

[Python]



# Python으로 배우는 소프트웨어 원리

## Chapter 03. 변수

# 목차

1. 정보의 표현
2. 변수의 개념과 활용

# 01

## 정보의 표현

# 01. 정보의 표현

## [컴퓨터의 데이터 처리 과정]

- 컴퓨터는 프로그램을 사용하여 다양한 종류의 데이터를 처리하는 장치이다.
- 개발자는 프로그래밍 언어를 사용하여 데이터를 처리하는 과정을 구성한다.

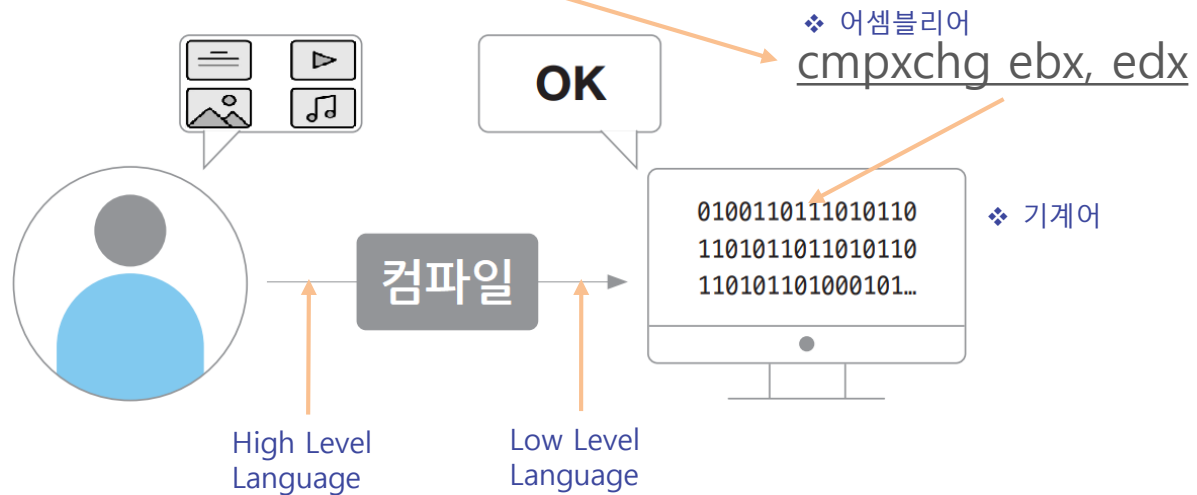
Problem → **Algorithm** → Coding → Program

- 다양한 언어로 작성된 **프로그램**은 컴퓨터가 이해 할 수 있는 기계어(2진 형태로 구성)로 변환시켜야 한다. → **컴파일(Compile)**
- 프로그램이 처리하는 **데이터**들도 컴퓨터가 이해할 수 있는 2진 형태로 변환이 되어 있어야 한다. → 인간이 구분할 수 없을 정도의 정밀도를 유지한 아날로그 신호 단위를 각각 비트 열로 구성된 2진 형태로 변환
- 프로그램은 문제 해결 절차를 **명령어**를 사용하여 구성한다.
- 프로그램 명령어는 다양한 데이터를 수용하기 위해 데이터를 대신할 수 있는 **변수**를 사용한다.
- 프로그램이 실행되는 과정에서 인간과 소통하기 위해 2진 형태의 데이터를 10진 또는 멀티미디어(그림, 영상, 소리 등) 형태로 **변환**하여 표현한다. → 2진 비트열 단위를 아날로그로 변환

