컴퓨터 및 프로그래밍입문 2주차

한상곤(sangkon@pusan.ac.kr)

2. 변수와 연산자

2. 변수와 연산자 - 목차

2. 변수와 연산자 - 학습목표

2.1 **변수의 개념과 사용법**에 대해 이해한다.
2.2 **변수의 자료형**에 대해 이해한다.
2.3 **식별자와 키워드**에 대해서 이해한다.
2.4 ~~연산자의 개념과 종류에 대해 알아본다.~~
2.5 ~~다양한 연산자를 활용하여 계산을 수행할 수 있다.~~
2.6 **문자열**을 사용할 수 있다.
2.7 **할당 연산자의 동작**을 이해한다.
2.8 수를 컴퓨터에서 **표현**하는 방법을 이해한다.
2.9 ~~비트 단위 연산자를 통해 수를 조작하는 법을 이해한다.~~
2.10 ~~주석문의 개념과 사용법에 대해 알아본다.~~

2.1 변수와 친해지기와 2.9 주석문

- 원의 반지름을 사용해서 면적과 둘레를 구하는 코드 #1

```
print('원의 반지름', 5.0) # 출력문 <-- 주석문
print('원의 면적', 3.14 * 5.0 * 5.0) # 계산 결과를 출력 <-- 주석문
print('원의 둘레', 2.0 * 3.14 * 5.0) # 원의 둘레를 출력 <-- 주석문
```

- 원의 반지름을 사용해서 면적과 둘레를 구하는 코드 #2

```
"""
반지름을 사용해서 면적과 둘레를 계산하는 코드 입니다.
"""
print('원의 반지름', 6.0)
print('원의 면적', 3.14 * 6.0 * 6.0)
print('원의 둘레', 2.0 * 3.14 * 5.0)
```

2.1 변수^{variable}를 활용해보자.

프로그램을 작성하다보면 계속해서 값(value)을 변경시켜 주어야하는 경우, 하나라도 실수를 하게 되면 프로그램의 오류(error)가 발생합니다. 값을 지칭하는 변수를 도입하면 이런 오류를 막을 수 있습니다.

```
radius = 4.0 # 변수 선언과 값 할당
print('원의 반지름', radius)
print('원의 면적', 3.14 * radius * radius)
print('원의 둘레', 2.0 * 3.14 * radius)
```

2.1 변수와 식별자

- 변수(variable)
  - 변수는 메모리 주소와 연결되거나 메모리 주소로 식별될 수 있음
  - 컴퓨터 프로그래밍에서 변수는 관련 기호 이름(식별자)과 쌍을 이루는 추상적인 저장 위치를 뜻 하며, 변수 이름은 컨텍스트(문맥)에 따라 변수 자체를 참조하는 것 외에도 저장된 값을 참조하는 것이 일반적인 방법
  - 식별자는 런타임(코드가 메모리에서 실행) 동안 값에 바인딩(이름과 값에 연결)될 수 있으므로 변수 값은 프로그램 실행 과정에서 변경될 수 있음
- 식별자(identifier)
  - 사용자가 정의하는 변수나 함수에 대해 구별가능한 이름을 부여해야 하는데, 이와 같이 서로 구별되는 이름을 식별자라 함
  - 다른 메모리 위치에는 서로 다른 이름을 부여해야 함

2.1 메모리와 메모리 주소

- 메모리(memory)
  - 데이터가 저장되어 읽기, 쓰기 그리고 덮어쓰기를 하는 곳
  - 메모리라고도 불림
- 메모리 주소(address)
  - 메모리에 데이터를 저장한 곳의 위치로 16진수로 표현
  - 저장된 데이터를 읽고 쓰기 위해서는 데이터가 저장된 곳(공간 또는 위치)이 어디인가를 알아야 함

---

https://docs.python.org/3.10/reference/datamodel.html#objects-values-and-types

2.1 비트^{bit}와 바이트^{byte}

- 비트(bit)
  - `0`과 `1`을 이용하여 정보를 표현으로 컴퓨터에서 사용하는 정보 표현의 `최소의 단위`
  - 한 비트만으로 표현 가능한 정보가 너무 적기 때문에 주로 8비트 단위로 저장
- 바이트(byte)
  - 8비트 단위를 바이트라고 함
  - 2^8 = 256가지의 서로 다른 상태 정보를 표현

2.2 변수의 선언

```
width = 10 # 변수의 선언
height = 5 # 변수의 선언
rectangle_area = width * height
print('사각형의 면적 :', rectangle_area)
```

`식별자`는 1) 문자와 숫자, 밑줄 문자(`_`)로 구성, 2) 첫 글자는 반드시 문자나 밑줄 문자 `_`로 시작해야 하며, 3) 대문자와 소문자는 구분(Count $\neq$ count), 4) 식별자의 길이에 제한은 없지만, `키워드`는 식별자로 사용할 수 없습니다.

