello ' * 4)   # 문자열 'Hello '가 네 번 반복 출력됨
    Hello Hello Hello Hello
```

문자열에 * 연산을 하고 숫자를 넣어 줄 경우 :
숫자만큼 문자열을 반복 출력한다.

2.1 파이썬의 출력 함수 print()

- 주석문

- # 문자열과 숫자를 쉼표로 구분하여 출력으로 나타난 부분은 주석문으로 소스 코드에 붙이는 설명글과 같다. 이 부분은 프로그램이 하는 일을 설명하는 목적으로 주로 사용된다.
- 이 주석문은 # 기호로 시작하는데, 이 기호가 나타난 부분부터 줄의 끝까지 파이썬 인터프리터가 해석하지 않기 때문에 실행에 전혀 영향을 주지 않는다.

```
>>> print('My age is', 20)  # 문자열과 숫자를 쉼표로 구분하여 출력
My age is 20
>>> print('오늘의 걸음 수', 8000, '걸음')
오늘의 걸음 수 8000 걸음
```

코드 실행에 영향을 주지 않음

2.1 파이썬의 출력 함수

- 대화식 실행모드 제공

[표 2-1] 파이썬의 대화식 실행 모드와 스크립트 실행 모드에서 이름 출력하기

	대화식 실행 모드	스크립트 실행 모드

입력	>>> print('당신의 이름은 :')
	당신의 이름은 :
	>>> name = '홍길동'
	>>> print(name)
	홍길동
		print('당신의 이름은 :')
		name = '홍길동'
		print(name)
		
		(파일명: print_name.py)

수행	입력 후 엔터 키를 입력하면 실행	$ python print_name.py 당신의 이름은 : 홍길동
비고	코드를 입력하고 엔터 키를 입력하면 실행 결과가 코드 아래에 나타난다.	코드를 저장하고 명령줄에서 파이썬 번역기를 호출하여 실행하거나 IDLE에서 F5 단축키로 실행시킬 수 있다.

2.1 파이썬의 출력 함수

LAB 2-1: 대화식 모드에서 출력하기

1. 다음 코드를 입력해 보고 그 출력 결과를 빈칸에 적으시오.

```
>>> print('나의 이름은 :', '홍길동')
    _____________________________________

>>> print('나의 나이는 :', 27)
    _____________________________________

>>> print('나의 키는', 179, 'cm 입니다.')
    _____________________________________

>>> print('10 + 20 =', 10 + 20, '10 * 20 =', 10 * 20)
    _____________________________________
```

직접 입력해 보고 그 결과를 확인해 보세요.

2.1 파이썬의 출력 함수

- 스크립트를 하나의 파일에 작성 후 일괄적으로 실행

코드 2-1: 스크립트 코드로 간단한 출력 프로그램 작성하기

print_test.py

파일을 만들어 보세요.

```
print('나의 이름은 :', '홍길동')
print('나의 나이는 :', 27)
print('나의 키는', 179, 'cm 입니다.')
print('10 + 20 =', 10 + 20)
```

위의 파일을 실행시켜 보세요.

```
나의 이름은 : 홍길동
나의 나이는 : 27
나의 키는 179 cm 입니다.
10 + 20 = 30
```

2.2 변수와 친해지기

코드 2-2: 원의 반지름, 면적, 둘레를 출력하는 프로그램

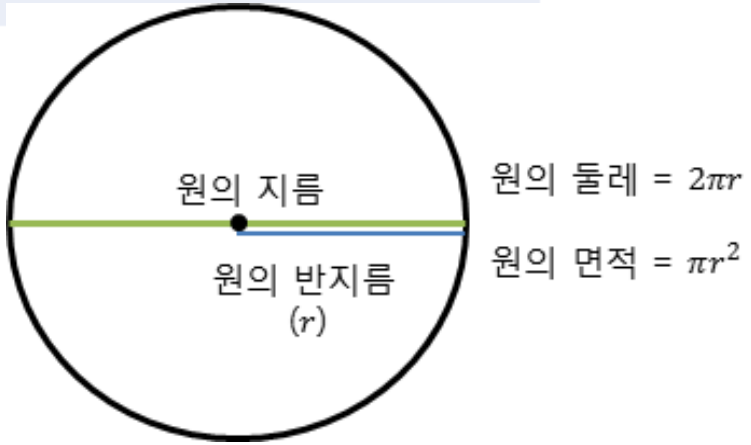
circle.py

```
print('원의 반지름', 4.0)
print('원의 면적', 3.14 * 4.0 * 4.0)
print('원의 둘레', 2.0 * 3.14 * 4.0)
```

반지름이 4.0인 원을 가정해 보고 원의 면적과 둘레를 구해봅시다.

실행결과

```
원의 반지름 4.0
원의 면적 50.24
원의 둘레 25.12
```



2.2 변수와 친해지기

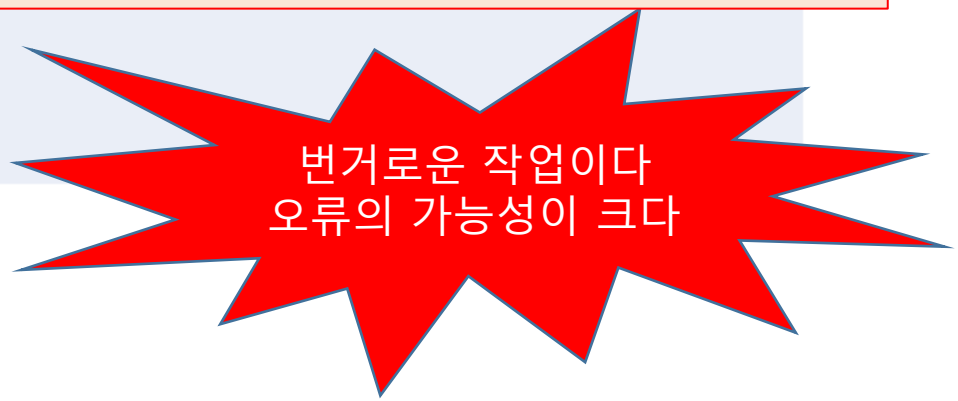
- 방금 작성한 프로그램에서 반지름이 5.0, 6.0인 원의 면적과 둘레를 새로 구하는 경우 다음과 같이 코드를 수정해야 함

```
print('원의 반지름', 5.0)
print('원의 면적', 3.14 * 5.0 * 5.0)
print('원의 둘레', 2.0 * 3.14 * 5.0)
```

반지름이 5.0인 원을 가정해 보고 원의 면적과 둘레를 구해봅시다.

