



모두를 위한 파이썬 프로그래밍

3주 전자화된 데이터 저장방식과
변수의 이해

유길상

학습내용 - 생각해보기

- ◆ 컴퓨터의 계산원리
- ◆ 컴퓨터에서 다양한 종류의 메모리를 사용하는 이유는?
- ◆ 휴대폰에 저장된 사진과 동영상을 삭제해도 복구가 가능한 이유는?
- ◆ 변수의 종류가 다양한 이유는 무엇일까?



Contents

01

전자데이터의 저장방식

컴퓨터의 계산원리, 데이터 저장방식

02

변수

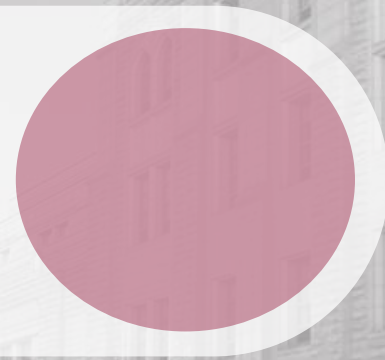
변수의 이해, 변수의 종류

03

자료형 변환

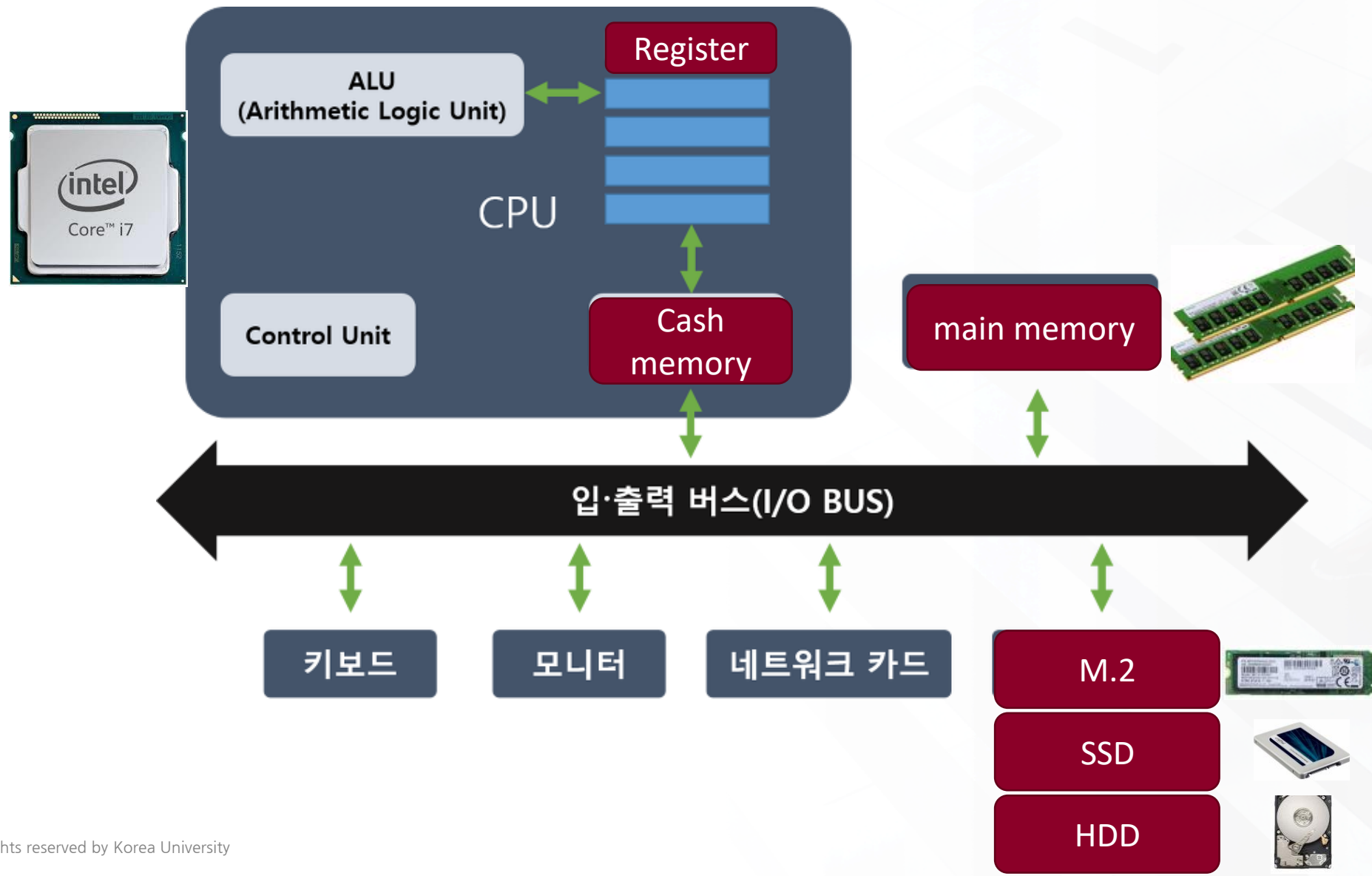
변수에 저장된 자료의 형변환

전자 데이터의 저장방식



1. 컴퓨터가 계산하는 방식

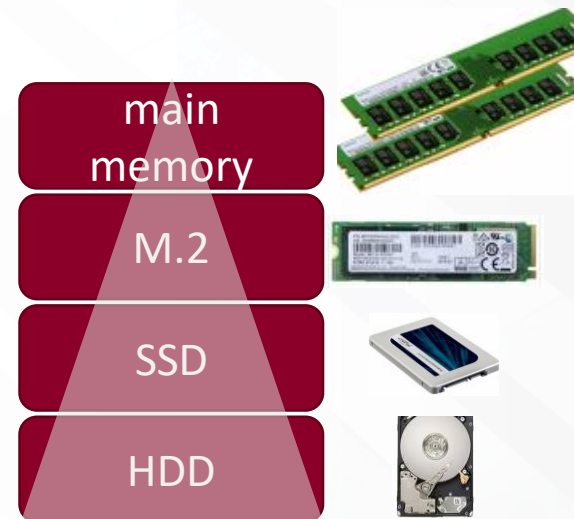
메모리의 활용, 피라미드 구조



2. 메모리에 데이터를 저장하는 방식

휴대폰, 컴퓨터에 저장된 사진, 동영상은 어떻게 복구가 가능할까?

