[파이썬]

특징

- 파이썬은 인터프리터 언어이다. 즉 컴파일 없이 한 번에 하나의 명령어만 실행한다.
- 파이썬은 다른 언어와 달리 중괄호{ } 대신 들여쓰기를 사용해서 코드를 구조화 한다.
- 파이썬은 모든것이 객체입니다.
  - 모든 자료구조, 함수, 클래스 등 모두를 객체로 간주한다.
  - 객체(object)는 속성(Attribute)과 함수(Method)를 가진다.

In [ ]:
```
!python --version
```

기본문법

- 주석(#) : 실행 안함, 부가적 설명 (한줄주석 #, 여러줄 주석 """ """)
- 한문장으로 그룹화(;) : 여러문장을 ; 로 연결해서 한줄로 만들 수 있음 (a = 1; b=2; print(a+b))
- 멀티라인(\ ) : 연속 문자 \ (₩) 의 사용으로 줄을 계속 사용가능
- 출력문 : ' ' , " ", ''' ''' , 개행문자(\n, \t)
- 블록 구분 : 들여쓰기(intention)으로 구분 (if, for, while, def, class ...)
- 객체(변수).메소드 : 예) 문자열.메소드, 리스드.메소드, 인스턴스.메소드

[print ("Hello, World")]

In [ ]:
```
print("Hello, World")
```

In [ ]:
```
print("한글도 쓸 수 있어요.")
```

In [ ]:
```
print("안녕하세요.")
print("반갑습니다~~~")
print()
```

In [ ]:
```
1 + 2
```

In [ ]:
```
print(1 + 2 )
print("1 + 2")
```

In [ ]:
```
# print(안녕하세요)
```

[계산기처럼 이용하기]

사칙연산 (더하기(+), 빼기(-), 곱하기(*), 나누기(/))

In [ ]:
```
# 정수 사칙연산
1 + 3
```

In [ ]:
```
1 + 3.0 # 실수
```

In [ ]:
```
5-2, 15*2, 10/4
```

In [ ]:
```
# 실수 사칙연산
1.2 + 5.3 , 3.5 - 5.0, 1.4 * 2, 5/2
```

In [ ]:
```
# 복합 사칙연산
2 + 3 * 4
```

In [ ]:
```
3/2 * 4 - 5/2
```

In [ ]:
```
10 / 5 + (5 - 2) * 2
```

In [ ]:
```
(5 * 4 - 15) + ((5 - 2) * (9-7))
```

- 괄호가 있으면 괄호안을 먼저 계산한다. 하지만 파이썬은 소괄호, 중괄호, 대괄호를 구분하지 않고 모두 소괄호 기호를 사용한다.
- 예를 들어 아래 수식을 파이썬 코드로 나타내면 다음과 같다.

$$100 \div \Big[3 \times \{10 - (3 \times 2)\} + 8\Big]$$

In [ ]:
```
# 수식을 써보세요.
```

[변수(Variable) ?]

' = ' 프로그램에서는 수학식의 '같다' 와는 전혀 다른 의미임

a = 1 등호 식은 변수를 설정한다고 표현하며, 이는 value(값)을 할당(assign)한다는 의미

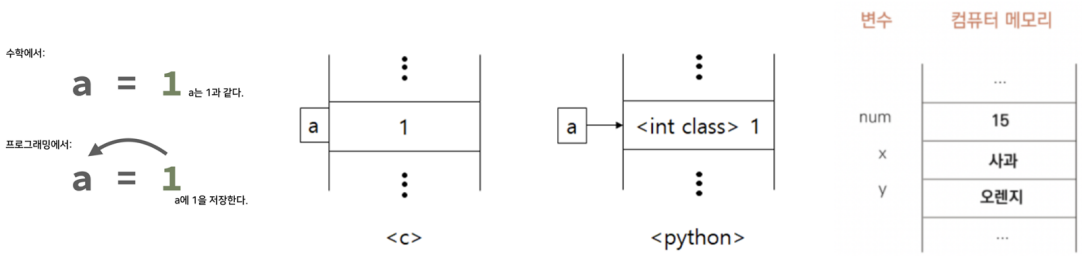
a = 1 --> a에 1을 할당한다 --> [저장한다(c), 가리킨다(참조한다) (파이썬)]