```
print('원의 반지름', 6.0)
print('원의 면적', 3.14 * 6.0 * 6.0)
print('원의 둘레', 2.0 * 3.14 * 6.0)
```

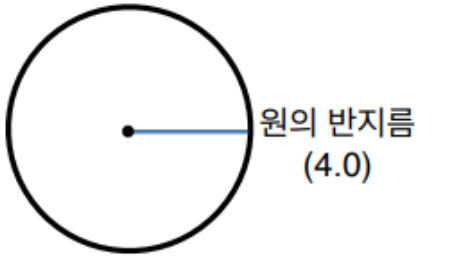
반지름이 6.0인 원을 가정해 보고 원의 면적과 둘레를 구해봅시다.



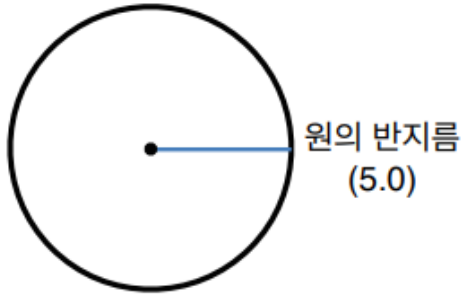
2.2 변수와 친해지기

- 변수 도입의 필요성

```
print('원의 반지름', 4.0)
print('원의 면적', 3.14 * 4.0 * 4.0)
print('원의 둘레', 2.0 * 3.14 * 4.0)
```




```
print('원의 반지름', 5.0)
print('원의 면적', 3.14 * 5.0 * 5.0)
print('원의 둘레', 2.0 * 3.14 * 5.0)
```



[그림 2-2] 원의 반지름이 4.0에서 5.0으로 변경되면 [코드 2-2]를 다시 수정해야 한다

2.2 변수와 친해지기

- **변수**variable의 도입

  - 프로그램을 작성하다 보면 상황에 따라 값이 변하는 것들이 있다. 이들이 여러 곳일 때 하나라도 실수를 하게 되면 프로그램의 오류가 나타나거나 의도한 바와 다른 결과를 얻게 됨
  - **변수**를 도입해서 번거로운 일을 간단하게 만들 수 있음
  - 변수에 값을 저장하고 이후에는 값이 아니라 변수를 사용해 보자.
    - radius = 4.0 와 같이 radius라는 이름을 가지는 변수를 만든다.
    - 이 변수 radius는 4.0이라는 값을 저장해 두고 있다.
    - 나중에 이 변수를 꺼내어 사용할 수 있다.

2.2 변수와 친해지기

- **변수variable의 도입**

  - 변수variable를 도입

코드 2-3: 변수를 이용하여 원의 면적과 둘레를 구하는 방법

circle_with_var.py

```
radius = 4.0
print('원의 반지름', radius)
print('원의 면적', 3.14 * radius * radius)
print('원의 둘레', 2.0 * 3.14 * radius)
```

radius라는 변수를 도입해 봅시다.

실행결과

```
원의 반지름 4.0
원의 면적 50.24
원의 둘레 25.12
```

오류의 가능성이 줄어든다.
수정이 용이하다.

2.2 변수와 친해지기

- **변수variable의 도입**

  - 변수variable를 도입

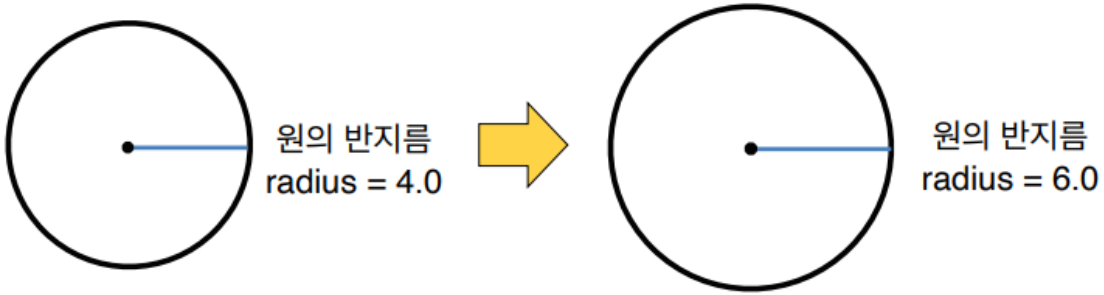
코드 2-4: 변수를 이용하여 원의 면적과 둘레를 구하는 방법
circle_with_var(수정).py

```
radius = 6.0
print('원의 반지름', radius)
print('원의 면적', 3.14 * radius * radius)
print('원의 둘레', 2.0 * 3.14 * radius)
```

변수에 값을 저장하고 이를 불러서 사용하면
프로그램의 수정이 쉬워지고 오류를 줄일 수 있다.

실행결과

```
원의 반지름 6.0
원의 면적 113.03999999999999
원의 둘레 37.68
```



[그림 2-3] 원의 반지름이 4.0에서 6.0으로 변경되면 변수 radius만 수정하면 된다

2.2 변수와 친해지기

LAB 2-3: 변수를 사용하는 프로그램

1. **[코드 2-4]**를 수정하여 다음과 같이 원의 반지름이 `8.0`일 때 원의 둘레와 면적을 출력하여라(변수값 `radius`를 `8.0`으로 변경하는 방식을 사용하여라).

```
원의 반지름 8.0
원의 면적 200.96
원의 둘레 50.24
```

2. **[코드 2-4]**를 수정하여 원의 반지름이 `10.0`일 때 원의 둘레와 면적을 출력하여라.

3. 사각형의 면적을 구하기 위하여 `width`(너비)와 `height`(높이)라는 변수를 이용하도록 하자. 이 값이 각각 `100`, `200`이 되도록 변수를 생성한 후 두 변수를 곱하여 다음과 같은 결과가 나오도록 `rectangle_test.py` 스크립트를 작성하여라.