- 전자 메모리에 데이터가 저장되는 방식
 - 인덱스(FAT) 영역과 데이터 저장영역으로 구분된다.
 - FAT 영역 파일 이름과 날짜, 용량 정보, 위치정보가 저장되어 있다
 - 데이터 저장영역에는 데이터가 저장되어 있다.
 - 기존 데이터에 새로운 데이터를 기록할 때 지우지 않고 저장해도 된다.
- 전자 데이터에서 삭제 명령을 수행하는 방식
 - 데이터 저장영역의 데이터를 지우지 않음
 - 인덱스 영역의 파일정보만 지움



3. 컴퓨터가 계산하는 방식

1) 메모지(메모리)를 활용하여 계산

=, 오른 쪽 값을 왼쪽 메모지에 기록하라

```
post_it_a = 10
```

```
post_it_b = 20
```

```
post_it_c = post_it_a + post_it_b
```

```
Print(post_it_c)
```

2) 메모지에 적힌 의미는?

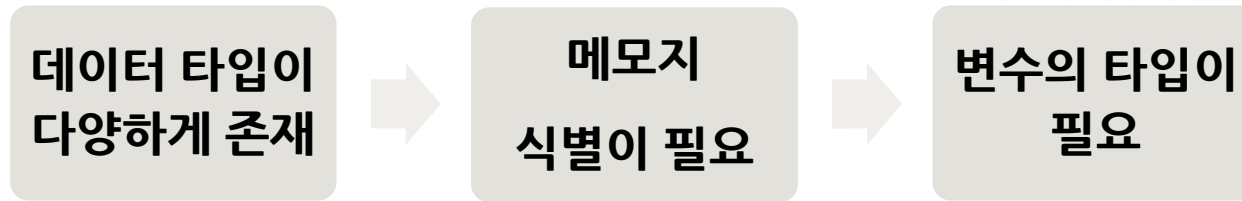
숫자?