# [용어] 컴파일

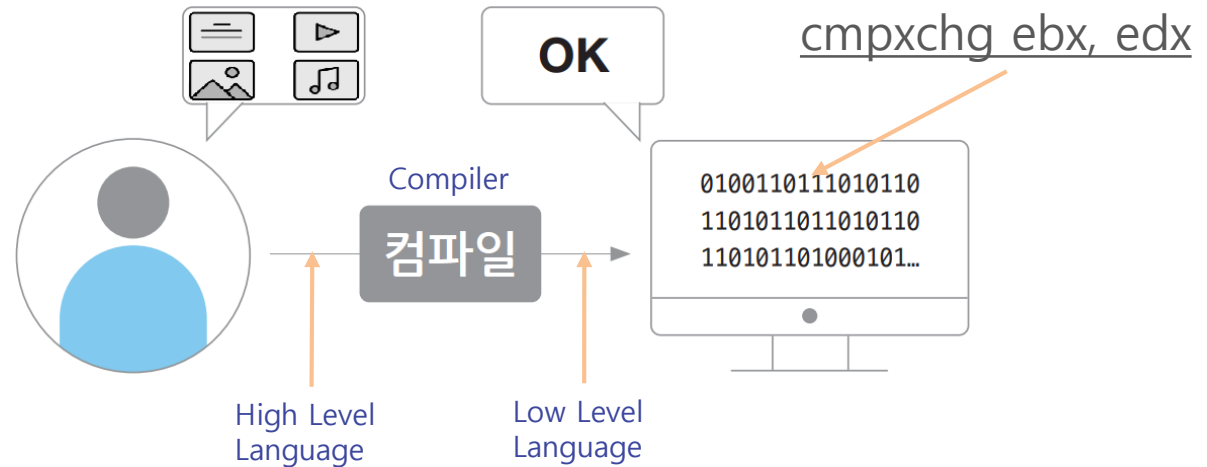
## [컴파일(compile)의 역할]

- 어떤 언어의 코드를 다른 언어로 바꿔주는 과정
- 인간 중심의 고급 언어를 컴퓨터가 처리하기 용이한 기계어로 번역하는 프로그램
- ❖ 인간은 자연어에 가까운 High Level Language로 프로그램 작성
- ❖ 컴퓨터는 비트(bit)로 구성된 워드(word) 형식의 기계 명령어 중심으로 처리



- ❖ 어셈블리어(Assembly Language)는 기계어 명령어를 인간이 이해하기 쉬운 문자 형태로 표현하는 언어
- ❖ 기계어(Machine Language)는 실제 컴퓨터가 직접 해석해서 실행 가능하도록 구성된 비트 형태의 언어

# [용어] 컴파일



C 언어 코드	어셈블리 코드(x86)
<pre>int CAS(int* pos, int oldval, int newval) {     int oldpos = *pos;     if(*pos == oldval)         * pos = newval;     return oldpos; }  int a, b; b = CAS(&amp;a, 10, 20);</pre>	<pre>CAS:     mov ecx, dword ptr[esp + 4]     mov eax, dword ptr[ecx]     mov ebx, dword ptr[esp + 8];     mov edx, dword ptr[esp + 12];     cmpxchg ebx, edx;     mov dword ptr[ecx], esp;     ret 16;</pre>

# 01. 정보의 표현

## I. 컴퓨터의 데이터 처리

### ■ 데이터와 정보

- **데이터(Data)**란 어떤 의미나 목적을 포함하지 않은 수집되거나 측정된 값 혹은 자료를 의미
- **정보(Information)**는 이런 데이터를 의도나 목적에 맞게 분석 혹은 가공하여 그 의미를 표현
- 컴퓨터는 정보처리장치이며, 데이터를 사람이 이용할 수 있는 정보로 제공하는 장치

# 01. 정보의 표현

## I. 컴퓨터의 데이터 처리

### ■ 이진 데이터의 표현

- 컴퓨터에서 데이터를 처리하는 기본 단위는 비트이며, 0과 1의 이진값을 표현
- 컴퓨터는 필요한 모든 데이터를 이진수로 처리하며, 비트는 스위치나 전구의 off나 on 처럼 0이나 1이라는 2개의 값만 저장
- 여러 개의 비트들을 비트열이라고 부르며, 비트열이 모이면 아주 많은 값을 표현할 수 있음

표 3-1 비트열의 개수에 따라 표현 가능한 값의 범주

비트 개수	표현 가능한 값의 개수	비트와 값의 패턴
1	2	$2^1$
2	4	$2^2$
3	8	$2^3$
4	16	$2^4$
8	256	$2^8$
16	65,536	$2^{16}$
n	$2^n$	$2^n$



# 01. 정보의 표현

## I. 컴퓨터의 데이터 처리

### ■ 이진 데이터의 효율적인 표현

- 이진 데이터를 4-비트 또는 3비트씩 잘라 16진수나 8진수로 표현하면 짧게 표현할 수 있다.