- 참고 : http://www.pythontutor.com/
- 파이썬에서 모든값 (단순한 정수값이든 , 복잡한값(문자열,리스트)) 메모리에 객체로 저장된다.
- 객체는 메모리에 적장된 값을 포함하는 어떤 상자 라고 생각

할당문 (Assign Statement) : 변수이름 = 값

- 할당문은 변수에 값을 지정하기 위해 사용됨
- 할당문의 왼쪽 : 변수

- 할당문의 오른쪽 : 값/객체



- 숫자값 1 이라는 값이 메모리의 어딘가에 할당되는데, 파이썬에서는 이를 객체(object)라고 부른다
- 변수 a 역시 메모리의 어딘가에 할당되는 공간인데, 그 공간에는 1 이라는 객체의 메모리 주솟값이 저장돼 있다.
- 이를 다르게 표현하자면 변수 a 는 1 이라는 값의 객체를 가리키고 있는 것

변수 규칙 (첫번째 문자만 , 빈칸 있음 X)

- 변수명은 **영어(한글가능), 숫자, _ (underscore)**로만 구성한다.
- 변수명은 숫자로 시작할 수 없다.(7person(오류))
- 대소문자를 구분한다. 즉, data와 Data는 다른 변수이다. (`apple` 과 `Apple` 과 `APPLE` 은 모두 서로 다른 변수)

- 특수문자(!@#$%^&*), 공백 은 변수명에 사용할 수 없음 (예외 : _ 사용가능)
- 키워드(keyword)를 변수명으로 사용하면 안된다.
  - False, None, True, and, as, assert, break, class, continue, def, del, elif, else, except, finally, for, from, global, if, import, in, is, lambda, nonlocal, not, or, pass, raise, return, try, while, with, yield

In [ ]:
```
number = 5
python_score = 95
_score = 100
math1 = 80
학생수 = 50    # 한글 변수명도 가능함
print(number,python_score,_score,math1,학생수)
```

In [ ]:
```
# 1score = 23          # syntax 에러, 변수명은 숫자로 시작할 수 없다.
# math score = 80     # syntax 에러, 빈칸 있음
# math-score = 90     # syntax 에러. 특수 기호는 _ 만 가능하다
# mathScore = 90
```

In [ ]:
```
print(math1) # math1 변수의 저장된 값을 확인
print("math1") # 비교
```

[데이터타입 (Data Types) / 자료형)] : 변수의 다양한 타입

[변수] 아래와 같은 데이터타입(형)(자료형) 속성을 갖는다

수치 자료형: int (정수), float (실수), complex (복소수)

부울 자료형: bool (True / False)

군집 자료형: str: "문자열" , list:[리스트], tuple:(튜플), set:{집합}, dict:{사전}

분류	자료형		예
수치 자료형	정수 (int)		..., -3, -2, -1, 0, 1, 2, 3, ...
	실수 (float)		3.14, 5.5, 8.0, 0.54, -3.89, ...
	복소수 (complex)		3+4j, 5.7+2J, 2+9j, 5+1j, ...
부울 자료형	부울 (bool)		True, False
군집 형태 자료형 (컨테이너 자료형)	시퀀스(Sequence) 자료형	문자열 (str)	'hello', "python", 'data', ...
		리스트 (list)	[1,2,3,4], ['red', 'blue'], [3.5, 2.4], ...
		튜플 (tuple)	(1,2,3,4), ('red', 'blue'), (3.4, 5.5, 1.2), ...
	집합 (set)		{1,2,3}, {'red', 'blue'}, {3.5, 1.2}, ...
	사전 (dict)		{1:'one', 2:'two'}, {'red':5, 'blue':2}, ...

파이썬에서는 모든값 (단순한 정수값, 복잡한값(문자열,리스트)) 메모리에 객체 형태로 저장된다.