2.2 변수의 선언시 주의사항

a, b, n, m,과 같은 단순한 이름의 식별자로도 그 기능을 구현할 수 있습니다. 하지만 프로그램이 복잡해지면 `walk_distance`, `num_of_hits`, `english_dict`, `student_name`과 같이 그 의미를 명확하게 이해할 수 있는 식별자를 사용하는 것을 권장합니다.

키워드(keyword) 혹은 예약어(reserved word)는 이미 예약된 문자를 뜻 합니다(import, for, if, def, class...).

2.2 중복된 변수

```
width = 10 # 변수의 선언과 할당
height = 5 # 변수의 선언과 할당
width = 20 # 변수의 선언과 할당
rectangle_area = width * height
print('사각형의 면적 :', rectangle_area) # 계산 결과는?
```

2.5 자료형의 의미와 자료형 확인

자료형(data type)은 프로그래밍 언어에서 처리할 수 있는 데이터의 형태를 말하는 것으로 부울형, 숫자형(정수, 실수, 복소수), 문자열, 리스트, 튜플, 집합, 딕셔너리 등이 있으며 객체가 어떤 자료형인지를 알려주는 `type`이라는 함수를 제공합니다.

```
num = 85
print(type(num)) # int
pi = 3.14159
print(type(pi)) # float
message = "Good morning"
print(type(message)) # str
```

num이라는 변수에 정수 값이 할당되면 변수의 자료형이 int 형으로 결정, 이와 같은 방식을 동적 자료형 결정(dynamic typing)이라고 합니다.

2.5 동적과 정적*

동적은 프로그램의 동작이 유연하고, 정적은 잘못된 값을 넣거나, 서로 연산할 수 없는 데이터를 가지고 연산을 실행하려는 동작을 프로그램 수행전에 소스코드 해석 단계에서 걸러낼 수 있음

- 동적(dynamic); 프로그램이 실행되는 도중에 일어나는 것을 의미
- 정적(static); 실행되기 전에 미리 결정되는 것을 의미

2.5 다양한 자료형

```
l = [100, 300, 500, 900]
print(type(l)) # list
d = {'apple': 3000, 'banana': 4200}
print(type(d)) # dict
t = ('홍길동', 30, '율도국의 왕')
print(type(t)) # tupe
```

2.6 문자열 자료형

연속된 문자로 이루어진 문자열(string) 자료형은 문자 하나로 구성된 문자와 여러 문자로 이루어진 문자열을 동일하게 취급하는것이 특징으로 작은따옴표(‘’), 큰따옴표(“”) 모두 사용이 가능합니다.

```
txt1 = '강아지 이름은 "햇님"이야'
txt2 = "강아지 이름은 '햇님'이야"
print(txt1) # '강아지 이름은 "햇님"이야'
print(txt2) # "강아지 이름은 '햇님'이야"
txt5 = 'banana\napple\norange'
print(txt5) # 출력 결과는?
```

2.6 문자열 자료형 - 이스케이프 문자

이스케이프(escape) 문자

- `\n`
- `\t`

파이썬의 변수로 `\n`이나 `\t`를 가진 문자열은 "hello\nworld"나 "hello\tworld"와 같이 `\n`, `\t`를 문자 그대로 저장하지만, print() 함수 내의 입력 값으로 사용시 `\n`은 줄바꿈을 수행 `\t`는 탭 문자의 삽입 기능을 수행합니다.