➤  $0b00110110 = 0x36 = 0o066$

10진수	16진수	8진수	2진수
0	0	0	0000
1	1	1	0001
2	2	2	0010
3	3	3	0011
4	4	4	0100
5	5	5	0101
6	6	6	0110
7	7	7	0111
8	8	10	1000
9	9	11	1001
10	a	12	1010
11	b	13	1011
12	c	14	1100
13	d	15	1101
14	e	16	1110
15	f	17	1111

# 01. 정보의 표현

## I. 컴퓨터의 데이터 처리

### ■ 이진 데이터의 표현

#### 여기서 잠깐 데이터 단위

표 3-2 데이터 단위

이름	비트 개수	값의 개수	저장 항목의 예
비트(bit)	1	$2^1 = 2$	예/아니오
바이트(byte)	8	$2^8 = 256$	알파벳 하나
킬로(kilo)	10	$2^{10} = \text{약 } 10^3$	문단 몇 개
메가(mega)	20	$2^{20} = \text{약 } 10^6$	1분 길이 노래
기가(giga)	30	$2^{30} = \text{약 } 10^9$	256개 노래
테라(tera)	40	$2^{40} = \text{약 } 10^{12}$	영화 1300편
페타(peta)	50	$2^{50} = \text{약 } 10^{15}$	국내 인구 1/2의 주민정보
엑사(exa)	60	$2^{60} = \text{약 } 10^{18}$	2018년 드롭박스 데이터 용량
제타(zetta)	70	$2^{70} = \text{약 } 10^{21}$	세계 인구 1인당 36t 분량의 책

# 01. 정보의 표현

## II. 다양한 정보의 표현

### ■ 수: 정수, 실수

- 2진수로 저장
- 데이터 크기 및 정밀도에 따라 사용 크기 결정
- 데이터 형식에 따라 저장 형식(Format)이 다름
- **정수**(integer)는 부호와 크기로 표현
- **실수**(float)는 부호와 가수 및 지수의 크기로 표현

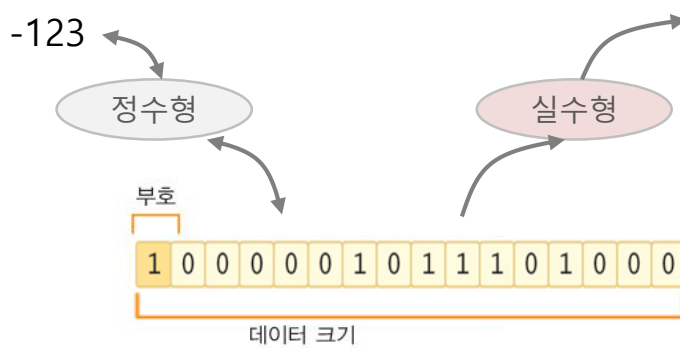


그림 3-7 • 양의 정수 표현방식

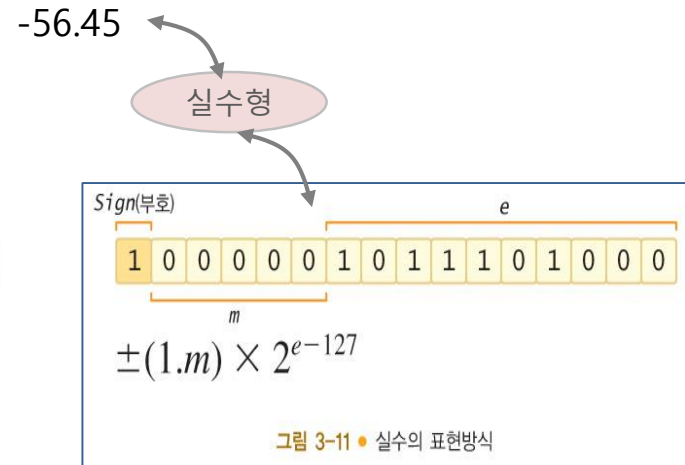


그림 3-11 • 실수의 표현방식

# 01. 정보의 표현

## II. 다양한 정보의 표현

### ■ 4bit 컴퓨터 세계에서 CPU 사칙연산 방식?

- 왼쪽 최상위 비트(LSB)를 부호 비트로, 나머지는 수의 크기 비트로 사용
- 음수는 양수에 2의보수를 취해서 사용
- 사칙연산은 덧셈 만으로 해결

#### ❖ 정수 연산

$$\begin{aligned} (+1) + (-1) &= (0001)_2 + (1111)_2 = (0000)_2 = (0) \\ (+1) + (-2) &= (0001)_2 + (1110)_2 = (1111)_2 = (-1) \\ (+1) - (2) &= (+1) + (-2) \end{aligned}$$

binary	Decimal	binary	Decimal
0000	+0	0000	-0
0001	+1	1111	-1
0010	+2	1110	-2
0011	+3	1101	-3
0100	+4	1010	-4
0101	+5	1011	-5
0110	+6	1010	-6
0111	+7	1001	-7