- 어떤 데이터 값따라 데이터 타입이 정해지고, 데이터 타입이 정해지면 사용되는 연산자와 함수(메소드)가 정해진다

In [ ]:
```
# 데이터형 알아보기
type(3), type("hi")
```

In [ ]:
```
i = 3       # 정수형 변수 할당
f = 1.3     # 실수형 변수 할당
s = "안녕"  # 문자형 변수 할당
b = True    # 논리형 변수 할당
t = (1,2)   # 튜플형 변수 할당
li = [1,2,"list"] #리스트 변수 할당
```

In [ ]:
```
type(i), type(f), type(s), type(b), type(t), type(li)
```

정수형(interger)

- 정수형(Integer)이란 말 그대로 정수를 뜻하는 자료형을 말한다.

In [ ]:
```
# 양의 정수와 음의 정수, 숫자 0을 변수 a에 대입하는 예
a = 123
b = -178
c = 0
print(a,b,c)
```

In [ ]:
```
type(a)
```

In [ ]:
```
a = 10
b = 20
c = a + b   # a와 b를 더하여 c에 저장하시오
a = a + 50  # a의 값을 50 증가하시오
b = b + a   # b의 값을 a 만큼 증가하시오
print(a, b, c)
```

실수형 (float)

- 파이썬에서 실수형(Floating-point)은 소수점이 포함된 숫자를 말한다.
- 소수점표기 방법 : 10.5, 11. , .5
- 과학적표기 방법 : 2.5e5, 2.5E5, 3.25e-4

In [ ]:
```
a = 1.2
b = -3.45
c = 5.0
print(a,b,c)
```

In [ ]:
```
type(c)
```

In [ ]:
```
a = 4.24e14
b = 4.24e-4
c = 5.23e16
d = 5.56e-5
print(a,b,c,d)
```

부울형(bool) : True / False 2가지 값만 있음

- 불(bool) 자료형이란 참(True)과 거짓(False)을 나타내는 자료형이다.
- 불 자료형은 다음의 2가지 값만을 가질 수 있다.
- True - 참 / False - 거짓
- True나 False는 파이썬의 예약어로 true, false와 같이 사용하지 말고 첫 문자를 항상 대문자로 사용해야 한다.

In [ ]:
```
x = True
y = False
print(x)
print(y)
```

In [ ]:
```
type(y)
```

[연산자(Operator)]

산술 연산자 (Arithmetic Operators)

- (덧셈 +), (뺄셈 -), (나눗셈 /), (곱셈 *) 등등

할당 연산자(Assignment Operators)

- (=, += , -=, *=, /=, %=, //= 등등

비교(관계) 연산자 (Comparison Operators)

- (크다 >), (작다 <), (같다 ==), (다르다 !=) 등등

논리 연산자 (Logical Operators)

- (그리고 and), (또는 or) 등등

멤버 연산자 (Membership Operators)

- in , not in

산술 연산자 : 숫자형을 활용하기 위한 연산자(operator)

- 수식(expression) = 피연산자, 연산자의 조합
- 연산자(operator): 연산을 나타내는 기호
- 피연산자(operand): 연산의 대상이 되는 값

In [ ]:
```
# 거듭제곱(**), 나머지(%), 몫 (//)
print(2*2*2*2*2)  # 2x2x2x2x2
print(2 ** 5)     # 2의5승
print(1.5**2)     # 1.5의2승
print(3 * 10 ** 8)
print(13//5)      # 13/5의 몫
print(13%5)       # 13/5의 나머지
```

In [ ]:
```
a = 5
b = 4
print(a + b)
print(a * b)
print(a / b)
```

할당연산자 (Assignment Operators)

- 변수에 값을 할당한다
- **Python** 예: a = 10 --> 10 이라는 'int' 객체로 메모리에 저장된다, a라는 변수는 그 객체를 가르키게 함 (참조, reference).

'=' 기호는 '할당 연산자' 또는 '대입 연산자'라고 함.