```
너비 100, 높이 200인 사각형의 면적 : 20000
```

2.2 변수와 친해지기

- **변수**variable
  - 변할 수 있는 수라는 의미
  - 데이터를 저장하는 저장 공간을 가리키는 이름
  - 변수(變:변할 변, 數:셀 수)라는 명칭을 사용, '수'는 단순한 수치라기 보다는 **데이터**로 이해하는 것이 더 정확하다.
  - 이름을 통해 자유롭게 데이터에 대한 읽기, 쓰기, 수정하기가 가능
- **식별자**identifier
  - 사용자가 정의하는 변수나 함수에 대해 서로 구별되는 이름을 부여해야 함
  - 이와 같이 서로 구별되는 이름을 식별자라고 한다
  - 하나의 변수 이름을 여러 개의 메모리 위치를 지칭하는데 사용하게 되면 어느 메모리 공간을 지칭하는지 알기 어려움
  - 다른 메모리 위치에는 서로 다른 이름을 부여해야 함

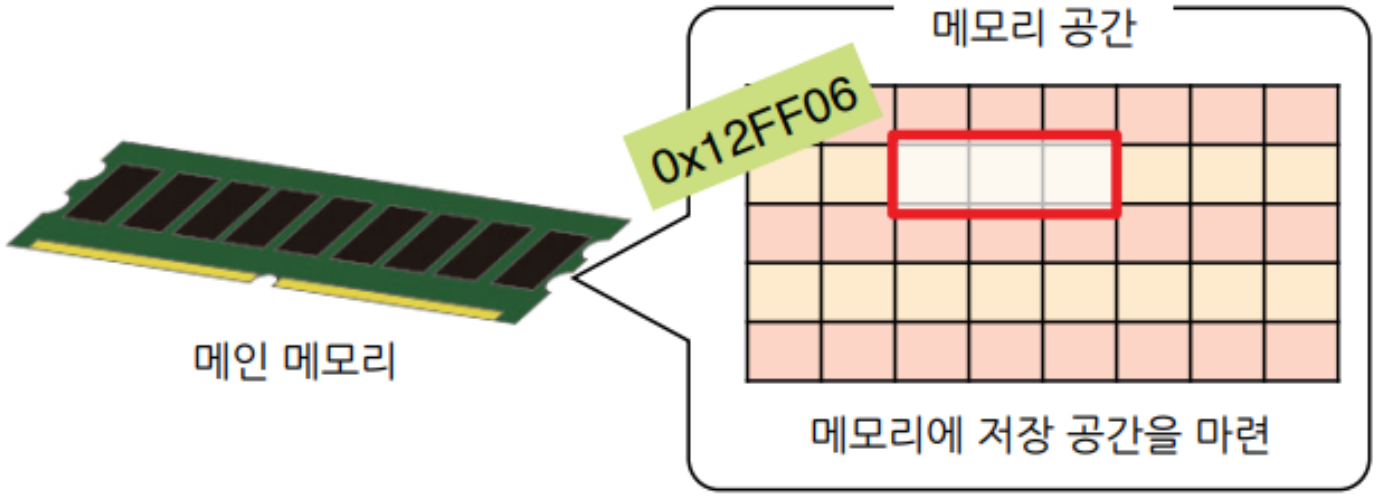
2.2 변수와 친해지기

- **메인 메모리**main memory
  - 컴퓨터의 데이터가 저장되어 읽기와 쓰기, 덮어쓰기를 하는 곳
  - 메모리라고도 불림

- **메모리 주소**memory address
  - 메모리에 데이터를 저장한 곳의 위치
  - 저장된 데이터를 읽고 쓰기 위해서는 데이터가 저장된 곳(공간 또는 위치)이 어디인가를 알아야 한다.
  - 주소는 보통 16진수로 표현

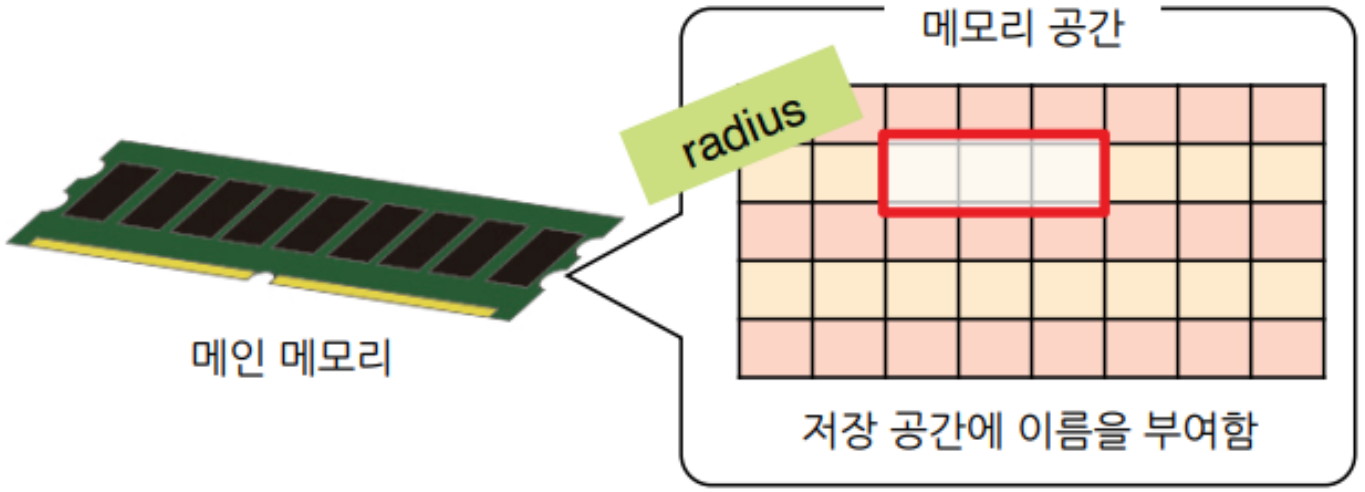
2.2 변수와 친해지기

- 컴퓨터의 메인 메모리와 메모리 주소



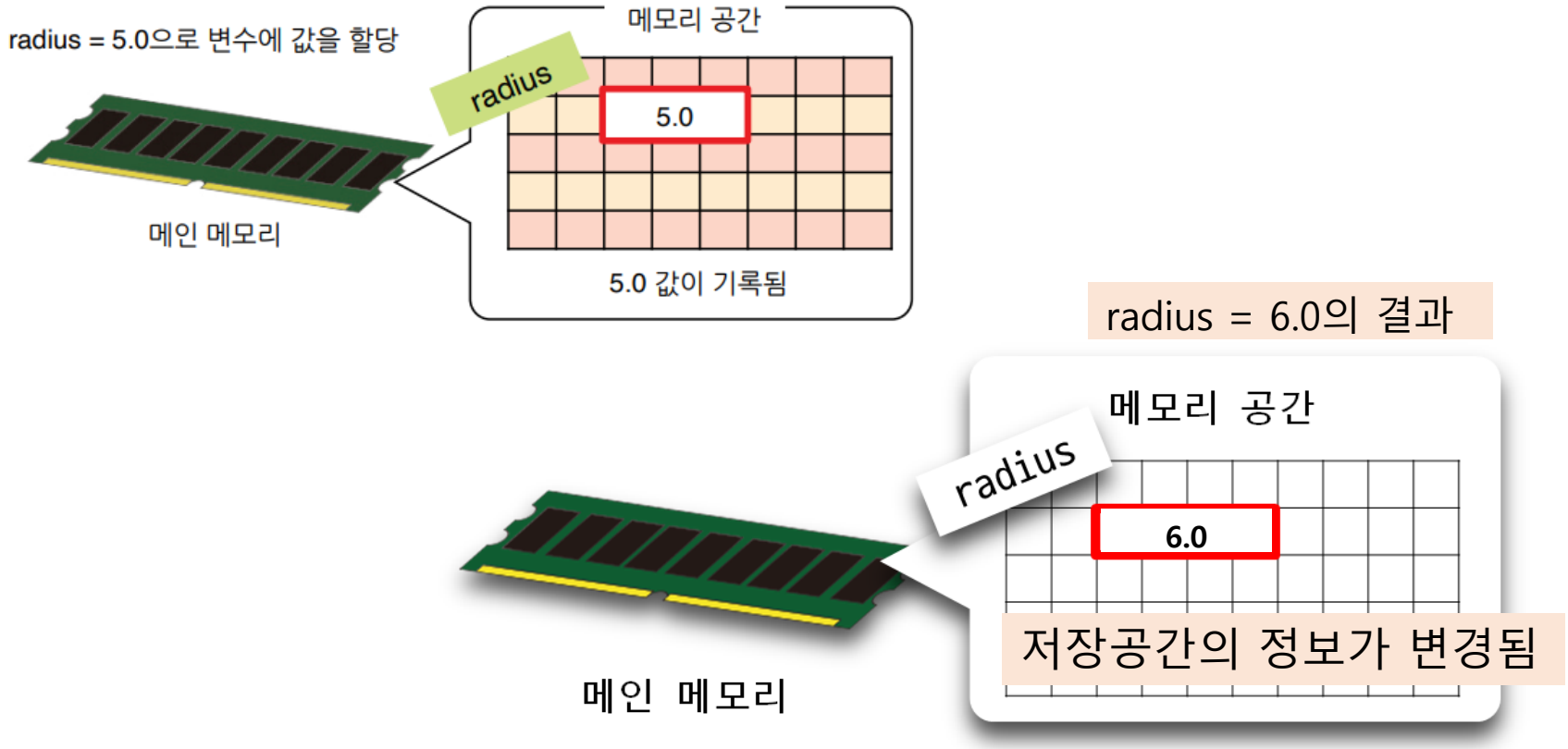
2.2 변수와 친해지기

- 식별을 위한 이름 radius를 부여