#### ❖ 2진수 정수값을 10진수로 표현하는 방법

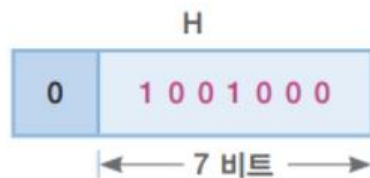
- 부호 판단 : Ex>  $(1111)_2$  : sign : -
  - > 만약  $LSB == 0$  이면: +부호 사용
  - > 만약  $LSB == 1$  이면: -부호 사용
- 크기 판단 : Ex>  $(1111)_2$ 의 2의 보수 값 =  $(0001)_2 = (1)$ 
  - > 만약  $LSB == 0$  이면: 전체 비트를 10진수로 변환하여 사용
  - > 만약  $LSB == 1$  이면: 전체 비트를 2의 보수를 취하여, 10진수로 변환하여 사용

# 01. 정보의 표현

## II. 다양한 정보의 표현

### ■ 텍스트

- 문자 코드표는 모든 컴퓨터에서 동일하게 유지되도록 해야 통신이나 파일 공유 작업이 가능하므로 **아스키(ASCII)**라고 하는 코드 표준을 사용
- 아스키는 전체 128개 문자에 대해 각 문자당 8개 비트(첫 번째 비트 0은 다른 용도로 사용)를 사용해 다양한 값을 표현



아스키 코드표  
일부

Binary	Char	Binary	Char
00100000	space	01000001	A
00100001	!	01000010	B
00100010	"	01000011	C
00100011	#	01000100	D
00100100	\$	01000101	E
00100101	%	01000110	F
00100110	&	01000111	G
00100111	'	01001000	H
00101000	(	01001001	I

그림 3-4 아스키 코드표

# [용어] 아스키 코드

- \*.txt와 같은 텍스트 파일은 각 문자 단위로 ASCII코드로 기록되어 저장되고, 메모장 같은 도구로 변환하여 문자로 보는 것이다.  
따라서 메모장으로 열어서 인식이 되는 파일은 문자 형태의 파일인 것이다.

## [아스키(ASCII) 코드]

- ASCII(American Standard Code for Information Interchange). 미국 정보 교환 표준 부호
- 1963년 미국 ANSI에서 표준화한 정보 교환용 7비트 부호 체계
- 오늘날 컴퓨터에서도 활용되어 각 **문자**를 표현하는 8비트 이진 코드로 사용