버스번호? 번지?



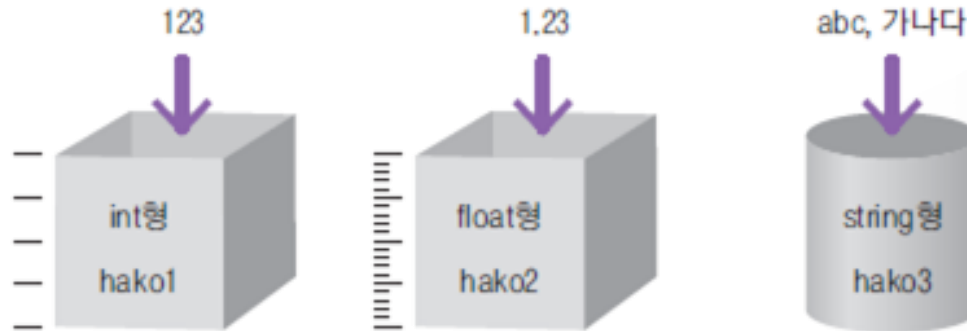
3. 컴퓨터가 계산하는 방식

2) 메모지의 의미는?



- 데이터 타입

ex) int - 정수형, float - 실수형, string - 문자

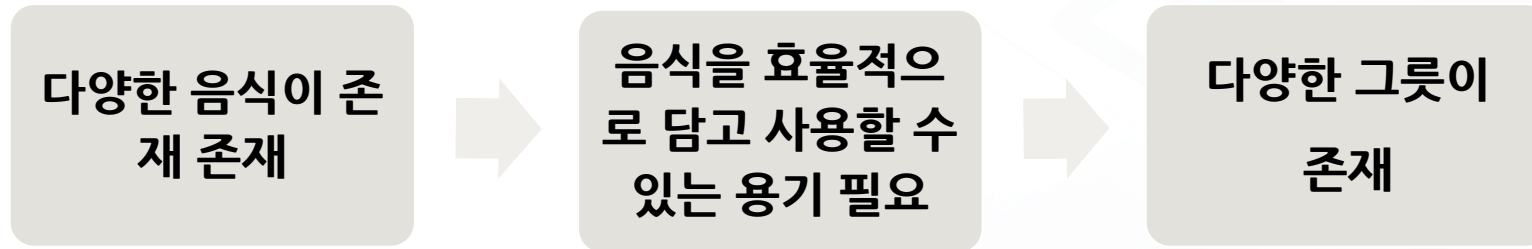


Key Point

데이터 의미를 식별을 위해 변수의 종류도 다양하게 존재한다.

3. 컴퓨터가 계산하는 방식

음식을 담는 그릇이 다양하게 존재하는 이유



- 음식의 종류와 그릇들

Ex) 밥(밥그릇), 국(대접), 음료(컵), 스파게티(접시), 샐러드(넓은 접시), 피자(피자 박스), 단체급식(식판) 등

Key Point

데이터를 메모리 공간에 효율적으로 저장하기 위해 다양한 변수가 필요하다

4. 변수의 기본 자료형

- 정수형(integer type) : 자연수를 포함해 값의 영역이 정수로 한정된 값.
-5 0 465
- 실수형(floating-point type) : 소수점이 포함된 값.
-0.5 1.0 465.0564 3.141
- 문자형(string type) : 값이 문자로 출력되는 자료형.
Korea 파이썬 ab24c
- 불린형(boolean type) : 논리형으로, 참(True) 또는 거짓(False)을 표현할 때 사용.
True False