2.2 변수와 친해지기

- 변수와 메모리 공간



2.2 변수와 친해지기

잠깐 - 비트와 바이트 그리고 데이터 용량의 표기법

컴퓨터는 데이터를 저장할 때 0과 1의 정보만을 사용하는 2진수(**BI**nary digi**T**)를 사용한다. 이와 같이 컴퓨터에서 사용하는 정보 표현의 최소의 단위를 **비트**bit라고 한다. 그러나 한 비트만으로는 표현 가능한 정보가 너무 적기 때문에 주로 8비트 단위로 저장하는데 이 8비트 단위를 **바이트**byte라는 용어로 표현한다. 1바이트는 256개의 서로 다른 상태 정보를 표현할 수 있다.
컴퓨터와 같은 정보 기기에서 정보의 표현에 사용되는 비트와 바이트 그리고 그 이상의 데이터 용량은 다음과 같은 이름과 기호로 표기한다.

이름	기호	설명
1비트	bit	0 또는 1의 정보를 나타냄
1바이트	byte	8bit(8개의 비트를 조합하여 정보를 표현)
1킬로바이트	KB	1,024byte(=2^{10}byte)
1메가바이트	MB	1,024KB(=2^{10} KB = 2^{20}byte)
1기가바이트	GB	1,024MB(=2^{10} MB = 2^{30}byte)
1테라바이트	TB	1,024GB(=2^{10} GB = 2^{40}byte)
1페타바이트	PB	1,024TB(=2^{10} TB = 2^{50}byte)
1엑사바이트	EB	1,024PB(=2^{10} PB = 2^{60}byte)
1제타바이트	ZB	1,024EB(=2^{10} EB = 2^{70}byte)

2.3 변수의 생성과 식별자

- 리터럴literal
  - 프로그래밍 언어에서 고정된 데이터 값을 나타냄

대화창 실습: 여러 가지 변수의 선언과 출력