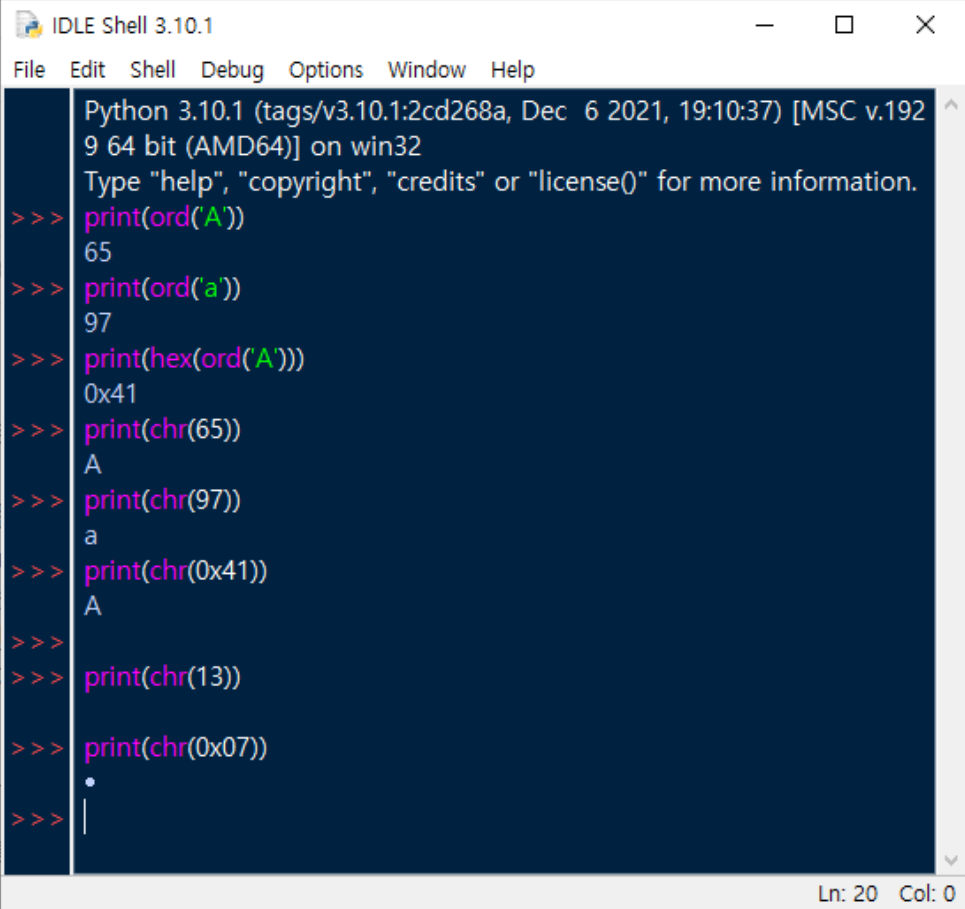
ASCII Table					$0_0_0$	$0_0_1$	$0_1_0$	$0_1_1$	$1_0_0$	$1_0_1$	$1_1_0$	$1_1_1$
b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	상위 3비트		0	1	2	3	4	5	6	7
			b <sub>4</sub>	b <sub>3</sub>								
0	0	0	0	0	NUL <sub>0</sub>	DLE <sub>16</sub>	SP <sub>32</sub>	0 <sub>48</sub>	@ <sub>64</sub>	P <sub>80</sub>	` <sub>96</sub>	p <sub>112</sub>
0	0	0	1	1	SOH <sub>1</sub>	DC1 <sub>17</sub>	! <sub>33</sub>	1 <sub>49</sub>	A <sub>65</sub>	Q <sub>81</sub>	a <sub>97</sub>	q <sub>113</sub>
0	0	1	0	2	STX <sub>2</sub>	DC2 <sub>18</sub>	" <sub>34</sub>	2 <sub>50</sub>	B <sub>66</sub>	R <sub>82</sub>	b <sub>98</sub>	r <sub>114</sub>
0	0	1	1	3	ETX <sub>3</sub>	DC3 <sub>19</sub>	# <sub>35</sub>	3 <sub>51</sub>	C <sub>67</sub>	S <sub>83</sub>	c <sub>99</sub>	s <sub>115</sub>
0	1	0	0	4	EOT <sub>4</sub>	DC4 <sub>20</sub>	\$ <sub>36</sub>	4 <sub>52</sub>	D <sub>68</sub>	T <sub>84</sub>	d <sub>100</sub>	t <sub>116</sub>
0	1	0	1	5	ENQ <sub>5</sub>	NAK <sub>21</sub>	% <sub>37</sub>	5 <sub>53</sub>	E <sub>69</sub>	U <sub>85</sub>	e <sub>101</sub>	u <sub>117</sub>
0	1	1	0	6	ACK <sub>6</sub>	SYN <sub>22</sub>	& <sub>38</sub>	6 <sub>54</sub>	F <sub>70</sub>	V <sub>86</sub>	f <sub>102</sub>	v <sub>118</sub>
0	1	1	1	7	BEL <sub>7</sub>	ETB <sub>23</sub>	' <sub>39</sub>	7 <sub>55</sub>	G <sub>71</sub>	W <sub>87</sub>	g <sub>103</sub>	w <sub>119</sub>
1	0	0	0	8	BS <sub>8</sub>	CAN <sub>24</sub>	( <sub>40</sub>	8 <sub>56</sub>	H <sub>72</sub>	X <sub>88</sub>	h <sub>104</sub>	x <sub>120</sub>
1	0	0	1	9	HT <sub>9</sub>	EM <sub>25</sub>	) <sub>41</sub>	9 <sub>57</sub>	I <sub>73</sub>	Y <sub>89</sub>	i <sub>105</sub>	y <sub>121</sub>
1	0	1	0	A	LF <sub>10</sub>	SUB <sub>26</sub>	* <sub>42</sub>	: <sub>58</sub>	J <sub>74</sub>	Z <sub>90</sub>	j <sub>106</sub>	z <sub>122</sub>
1	0	1	1	B	VT <sub>11</sub>	ESC <sub>27</sub>	+ <sub>43</sub>	; <sub>59</sub>	K <sub>75</sub>	[ <sub>91</sub>	k <sub>107</sub>	{ <sub>123</sub>
1	1	0	0	C	FF <sub>12</sub>	FS <sub>28</sub>	, <sub>44</sub>	< <sub>60</sub>	L <sub>76</sub>	\ <sub>92</sub>	l <sub>108</sub>	<sub>124</sub>
1	1	0	1	D	CR <sub>13</sub>	GS <sub>29</sub>	- <sub>45</sub>	= <sub>61</sub>	M <sub>77</sub>	] <sub>93</sub>	m <sub>109</sub>	} <sub>125</sub>
1	1	1	0	E	SO <sub>14</sub>	RS <sub>30</sub>	. <sub>46</sub>	> <sub>62</sub>	N <sub>78</sub>	^ <sub>94</sub>	n <sub>110</sub>	~ <sub>126</sub>
1	1	1	1	F	SI <sub>15</sub>	US <sub>31</sub>	/ <sub>47</sub>	? <sub>63</sub>	O <sub>79</sub>	_ <sub>95</sub>	o <sub>111</sub>	DEL <sub>127</sub>