[참고] 문자 폰트

- 폰트에 따른 가독성 비교

```
1 print('0o0i1')
2 print(' 고려대학교 Python')
```

<바탕체 (font)>

D2Coding-Ver1.3.2-20180524-all.ttc

```
1 print('0o0i1')
2 print(' 고려대학교 Python')
```

< D2Coding (font) >

<https://github.com/naver/d2codingfont/releases>

- 아래 코드를 복사해서 붙여넣고 실행 결과를 예측해 보자

```
print('고려대학교')
print("모두를위한파이썬")
```



```
print('고려대학교')
^
SyntaxError: invalid character
```

```
1 print('고려대학교')
2 print("모두를위한파이썬")
```

변수

02

1 변수의 필요성

■ 지난 주 실습에 대한 복습

- 두 숫자에 대한 기초 연산

```
1 # 연산자 연습
2 print(10 + 7)
3 print(10 - 7)
4 print(10 * 7)
5 print(10 / 7) # 나누기 계산
6 print(10 % 7) # 나머지 계산
```

- 만약 20과 17에 대한 연산 결과를 알고 싶다면? 22와 8은?

```
1 # 연산자 연습
2 print(20 + 17)
3 print(20 - 17)
4 print(20 * 17)
5 print(20 / 17) # 나누기 계산
6 print(20 % 17) # 나머지 계산
```

```
1 # 연산자 연습
2 print(22 + 8)
3 print(22 - 8)
4 print(22 * 8)
5 print(22 / 8) # 나누기 계산
6 print(22 % 8) # 나머지 계산
```

1 변수의 필요성

■ 자주 사용되는 값을 변수로 저장해 두면 코딩이 편리하다

```
1 # 연산자 연습
2 print(22 + 8)
3 print(22 - 8)
4 print(22 * 8)
5 print(22 / 8) # 나누기 계산
6 print(22 % 8) # 나머지 계산
```



```
1 A = 22
2 B = 21
3 # 연산자 연습
4 print(A - B)
5 print(A + B)
6 print(A * B)
7 print(A / B) # 나누기 계산
8 print(A % B) # 나머지 계산
```

- A가 15이고 B가 13일 경우 각각의 연산자에 대한 결과값은?
- A가 16이고 B가 12일 경우 각각의 연산자에 대한 결과값은?
- A가 17이고 B가 11일 경우 각각의 연산자에 대한 결과값은?

2 변수

■ 변수(variable)

- 변할 수 있는 값이 저장되는 공간, 그 공간을 나타내는 이름
- 변수의 선언 → 값 할당 → 참조해서 사용

■ 변수의 선언과 할당 예

- 변수에 값을 할당하기 위한 대입문(=)를 사용한다. 오른 쪽 값을 왼쪽에 넣어라

```
1 A = 22
2 B = 21
3 # 연산자 연습
4 print(A, B)
```

- 22라는 숫자를 “A”라고 불리는 공간(변수)에 저장
- 21이라는 숫자를 “B”라고 불리는 공간(변수)에 저장
- A와 B 변수에 들어있는 내용을 출력하라

3 변수에 값 넣기

■ 변수 선언, 변수 할당, 변수 참조 예

- ✓ 2행에서 할당된 PI변수의 값을 꺼내어 사용하고 있음 → 변수 참조

```
1  PI = 3.14 # 3.14 실수를 저장
2  x = 5 * (3/2) + 3 * 2 - PI # 여러가지 연산을 수행한 결과를 저장
3  z = "python" # 문자열을 저장
4  x = x + 1 # 변수에 대한 연산
5  i = j = k = 1 # 여러가지 변수에 같은 값을 저장
```

✓ 주의


- 대입문(=)은 “같다” 를 의미하는 것이 아니다
- 대입문의 왼쪽에 값이 나올 수 없다. (예: 1 = x)

3 변수에 값 넣기

■ 변수의 동시 할당(대입)

- ✓ 각각의 변수에 값을 동시에 할당 할 수 있다
- ✓ 형식: 변수명들의 리스트 = 변수에 저장 할 값들의 리스트

```
1  a, b, c = 10, 20, 30
2  print(a)
3  print(b)
4  print(c)
5  a, b, c = c, b, a
6  print(a, b, c)
```



10
20
30

- ✓ 6행의 연산 결과는 어떻게 될까?