```
>>> name = '홍길동'      # 문자열 리터럴 '홍길동'과 이를 참조하는 변수 name
>>> print('이름 :', name)
    이름 : 홍길동
>>> width = 10          # 정수형 리터럴 10과 이를 참조하는 변수 width
>>> height = 5
>>> rectangle_area = width * height
>>> print('사각형의 면적 :', rectangle_area)
    사각형의 면적 : 50
```

[그림 2-7] = 연산자를 이용하여 name 변수에 문자열 값을 할당하는 방법

2.3 변수의 생성과 식별자

LAB 2-4: 다음 코드를 입력해 보고 그 출력 결과를 밑줄 부분에 적으시오.

```
>>> name = '전우치'
>>> print('나의 이름은 :', name)
_____________________

>>> age = 27
>>> print('나의 나이는 :', age)
_____________________

>>> height = 179
>>> print('나의 키는', height, 'cm 입니다.')
_____________________

>>> sum = 10 + 20
>>> print('10 + 20 =', sum)
_____________________

>>> mult = 10 * 20
>>> print('10 * 20 =', mult)
_____________________
```

2.3 변수의 생성과 식별자

- 식별자identifier

  - 다른 함수와 구별되는 고유의 이름
  - 프로그램이 단순할 경우 a, b, n, m과 같은 단순한 이름의 식별자로도 그 기능을 구현할 수 있다.
  - 프로그램이 복잡해지면 walk_distance, num_of_hits, english_dict, student_name과 같이 그 의미를 이해하기 쉬운 변수를 사용하는 것이 편리하다.
  - C나 자바 언어와 마찬가지로 파이썬은 대소문자를 구분하기 때문에 'index'와 'Index', 'INDEX'는 다른 식별자이다.

2.3 변수의 생성과 식별자

- 식별자identifier

식별자 이름 규칙

1. 식별자의 이름은 문자와 숫자, 밑줄 문자 _로 이루어진다.
2. 중간에 공백이 들어가면 안 된다.
3. 첫 글자는 반드시 문자나 밑줄 문자 _로 시작해야만 한다.
4. 대문자와 소문자는 구분된다. 따라서 Count와 count는 서로 다른 식별자이다.
5. 식별자의 길이에 제한은 없다.
6. 키워드는 식별자로 사용할 수 없다.

2.3 변수의 생성과 식별자

- 식별자identifier

[표 2-2] 파이썬에서 사용 가능한 식별자들

사용 가능한 식별자	특징
number4	영문자로 시작하고 난 뒤에는 숫자를 사용할 수 있음
__code__	밑줄 문자는 일반 문자와 같이 식별자 어디든 나타날 수 있음
my_list	
for_loop	키워드라 할지라도 다른 문자와 연결해 쓰면 문제가 없음
높이	유니코드 문자인 한글 문자도 변수로 사용 가능

[표 2-3] 파이썬에서 사용 불가능한 식별자들

사용 불가능한 식별자	사용할 수 없는 이유
1st_variable	숫자 1로 시작하는 식별자임
my list	공백이 들어간 식별자임
global	global은 파이썬의 키워드임
ver2.9	특수 기호가 사용되었음(.)
num&co	특수 기호가 사용되었음(&)

2.3 변수의 생성과 식별자

- 식별자identifier

  - 키워드keyword 혹은 예약어reserved word
  - 미리 지정된 역할을 수행하도록 예약된 단어
  - import, for, if, def, class... 등과 같은 단어가 이에 해당

[표 2-4] 파이썬 키워드 목록: 파이썬 키워드는 사용 용도가 정해져 있어서 변수로 사용할 수 없다

파이썬의 키워드				
False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

2.3 변수의 생성과 식별자

- 식별자identifier

> **잠깐 - 파이썬의 식별자 이름 짓기 가이드라인**
>
> - 식별자의 이름은 a, b 혹은 A, B와 같은 간단한 이름을 사용할 수 있다. 하지만 I(i의 대문자), l(L의 소문자)과 같이 잘 구분되지 않는 글자는 사용하지 않도록 한다. 그리고 대문자 O와 숫자 0도 잘 구분되지 않으니 주의하자.
> - lowercase 혹은 lowercase_with_underscore와 같이 소문자나 밑줄 문자로 연결된 소문자 식별자도 문제없이 사용 가능하며, 대문자와 밑줄로 지어진 UPPERCASE나 UPPER_CASE_WITH_UNDERSCORE와 같은 이름도 가능하다.
> - 파이썬의 변수와 함수 이름은 my_variable과 같이 밑줄 문자를 이어서 사용하는 것이 좋다. CapitalizedWords라는 이름이나 mixedCase와 같이 단어의 시작을 대문자로 연결한 식별자도 가능하다. 이와 같은 표기법을 **캡워드**capword 표기법 혹은 **낙타등**camel case 표기법이라고 한다(낙타의 등과 같이 올라가고 내려간 부분이 있다는 것에서 유래함).
> - 뒤에 배울 내용이지만 __(두 개의 언더스코어) 문자로 시작하는 식별자는 특별한 클래스의 속성을 지칭하거나 특수 메소드의 이름으로만 사용하고 일반 함수나 변수의 이름으로는 사용하지 않는다.

2.3 변수의 생성과 식별자

- 식별자identifier

  - "CapitalizedWords" "mixedCase" 등과 같은 것을 **캡워드**capword 표기법 혹은 **낙타등** camel case 표기법이라고 함
    - 좋은 변수 이름을 선택해야 코드를 쉽게 이해할 수 있음

주의 - 변수 이름과 내장함수 이름

파이썬에서 sum, max, min, len, list 등은 변수 이름으로 사용할 수 있다. 하지만 이렇게 사용된 변수 이름은 sum(), max(), min(), len(), list() 등과 같은 파이썬에서 제공하는 내장함수의 이름과 중복되므로 사용하지 않도록 한다. 즉, 다음과 같이 sum이라는 변수를 사용한 후 sum() 내장함수를 호출하면 내장함수의 호출이 일어나지 않고 오류가 발생한다.

```
>>> sum = 100          # sum()이라는 내장함수 명과 같은 변수 이름 sum
>>> lst = [10, 20, 30]
>>> total = sum(lst)    # sum()이라는 내장함수 호출 시 오류 발생
    ...
    TypeError: 'int' object is not callable
```

2.3 변수의 생성과 식별자

- 식별자identifier

LAB 2-5: 다양한 식별자를 활용한 변수 사용

1. 다음 코드를 작성하고 실행해 보자. 이 코드는 어떤 에러를 유발할까? 또 이 에러를 어떻게 수정해야 할까?

```
global = 300
print(global)
```

2. 다음 코드는 어디에서 에러가 발생할까?

```
width = 20
height = 40
area = ( width * Height )
print('사각형의 면적', area)
```

이 코드의 에러를 수정하여 너비가 20, 높이가 40인 사각형의 면적을 다음과 같이 출력해 보자.

```
사각형의 면적 : 800
```

3. 다음 코드는 어떤 문제가 있을까?

```
iixxjjkk = 20
print('나의 나이는', iixxjjkk, '세 입니다')
```

1) 이 코드는 에러를 출력하는가?(예, 아니오로 대답하여라)
2) 이 코드는 어떤 측면에서 문제점이 있는가?

2.3 변수의 생성과 식별자

- 식별자identifier

코드 2-5: 변수에 값을 지정하고 출력하기

variable_test.py

```
name = '홍길동'      # '홍길동'을 참조하는 변수 name의 자료형은 문자열임
age = 27            # 27을 참조하는 변수 age의 자료형은 정수형임
print('안녕! 나는', name , '이야. 나는 나이가', age, '살이야.')
```

문자열 '홍길동'을 저장하는 변수 name

정수 값 27을 저장하는 변수 age

실행결과

```
안녕! 나는 홍길동 이야. 나는 나이가 27 살이야.
```

각 변수는 그 값을 표현하는 데에 적합한 크기의 공간을 메모리에서 확보해야 한다. 이렇게 필요한 공간을 결정하고, 그 공간에 값을 쓰고, 읽기 위해서는 자료의 형type을 결정해야 하는데, 사용자가 지정하지 않아도 파이썬이 가장 적합한 형을 결정한다

2.3 변수의 생성과 식별자

- 식별자identifier

코드 2-6: 변수에 새로운 값을 할당하기

change_var.py

```
name = '홍길동'
age = 27
print('안녕! 나는', name , '이야. 나는 나이가', age, '살이야.')
name = '홍길순'
age = 23
print('안녕! 나는', name , '이야. 나는 나이가', age, '살이야.')
```

실행결과

```
안녕! 나는 홍길동 이야. 나는 나이가 27 살이야.
안녕! 나는 홍길순 이야. 나는 나이가 23 살이야.
```

2.3 변수의 생성과 식별자

- 식별자identifier

LAB 2-6: 다양한 식별자를 활용한 변수 사용

다음과 같은 코드는 최종적으로 어떤 결과를 출력할까? 출력 결과를 미리 예상해 보고 코딩해 보자.