한글 윈도 운영체제의 키보드에서는 이스케이프 문자 역슬래시는 화폐의 단위를 표기하는 원표시(₩)로 나타나기 때문에 표기법에 주의할 것

2.6 문자열 자료형 - """ 문자열

```
txt6 = '''Let's go'''
txt7 = '''큰따옴표(")와 작은따옴표(')를 모두 포함한 문장'''
long_str = """사과는 맛있어
맛있는 건 바나나
"""
print(txt6) # "Let's go"
print(txt7) # '큰따옴표(")와 작은따옴표(')를 모두 포함한 문장'
print(long_str) # 출력 결과는?
```

2.7 수치 자료형 - 정수와 실수

정수(int) - 음의 자연수, 0 그리고 자연수를 포함
실수(float) - 소수점 이하의 값 포함

부동소수점수(floating point number)는 미세한 오류를 포함합니다.

```
print(0.1 + 0.1 == 0.2) # True
print(0.1 + 0.1 + 0.1 == 0.3) # False
print(0.1 + 0.1 + 0.1) # 0.30000000000000004
print(0.1 * 19.0) # 1.9000000000000001
```

정수와 실수를 구분해서 사용하세요.

```
print(14 / 5) # 2.8
print(14 // 5) # 2
```

2.7 수치 자료형 - 복소수

```
c1 = 2 + 3i # error
c1 = 2 + 3j
print(c1.real) # c1의 실수부 출력, 2.0
print(c1.imag) # c1의 허수부 출력, 3.0
c2 = 5 + 6j
c3 = c1 + c2
print(c3) # 7+9j
print(c1.conjugate()) # 켤레 복소수
print(abs(c1)) # 복소수의 크기
```

2.8 여러 가지 연산자 - 할당문

우변의 값을 좌변의 변수에 대입 또는 할당assign하라는 의미
num1 = num2 = num3 = 200과 같이 다중 할당multiple assignment도 가능

```
num1 = num2 = num3 = 200 # 다중 할당문
print(num1, num2, num3) # 200 200 200
num4, num5 = 300, 400 # 동시 할당문
print(num4, num5) # 300 400
str = 'world'
'hello' = str # Error!
```

2.8 여러 가지 연산자 - 산술 연산자

```
num = 200
num = num + 100 # 200 + 100
num # 300
num = 200
num = num + 100 # 200 + 100
num # 300
num = num - 100 # 300 - 100
num # 200
num = num * 20 # 200 * 20
num # 4000
num = num / 2 # 4000 / 2
num # 2000.0
```

2.8 여러 가지 연산자 - 비교 연산자와 논리식

```
# 비교 연산자
a, b = 100, 200
print(a == b) # False
print(a != b) # True
print(a > b) # False
print(a < b) # True
print(a >= b) # False
# 논리식
x, y = True, False
print(x and y) # False
print(x or y) # True
print(x not y) # False
```

2.8 여러 가지 연산자 - 비트연산자

```
print(9 & 10) # 8
print(bin(8)) # '0b1000'
print(9 | 10) # 11
print(bin(11)) # '0b1011'
print(9 ^ 10) # 3
print(bin(3)) # '0b11
print(~9) # -10
print(bin(-10)) # '-0b1010'
print(4 << 1) # 00100 -> 01000 => 8
print(4 << 2) # 00100 -> 10000 => 16
print(4 >> 1) # 00100 -> 00010 => 2
print(4 >> 2) # 00100 -> 00001 => 1
```

선택된 연습문제

- 2.5 정사각형의 면적을 구하기 위하여 사용자로부터 밑변의 길이를 정수로 입력받아서 다음과 같이 출력하시오.

```
정사각형의 밑변을 입력하시오 : 40
정사각형의 면적 : 1600
```

- 2.18 사용자의 키보드 입력을 통해 n 값을 입력받아, 주어진 정수 n이 짝수이면 True, 홀수이면 False를 출력하는 코드를 작성해 보라. n이 20, 21인 경우에 대하여 다음과 같이 출력하여라.

```
정수를 입력하세요 : 20
이 수가 짝수인가요? True
```

Appendix

```c
int main()
{
        int n = 1;

        printf("n is %d\n", n);

        int* ptr = &n;
        printf("&n is %p\n", &n);
        printf("ptr is %p\n", ptr);

        *ptr = 128;

        printf("%d\n", n);

        return 0;
}

/*
    n is 1
    &n is 000000EFA3CFF934
    ptr is 000000EFA3CFF934
    128
*/
```