4 변수를 이용하여 연산하기

- 변수에 저장된 값을 바탕으로 연산을 수행 할 수 있음

```
1 a = 10
2 b = 20
3 print(a + b)
4 print(a - b)
5 print(a * b)
6 print(a / b)
```

```
1 a = 10
2 print(a)
3 a = a + 5
4 print(a)
```

- 오른쪽 예제의 1행을 삭제하면 결과가 어떻게 나타나는가?

Traceback (most recent call last):

File "C:/Users/유길상/PycharmProjects/exam1/exam2.py", line 2, in <module>

print(a)

NameError: name 'a' is not defined

5 변수이름 만들기

■ Naming rule

- ✓ 문자, 숫자, _ 밑줄로 구성, 공백이 들어갈 수 없음, 길이 제한이 없음
- ✓ 문자나 밑줄로 시작, 숫자로 시작 할 수 없음
- ✓ 파이썬에서 사용되는 키워드(keyword)는 변수 이름으로 사용 불가
 - 예) import, print, for, while, if ...
- ✓ 대소문자가 구별됨
 - KoreaUniversity, koreaUniversity, KOREAUNIVERSITY는 모두 다른 변수명

자주사용되는 용어

식별자 (identifier) : 사용자가 만들어 주는 단어(변수나 클래스명)

키워드 (keyword) : 파이썬 언어에서 특별히 의미가 부여된 단어

5 변수이름 만들기

■ Naming Guide (권고 사항일 뿐 필수 사항은 아님)

- ✓ 의미있는 이름으로 주면 이해하기 쉬움
 - `A = 30, b = 50` # 어떤 의미인지 전혀 알 수 없음(임시 계산을 위한 용도로 사용)
 - `speed = 90, score = 100`
- ✓ 이름이 길어질 경우, 언더바 "_" 를 이용하여 변수명을 만듦(snake_case)
 - `this_is_my_score`
 - `average_score`

Camel case and Snake case

Python recommend **UpperCamelCase** for class names, **CAPITALIZED_WITH_UNDERSCORES** for constants, and **lowercase_separated_by_underscores** for other names.



6 상수

■ 변수(variable) vs 상수(constant)

- ✓ 변수: 변할 수 있는 값
- ✓ 상수: 변하지 않는 값

```
1  PI = 3.14
2  radius = 5
3  print(PI * radius * radius)
```

- ✓ 실제로는 값을 변화시킬 수 있기 때문에 추상적인 의미의 상수를 뜻함
- ✓ "상수"의 의미 전달을 위해 변수와는 다른 명명 스타일 적용
 - 상수는 대문자만으로 이름을 작성하여 구별하고 있음 (예: PI)

실습

- 아래와 같이 작성하고 결과를 확인해 봅시다

```
1 a = 10
2 b = 20
3 print(a + b)
4 print(a - b)
5 print(a * b)
6 print(a / b)
7 print(a % b)
```

- a와 b의 값을 각각 100, 3으로 변경하고 결과를 확인 해 봅시다.