- 복합연산자: += 처럼 대입 연산자와 다른 연산자를 합쳐 놓은 연산자 (산술 연산자 간략히 쓰기)

In [ ]:
```
a = 5

a += 5   # a = a + 5 (a값 5 만큼 증가)
a -= 3   # a = a - 5 (a값 5 만큼 감소)
print(a)
a *= 5   # a = a * 5 (a값 5배 증가)
a /= 3   # a = a / 5 (a값 1/5 값으로)
print(a)
a %= 5   # a = a % 5 (나머지)
a //= 3  # a = a // 5 (몫)
print(a)
```

In [ ]:
```
# 변수 사용
a = 10 ; b = 5 ; c = 22 ; d = 3
a += c    # a가 32가 됨 a = a + c
```

```
b -= d     # b가 2가 됨
print(a,b)

a //= 3    # a가 10이됨
c %= 5     # c가 2가 됨
print(a,c)

a = 3; b = 2; c = 5; d = 10
d += b + c - a ** b    # d가 8이 됨
print(d)
```

비교(관계) 연산자 (==, <, >, != ...) : 부등식의 참과 거짓 계산

- 조건이 성립하면 True(참), 성립하지 않으면 False(거짓)
- 비교 연산의 결과는 불 데이터로 출력됨.
- 따라서 **비교 연산**은 **논리 연산**과 함께 이용하는 경우가 많음

- 숫자 비교 , 변수 비교

- 1 == 1 은 "1과 1이 같은가?"를 묻는 조건문이다.
- 이런 조건문은 결과로 True 또는 False에 해당되는 불 자료형을 리턴하게 된다.
- 1과 1은 같으므로 True를 리턴한다.

In [ ]:
```
1 == 1
```

In [ ]:
```
7 > 3
```

In [ ]:
```
2 > 2
```

In [ ]:
```
2 != 1
```

In [ ]:
```
3 <= 3
```

In [ ]:
```
# 비교 연산자를 이용해 연산한  예
x = 5
y = 3

print(x == y) # 5는 3과 같은가 ?
print(x != y)
print(x < y)
print(x > y)  # 5는 3보다 큰가 ?
print(x <= y)
print(x >= y)
```

논리연산자 (and(&), or(|), not) : 부울리언 (Boolean) 연산자

- 논리 연산(logical operation) : 불린 연산(Boolean operation)

- 다양한 조건에 따라 코드가 다르게 실행되도록 작성
- 어떤 조건을 만족하는 참(True)과 만족하지 않는 거짓(False)을 이용
- 논리 연산을 위한 데이터 타입: 불(bool)

- 불 데이터 타입에는 논리 참(True) 혹은 논리 거짓(False)이 있음
- 참(True) 혹은 거짓(False)을 입력할 때 참은 True, 거짓은 False를 입력

- 불 데이터의 경우 논리 연산만 가능

- 논리곱(and): 두 개의 불 데이터가 모두 참일 때만 참이고 나머지는 거짓
- 논리합(or): 두 개의 불 데이터 중 하나라도 참이면 참이고 둘 다 거짓이면 거짓
- 논리부정(not): 하나의 불 데이터가 참이면 거짓이고 거짓이면 참

In [ ]:
```
# 논리 연산(and, or, not)의 예
print(True and False)
print(True or False)
print(not True)
```

In [ ]:
```
a = True
b = False

print(a and b)
print(a or b)
print(not a)
```

In [ ]:
```
# a는 10이상 이고 50 미만 인가 ?
a = 15
(a >=10) and (a < 50 ) # and &
```

In [ ]:
```
(a >=10) & (a < 50 )
```

In [ ]:
```
# a가 1 또는 100 인가 ?
a = 100
(a == 1) or (a == 100) # or |
```

In [ ]:
```
(a == 1) | (a == 100)
```

In [ ]:
```
# a 가 100이 아니다.
a = 100
not (a==100)
```

In [ ]:
```
# 어린이 놀이 기구 탈 수 있는지 ?  ( 안전을 위한 수칙 : 나이10살 이상이고, 키 140 cm)
age = 13   # 나이
height = 135  # 키

age >=10 and height >= 140 # 안전수칙 조건
```

---------- [End] ----------