```
width = 20
height = 40
width = 30
area = width * height
print('사각형의 면적', area)
```

width 변수에 새로운 값 30을 할당함

width 변수에 새로운 값 30이 사용됨

2.4 변수와 연산자

- 컴퓨터의 자료 값은 덧셈, 뺄셈, 곱셈, 나눗셈들과 같은 **산술 연산**mathematical operation이 가능
- 파이썬은 이러한 산술 연산을 위한 여러 연산자를 제공
- "27이라는 값을 age라는 변수에 할당하여라."라는 명령어와 변수의 할당 과정을 보여주고 있음

```
age = 27
```

객체는 프로그램상의 어떤 자료로 데이터와 함수를 가질 수 있는 것으로 추후 상세히 설명함

1. 27이라는 값을 가지는 정수 객체가 생성된다.
2. age라는 변수가 27이라는 값을 가지는 정수 객체를 참조한다.

2.4 변수와 연산자

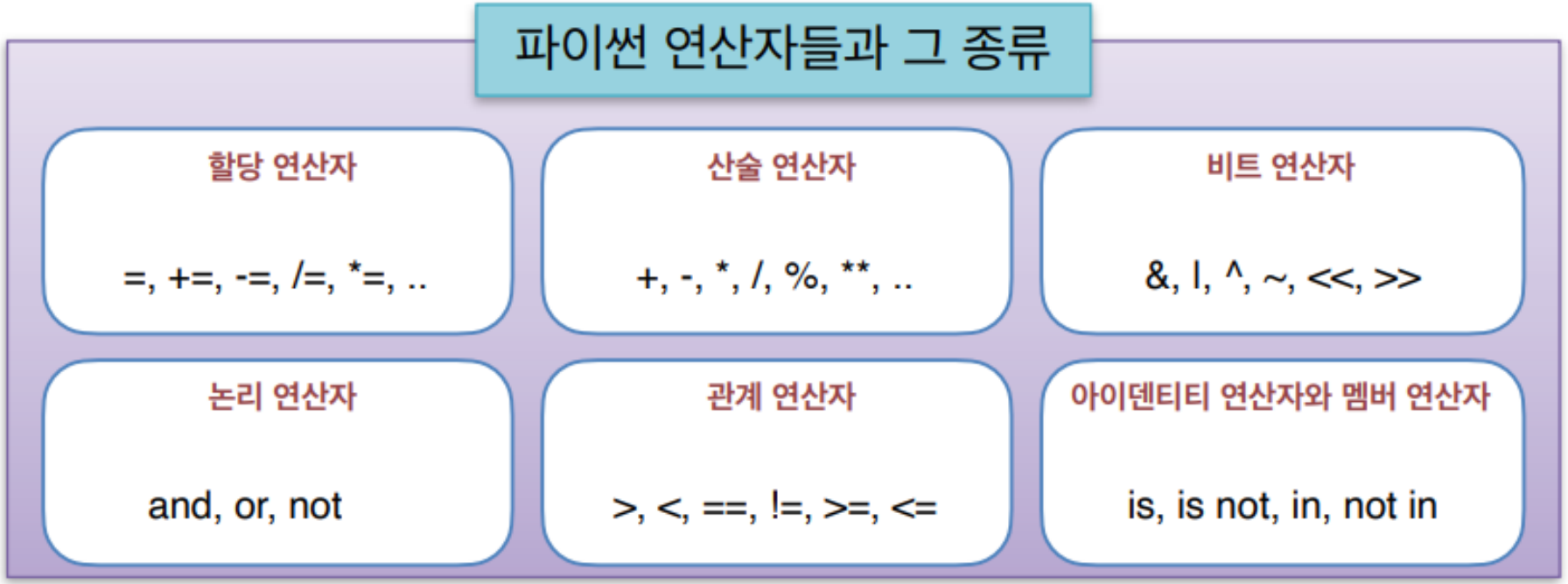
- 위의 과정을 통해 정수 27을 age라는 변수명이 참조함
- 할당 연산자 =의 왼쪽에는 변수 이름이 오고, 오른쪽에는 값으로 계산될 수 있는 상수, 변수 혹은 수식이 올 수 있음 (반대는 성립하지 않음)
- 어떤 변수에 처음으로 할당 연산자가 적용될 때 "변수 age가 **선언**declare되었다"라고 함
- = 기호를 **할당 연산자**assignment operator라고 한다.
- 아래 문장은 그므으로syntax error가 발생함

```
>>> 27 = age

File "", line 1
SyntexError: can't assign to literal
```

2.4 변수와 연산자

- 파이썬은 숫자 값에 대해 같이 기본적으로 **사칙연산**과 **나머지연산**, **제곱연산**을 수행하는 연산자를 제공



2.4 변수와 연산자

- 파이썬 연산자와 그 의미

[표 2-5] 산술 연산자들과 그 동작

연산자	의미	동작
+	덧셈	왼쪽 피연산자와 오른쪽 피연산자를 더한다.
-	뺄셈	왼쪽 피연산자에서 오른쪽 피연산자를 뺀다.
*	곱셈	왼쪽 피연산자와 오른쪽 피연산자를 곱한다.
/	실수 나눗셈	왼쪽 피연산자를 오른쪽 피연산자로 나눈다. 파이썬의 나눗셈은 기본적으로 실수값을 반환한다.
//	정수 나눗셈(몫)	/와 달리 나눗셈의 결과를 소수점 이하를 버리고 정수 부분만을 얻고자 할 경우에 사용한다.
%	나머지	**모듈로 연산자**라고 읽으며, 비율을 의미하는 퍼센트와는 상관이 없다. 나눗셈의 나머지를 구한다.
**	거듭 제곱	왼쪽 피연산자를 오른쪽 피연산자로 거듭제곱한다.

2.4 변수와 연산자

- 기본 연산자들을 사용해 파이썬을 계산기로 사용

2 + 4와 같은 문장을 **표현식**expression이라고 한다

대화창 실습: 파이썬 표현식의 사용

```
>>> 2 + 4       # 덧셈 연산
6
>>> 4.0 - 0.1   # 뺄셈 연산
3.9
>>> 20 * 30     # 곱셈 연산
600
>>> 11 / 2      # 실수 나눗셈 연산
5.5
>>> 11 // 2     # 정수 나눗셈 연산으로 11을 2로 나눈 몫을 구함
5
>>> 11 % 2      # 정수 나눗셈 후의 나머지 연산
1
>>> 4 ** 2      # 거듭제곱 - 4의 2승 연산으로 4 * 4를 구함
16
>>> 4 ** 0.5    # 거듭제곱 - 4의 0.5승으로 4의 제곱근을 구함
2.0
```

2.4 변수와 연산자

LAB 2-7: 파이썬 연산자의 사용

1. 다음과 같은 계산을 파이썬 대화창에서 수행하고 그 결과를 적으시오.

   1) `123 * 456`
   2) `1357 + 2468`
   3) `5 ** 4`
   4) `20 / 4`
   5) `20 // 5`
   6) `20 % 5`

2. 5를 2로 나눈 나머지를 구하시오. 이를 구하기 위한 파이썬 수식을 적으시오.

3. 2의 제곱근 $\sqrt{2}$ 와 3의 제곱근 $\sqrt{3}$ 을 ** 연산자를 사용하여 각각 구하시오.

2.4 변수와 연산자

코드 2-7: 문자열과 정수의 덧셈 연산
number_and_string1.py

