03.연산자

연산자

- 프로그램 연산자
- . 컴퓨터가 처리 해야할 명령어를 수학적 기호 키워드로 표현한것.
- 연산자 의 종류

연산자	기호
산술 연산자	+ - * / // %
지수 연산자	**
대입(복합) 연산자	= += -= *= /= //= %=
관계(비교) 연산자	> >= < <= == !=
논리 연산자	not and or
삼항 연산자	a if 조건식 else b

- 대입 연산자
- . 변수에 값을 할당 하는 연산자
 - > 변수 = 수식

```
Chap01_intro > ♥ 03_Operator[연산자].py > ...

1 iValue = 2
```

- * iValue 라는 변수 에 2를 대입한다
- > 한번 대입(할당) 한 변수의 값은 다른 값을 대입하기 전까지 로직을 구동하는 동안 계속 반복해서 사용할수 있다.
 - * 변수는 반드시 초기화 (초기값을 대입) 하여 생성 해야 한다. 만약 초기화 되어있지 않은 상태 의 변수를 호출시 오류가 발생한다.

```
ChapO1_intro > ♠ 03_Operator[연산자].py > ...

1  print(iValue)

2  iValue = 3

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL

File "d:\Python\ChapO1_intro\03_Operator[연산자].py", line 1, in <module> print(iValue)

NameError: name 'iValue' is not defined
```

* iValue 를 찾을 수 없습니다.

- 산술 연산자
- . 더하기 빼기 곱하기 나누기 연산을 수행한다. (사칙연산자)

```
Chap01_intro > 🕏 03_Operator[연산자].py
      print(1 + 2)
                     # 더하기 연산자
      print(2 - 1)
                      # 빼기 연산자
                      # 곱하기 연산자
      print(3 * 3)
                      # 나누기 연산자 (결과 를 소수로 표현)
      print(5 / 2)
      print(5 // 2) # 나누기 연산자 (몫을 표현)
                       # 나누기 연산자 (나머지를 표현)
      print(5 % 2)
                         TERMINAL
Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.
D:\Python>C:/Users/MasterD/AppData/Local/Programs/Python/Python311/python.exe d:/Python/
9
```

- 복합 대입 연산자

. 기존의 값에 새로운 값을 누적하여 연산 하고자 할때 사용한다

```
Chap01_intro > © 03_Operator[연산자].py > ...

1  Value = 100

2  Value = Value + 10

3  print(Value) #110

4  5  Value += 10 # 기존의 값에 10을 더한값을 대입한다

6  print(Value) #120

7

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

D:\Python>C:/Users/MasterD/AppData/Local/Programs/Python/Python311/python.exe d:/P110
120
```

. 복합 대입 연산자 의 예시

```
Chap01_intro > 🕏 03_Operator[연산자].py > ...
      Value = 100
   2
      Value += 10 # 기존의 값에 10을 더한값을 대입한다
      print(Value) # 110
      Value -= 10 # 기존의 값에 10을 차감한값을 대입한다
      print(Value) # 100
      Value *= 10 # 기존의 값에 10을 곱한값을 대입한다
      print(Value) # 1000
      Value /= 3 # 기존의 값을 3으로 나눈 결과를 대입한다
  12
      print(Value) # 333.333333...
      Value //= 10 # 기존의 값에 10을 나눈 몫을 대입한다
      print(Value) # 33
      Value %= 10 # 기존의 값에 10을 나눈 나머지를 대입한다
      print(Value) # 3
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
D:\Python>C:/Users/MasterD/AppData/Local/Programs/Python/Python311/python.exe d:/Python/Cha
110
100
1000
333.33333333333333
33.0
3.0
```

타입 변화

- 문자열의 연산
- . 문자열 간의 +
- > 두 문자열을 나열하여 표현한다.

```
ChapO1_intro > ● 03_Operator[연산자].py

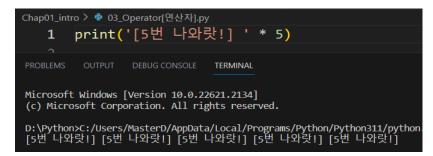
1 print('안녕하세요?' + "정말쉬운 파이썬입니다.")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

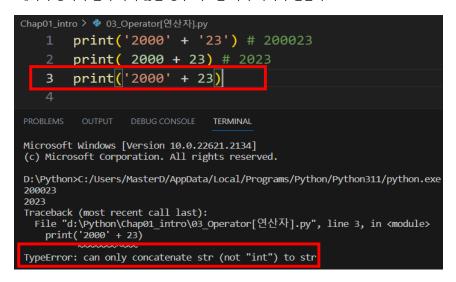
Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

D:\Python>C:/Users/MasterD/AppData/Local/Programs/Python/Python311/python.exe C안녕하세요?정말쉬운 파이썬입니다.
```

- . 문자열의 에 곱하기 연산자 *
- > 문자 열이 반복적으로 나열될 횟수를 지정 할 수 있다.