실습

■ 아래와 같이 작성하고 결과를 확인해 봅시다

```
1 radius = 1.5
2 print('원의 둘레는 ', 2 * 3.141592653589793 * radius, '입니다')
3 print('원의 면적은 ', 3.141592653589793 * radius * radius, '입니다')
```

■ 위의 프로그램을 가장 이상적으로 수정해보자

- ✓ (프로그램 간략화) 자주 사용되는 파이값을 변수(pi)로 지정
- ✓ (변수 네이밍) 상수로 사용되는 변수 이름을 대문자로 설정

실습

■ 여러 값을 함께 출력하기

- ✓ `print(값)` : 단일 값의 출력
- ✓ `print(값1, 값2, 값3...)` : 복수개의 값을 출력

```
1 age = 20
2 student_id = 2019
3 grade = 1
4 print('나이:', age, ' 학년:', grade, ' 학번:', student_id)
```



나이: 20 학년: 1 학번: 2019

실습

■ 값 입력 받아 주어진 형태로 출력하기

- ✓ `input("메시지")`
 - 메시지를 출력한 후,
 - 사용자가 입력한 값을 문자열 형태로 받아들인다.

```
1 name = input("이름을 입력하세요 >>")  
2 age = input("나이를 입력하세요 >> ")  
3 print("당신의 이름은 ", name, "이고, 나이는 ", age, "살 입니다")
```



이름을 입력하세요 >> 홍길동

나이를 입력하세요 >> 30

당신의 이름은 홍길동 이고, 나이는 30 살 입니다

실습

■ 값을 입력 받아 계산해서 출력하기

- ✓ 나이를 입력하고 10년 뒤 나이를 알려주는 프로그램을 작성해 보자

나이를 입력하세요 >> 21
당신은 10년 뒤 31 살 입니다.

```
1 age = input("나이를 입력하세요 >> ")
2 print("당신은 10년 뒤", age + 10, "살 입니다.")
```

변수의 타입 확인하기

```
print(type(age)) # age 변수형 확인
# 문자형(str)은 산술 연산이 되지 않는다.
```



자료형 변환

03

자료형 변환

■ 정수형과 실수형 간 변환

- ✓ `int()` 함수 : 실수형 또는 문자형자료를 정수형으로 변환해 주는 함수.

```
1 a = 10.3
2 b = 3
3 print(int(a) + b)
```

13

자료형 변환

■ 정수형과 실수형 간 변환

- ✓ float() 함수 : 정수 또는 문자형 자료를 실수형으로 변환해 주는 함수

```
1 a = 10
2 print(a)
3 a = float(a)
4 print(a)
```

```
1 a = 10
2 b = 3
3 print(a / b)
```

결과값은 자동으로
실수형으로
계산됨

```
3.3333333333333335
```

자료형 변환

■ 숫자형과 문자형 간 변환

- ✓ `str()` 함수 : 기존의 정수형이나 실수형을 문자열로 바꿔 준다. 문자 간의 덧셈은 숫자 연산이 아닌 붙여서 출력된다.

```
1 a = 1.645
2 b = 4632.25
3 print(a+b)
4 print(str(a) + str(b))
```


실습

■ 정수값을 입력 받아 처리하기

- ✓ 나이를 입력하고 10년 후의 나이를 알려주는 프로그램을 작성해 보자

나이를 입력하세요 >> 21

당신은 10년 뒤 31 살 입니다.

```
1 age = input("나이를 입력하세요 >> ")
2 print("당신은 10년 뒤", age + 10, "살 입니다.")
```

```
print("당신은 10년 뒤", age + 10, "살 입니다.")
```

TypeError: can only concatenate str (not "int") to str



```
1 age = input("나이를 입력하세요 >> ")
2 print("당신은 10년 뒤", int(age) + 10, "살 입니다.")
```

```
1 age = int(input("나이를 입력하세요 >> "))
2 print("당신은 10년 뒤", age + 10, "살 입니다.")
```

실습

■ 원의 반지름 숫자를 입력받아 처리하기

- ✓ 원의 반지름을 입력하면 원의 둘레와 면적을 출력하는 프로그램을 작성하시오.

```
원의 반지름을 입력 하시오 >> 1.5
원의 둘레는 9.42477796076938 입니다
원의 면적은 7.0685834705770345 입니다
```

```
를 입력 하시오>>'))
'입니다')
05.01.15 '입니다')
```

Thank you