```
my_age = 22                    # 정수 자료형
my_height = '177'              # 문자열 자료형

my_age = my_age + 1            # 정수 자료형끼리의 덧셈은 가능!
my_height = my_height + 1      # 문자열 자료형과 정수 자료형의 덧셈은 불가능!!

print(my_age, my_height)
```

실행결과

```
Traceback (most recent call last):
....
my_height = my_height + 1
TypeError: must be str, not int
```

- 변수 my_height와 같은 문자열 자료형 변수에 정수인 숫자 1을 더하는 연산이 불가능
- 따라서 위의 코드에서는 TypeError라는 오류가 발생
- 연산자는 특정한 자료형에서만 사용이 가능하다

2.4 변수와 연산자

코드 2-8: 실수와 정수의 덧셈 연산

number_and_string2.py

```
my_age = 22                         # 정수 자료형
my_height = 177.5                   # 실수 자료형

my_age = my_age + 1                 # 정수 자료형끼리의 덧셈

my_height = my_height + 1           # 실수와 정수 자료형끼리의 덧셈

print('my_age =', my_age, ', my_height =', my_height)
```

실행결과

```
my_age = 23 , my_height = 178.5
```

2.4 변수와 연산자

- 정수나 실수 사이에는 덧셈, 뺄셈, 곱셈, 나눗셈의 사칙연산이 잘 적용됨
- 정수에 대해서는 정수 나눗셈과 나머지 연산을 수행 가능
- ** 연산을 사용하여 거듭제곱 연산을 정수와 실수에 대해서도 적용 가능

대화창 실습: 거듭제곱 연산의 적용

```
>>> 0.2 ** 4
    0.0016000000000000003
>>> 4 ** 0.2     # 거듭제곱 연산에서 지수로 실수를 사용할 수 있다.
    1.3195079107728942
```

2.5 자료형의 의미와 자료형 확인

- 자료형data type
- 프로그래밍 언어에서 처리할 수 있는 데이터의 유형
  - 기본 자료형, 부울형, 숫자형(정수, 실수, 복소수), 문자열, 리스트, 튜플, 집합, 딕셔너리
  - 개체가 어떤 자료형인지를 알려주는 type()이라는 함수를 제공

대화창 실습: 다양한 자료형의 이해와 type() 함수

```
>>> num = 100
>>> type(num)          # num 변수의 자료형을 알려주는 함수
    <class 'int'>
>>> pi = 3.141592
>>> type(pi)           # pi 변수의 자료형을 알려주는 함수
    <class 'float'>
>>> message = "I love Python"
>>> type(message)      # message 변수의 자료형은 str 형임
    <class 'str'>
```

2.5 자료형의 의미와 자료형 확인

- 변수 num에는 100이라는 정수 값, 변수 pi에는 3.141592라는 실수 값, 변수 message에는 “I love Python”이라는 문자열 값이 각각 할당되어 있음.

- 파이썬의 내장함수 type()을 사용해서 살펴보면 num은 int 클래스, pi는 float 클래스, message는 str 클래스 자료형임을 알 수 있음

- num이라는 변수에 정수 값이 할당되면 변수의 자료형이 int 형으로 결정됨

- 이와 같이 프로그램이 실행되는 과정에서 자료형이 결정되는 방식을 **동적 형 결정** dynamic typing이라고 함

2.5 자료형의 의미와 자료형 확인

- 동적 형 결정과 정적 형 결정(용어해설)

  - **동적**dynamic : 어떤 행위가 프로그램이 실행되는 도중에 일어나는 것을 의미
  - **정적**static : 이와 달리 어떠한 행위가 프로그램이 실행되기 전에 미리 결정되는 것을 의미
  - 동적 형 결정은 프로그램의 동작이 유연함
  - 정적 형 결정은 잘못된 값을 넣거나, 서로 연산할 수 없는 데이터를 가지고 연산을 실행하려는 동작을 프로그램 수행 전에 소스코드 해석 단계에서 걸러낼 수 있음

2.5 자료형의 의미와 자료형 확인

- 파이썬의 자료형을 결정하는 할당 연산자

  - foo = 100에서 foo 변수는 int 형을 참조하는 변수
  - foo = 'Hello'를 통해 foo에 문자열을 할당하면 foo는 str 클래스를 참조하는 변수

변수 foo는 정수 형 객체를 참조하다가 나중에 문자열 형 객체를 참조할 수 있다.
이 변수가 참조하는 자료형은 변경 가능하다.
이를 동적 형 결정(타이핑)방식이라 한다.

대화창 실습: 동적 형결정의 이해

```
>>> foo = 100          # foo 변수에 정수형 객체를 할당함
>>> type(foo)          # foo 변수의 자료형은 'int'
<class 'int'>
>>> foo = 'Hello'      # foo 변수에 문자열 객체를 할당함
>>> type(foo)          # foo 변수의 자료형은 'str'
<class 'str'>
```

2.5 자료형의 의미와 자료형 확인

- 정적 타이핑 vs 동적 타이핑

정적 타이핑 static typing	동적 타이핑 dynamic typing
-어떤 변수가 정수형으로 미리 결정되어 있는 경우 프로그램이 실행되는 도중에 문자열과 같은 다른 자료형의 값을 이 변수에 집어넣을 수 없음.	- 특정 변수에 새로운 값을 넣으면 그 변수의 자료형이 변경될 수 있음을 의미함.

2.5 자료형의 의미와 자료형 확인

대화창 실습: 다양한 자료형의 이해

```
>>> l = [100, 300, 500, 900]          # 여러 숫자를 담을 수 있는 리스트 자료형
>>> type(l)
    <class 'list'>
>>> d = {'apple': 3000, 'banana': 4200} # 딕셔너리 자료형
>>> type(d)
    <class 'dict'>
>>> t = ('홍길동', 30, '율도국의 왕')      # 튜플 자료형
>>> type(t)
    <class 'tuple'>
```

파이썬은 다양한 자료형을 제공함.
리스트, 딕셔너리, 튜플에 대해서는 5, 6장에서 상세하게 알아볼 예정

2.5 자료형의 의미와 자료형 확인

- **문자열 변환 함수 str()**

  - str() 함수는 인수로 입력된 값을 문자열 객체로 만들어서 반환
  - 정수형 데이터 값인 숫자 100과, 실수형 데이터 값인 숫자 123.5를 str() 함수의 인자로 넘겨주면 따옴표(‘ ’)로 둘러싸인 문자열 값이 반환됨
  - 리스트형인 [‘A’, ‘B’, ‘C’]를 str() 함수의 매개변수로 넘겨줘도 리스트의 요소인 문자들과 혼동되지 않도록 큰따옴표(“ ”)로 둘러싸인 문자열 객체가 반환됨

2.5 자료형의 의미와 자료형 확인

대화창 실습: str() 함수 실습