- . 문자열 과 정수 의 더하기 연산자
- > 데이터 형식이 일치 하지 않을 경우 의도를 파악 하기가 힘들다.



> 표현하고자 하는 결과의 데이터 형식을 맞추어 주어야 한다.

```
print('2000' + str(23)) # 23 을 문자로 바꿔서 붙여서 표현
print(int('2000') + 23) # 2000 을 숫자로 바꾸고 23을 더하여 표현
134

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Microsoft Windows [Version 10.0.22621.2283]
(c) Microsoft Corporation. All rights reserved.

D:\MyDoc\11.EDUCATION\1. Master Source\15.Python\0.MyFirstPython\0.Source\Python> cmd /c "C:\Users\Master scode\extensions\ms-python.python-2023.16.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher hon\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.Source\Python\0.S
```

. 숫자로 표현 가능한 진법의 수는 해당 진수를 같이 표현 함으로써 10진수 표현을 할 수 있다.

```
ChapO1_intro > * 03_Operator[연산자].py

1 print(int("1a", 16)) # 16 진수로 변경 가능한 문자열 "1a" 를 10진수로 표현
2 print(int("15", 8)) # 8 진수로 변경 가능한 문자열 "15" 를 10진수로 표현
3 print(int("100", 10)) # 10 진수로 변경 가능한 문자열 "100" 을 10 진수로 표현
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

D:\Python>C:/Users/MasterD/AppData/Local/Programs/Python/Python311/python.exe d:/Python/Chap01_intro/03_Operator[연산자].py 26
13
100
```

. 실수 문자열의 변환

> 실수 가 표현된 문자열을 실수 형식으로 변환 시에는 float() 함수를 사용한다.

```
ChapO1_intro > ● 03_Operator[연산자].py

1  print(float("23.2"))

2  print(float("23.2e+2"))

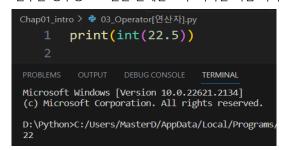
3  print(float("23.2e-2"))

4  PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

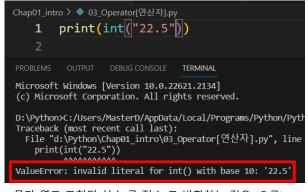
Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

D:\Python>C:/Users/MasterD/AppData/Local/Programs/Py 23.2
2320.0
0.232
```

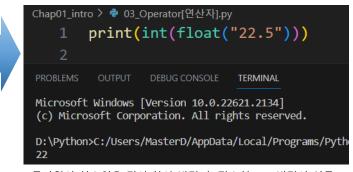
> 실수를 정수형으로 변환 할때는 소수 이하는 버림 처리 되며 문자열 로 표현된 실수는 곧바로 정수형으로 바꿀 수 는 없다.



< 실수 를 정수 로 나타 낸 경우 >



< 문자 열로 표현된 실수 를 정수 로 변환하는 경우 (오류) >



< 문자열이 실수임을 명시 하여 변경 후 정수형으로 변경시 성공 >

> 소수 자르기 Round

. Round() 는 실수 를 지정한 위치에서 반올림 하여 표현한다.

```
149 print(round(12.834)) # 소수 1번째 위치에서 반올림
150 print(round(12.834,1)) # 소수 2번째 위치에서 반올림
151 print(round(11555.12,-3)) # 정수 3번째 위치에서 반올림
152

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

12000
d:\MyDoc\11.EDUCATION\1. Master Source\15.Python\0.MyFirstPython\0.Source\Python> d: && cd "d:\M cmd /C "C:\Users\MasterD\AppData\Local\Programs\Python\Python311\python.exe c:\Users\MasterD\.vs /../..\debugpy\launcher 64195 -- "D:\MyDoc\11.EDUCATION\1. Master Source\15.Python\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFirstPython\0.MyFir
```