```
>>> str(100)        # 정수 100을 문자열 '100'으로 변환시키는 함수
'100'
>>> str(123.5)          # 실수를 문자열 자료형으로 변환시키는 함수
'123.5'
>>> x = ['A', 'B', 'C']
>>> str(x)
"['A', 'B', 'C']"
>>> x = ["A", "B", "C"]
>>> str(x)
"['A', 'B', 'C']"
```

str() 함수는 여러가지 자료형의 값을 문자열 형으로 변환시켜 준다

2.6 문자열 자료형

- 연속된 문자로 이루어진 **문자열**string 자료형에 대한 처리도 가능
- 문자 하나로 구성된 문자와 여러 문자로 이루어진 문자열을 동일하게 취급
- 작은따옴표(‘ ’), 큰따옴표(“ ”) 모두 사용이 가능

```
>>> txt1 = '고양이 이름은 "미미"야'
>>> txt1
    '고양이 이름은 "미미"야'
```

```
>>> txt2 = "고양이 이름은 '미미'야"
>>> txt2
    "고양이 이름은 '미미'야"
```

2.6 문자열 자료형

- 큰따옴표 내에 **“햇님이 좋아!”**와 같은 큰따옴표를 가진 문자열을 넣어주면 에러 발생

```
>>> txt3 = "친구가 "미미가 좋아!"라고 말했다."
  File "", line 1 txt3 = "친구가 "미미가 좋아!"라고 말했다."
  ^
  SyntaxError: invalid syntax
```

아래와 같이 ₩"라고 입력해야 화면에 따옴표가 출력됨

```
>>> txt3 = "친구가 \"미미가 좋아!\"라고 말했다."
>>> txt3
  "친구가 "미미가 좋아!"라고 말했다."
```

2.6 문자열 자료형

- 문자열은 둘 이상이 연속적으로 나타나거나 중간에 공백 문자나 줄바꿈 문자가 있더

```
>>> txt4 = 'Hello ''Python'
>>> txt4
    'Hello Python'
```

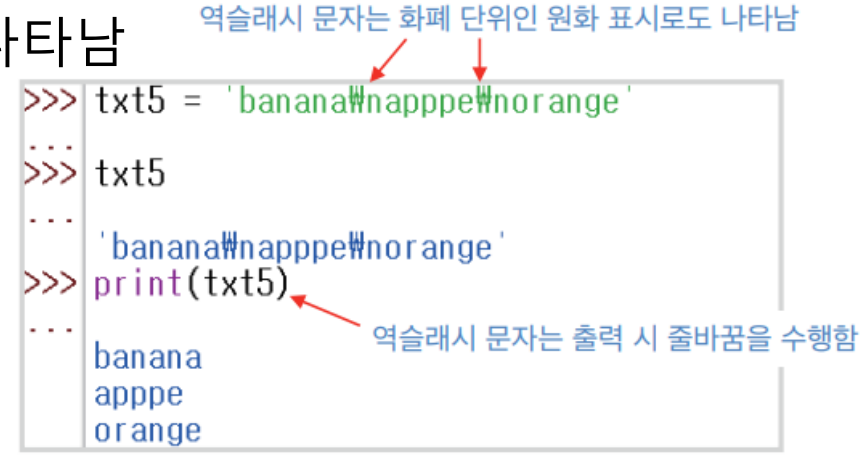
윈도를 비롯한 여러 한글 운영체제에서 역슬래시 문자는 ₩로 나타남. 그러나 IDLE에서는 아래와 같이 출력됨

- 여러 줄의 문자열을 표현하기 위해서는 \n 문자를 삽입

```
>>> txt5 = 'banana\napple\norange'
>>> txt5
    'banana\napple\norange'
>>> print(txt5)
    banana
    apple
    orange
```

2.6 문자열 자료형

- 이스케이프escape 문자
  - ₩n, ₩t
- print() 함수 내의 입력 값으로 사용시 ₩n은 줄바꿈을 수행
- ₩t는 탭 문자의 삽입 기능을 수행
- 한글 윈도 운영체제의 키보드에서는 이스케이프 문자 역슬래시는 화폐의 단위를 표기하는 원 표시(₩)로 나타남



[그림 2-10] 한글 윈도우에 나타나는 이스케이프 문자의 표시

2.6 문자열 자료형

- 따옴표 3 개로 둘러싸는 방법

  - 줄 바꿈을 포함한 문장을 표현할 때
  - 큰따옴표, 작은따옴표가 동시에 포함된 문장을 표현할 때

```
>>> txt6 = '''Let's go'''
>>> txt6
    "Let's go"
>>> txt7 = '''큰따옴표(")와 작은따옴표(')를 모두 포함한 문장'''
>>> txt7
    '큰따옴표(")와 작은따옴표(')를 모두 포함한 문장'
>>> long_str = """사과는 맛있어
    맛있는 건 바나나"""
>>> long_str
    '사과는 맛있어\n맛있는 건 바나나'
>>> print(long_str)
    사과는 맛있어
    맛있는 건 바나나
```

2.6 문자열 자료형

잠깐 - 문자열의 출력

파이썬의 대화창에서는 다음과 같은 두 가지 방식으로 문자열을 살펴볼 수 있다.

```
>>> txt = 'hello'
>>> txt           # txt는 문자열 객체로 문자열이 출력됨
'hello'
>>> print(txt)    # txt가 가진 값이 hello로 출력됨
hello
```

이때 프롬프트에서 txt를 입력하면 txt가 참조하는 객체인 문자열의 내용 'hello'가 따옴표와 함께 출력되며, print(txt)를 입력하면 txt가 가진 값 hello가 화면에 출력된다.
또한 문자열의 길이가 길어서 한 줄을 넘어설 경우 또는 파이썬 표현식의 길이가 한 줄을 넘어서는 경우에도 **역슬래시를 넣어서 두 줄이 이어짐을 나타낼 수 있다**.

```
>>> long_statement = '어제는 하루종일 비가 내렸어 \
자욱하게 내려앉은 먼지 사이로'
>>> long_statement
'어제는 하루 종일 비가 내렸어 자욱하게 내려앉은 먼지 사이로'
>>> result = 12345678901234567890 + 987654321 \
        + 4567890  # 한 줄을 넘어서는 긴 표현식을 연결하기 위해 \ 사용
>>> result
12345678902222679000
```

Leistung ist nicht alles / Keinen Studierenden zurücklassen

DAEJEON UNIVERSITY
SINCE 1980
대전대학교