# [용어] 아스키 코드

## [Python: ASCII 코드 관련 함수]

- **ord()** 함수 : 특정한 문자에 대한 아스키 코드 값을 반환 (10진 정수 값으로 반환)
- **chr()** 함수 : 아스키 코드 값을 문자로 변환

- `print(ord('A'))`
- `print(ord('a'))`
- `print(hex(ord('A')))`
- `print(chr(65))`
- `print(chr(97))`
- `print(chr(0x41))`
- `print("%c %d " % (65, 65))`
- `print(chr(13))`
- `print(chr(0x07))`



```
Python 3.10.1 (tags/v3.10.1:2cd268a, Dec 6 2021, 19:10:37) [MSC v.192
9 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print(ord('A'))
65
>>> print(ord('a'))
97
>>> print(hex(ord('A')))
0x41
>>> print(chr(65))
A
>>> print(chr(97))
a
>>> print(chr(0x41))
A
>>>
>>> print(chr(13))
.
>>> print(chr(0x07))
|
>>>
```

# 01. 정보의 표현

## II. 다양한 정보의 표현

### ■ 이미지

- 컴퓨터에서 이미지를 처리하는 방식은 연속적인 점을 찍어 그림을 그리듯 아주 작은 격자를 사용해 표현, 이것을 **픽셀(Pixel)** 또는 화소라고 함
- 해상도는 동일한 면적에 표시할 수 있는 픽셀 수를 의미
- 해상도가 높을수록 더 또렷하게 볼 수 있고 이러한 해상도를 측정하는 단위로 PPI(Pixels per Inch)를 사용

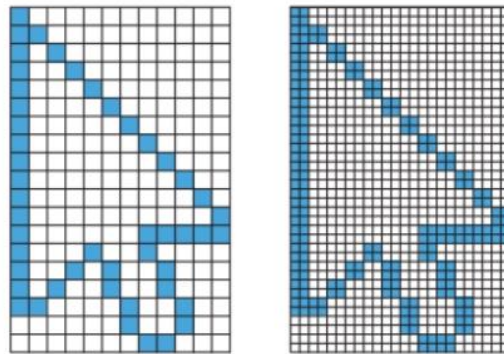


그림 3-6 픽셀 수가 다른 동일 이미지



# [용어] 픽셀

## [픽셀(Pixel)]

- '픽처 엘리먼트(Picture Element)'의 준말 → 화소(畵素)
- 이미지를 디지털로 변환하기 위해 사각형으로 분해한 색 표현의 최소 단위
- 픽셀은 색에 대한 정보를 빛의 삼원 색인 (red, green, blue)의 혼합 정도로 표현 가능
  - 각 색의 크기 범위는 0~255(0x00~0xff)
  - (r, g, b) => black (0, 0, 0), white (255, 255, 255)



## [용어] 픽셀

### [Python: 픽셀(Pixel) 값 적용]

- 픽셀은 색에 대한 정보를 빛의 삼원 색인 (red, green, blue)의 혼합 정도로 표현
  - 각 색의 크기 범위는 0~255(0x00~0xff)
  - (r, g, b) => black (0, 0, 0), white (255, 255, 255)
- Turtle Demo 예제 실행
  - IDLE 쉘에서 [Help] → [Turtle Demo] 실행
  - Turtle Demo 창에서 [Examples] → Colormixer를 선택, 실행



Ch03-TurtleColor.py

# [용어] 픽셀

## [Python: 픽셀(Pixel) 값 적용]

- 픽셀은 색에 대한 정보를 빛의 삼원 색인 (red, green, blue)의 혼합 정도로 표현
  - 각 색의 크기 범위는 0~255(0x00~0xff)
  - (r, g, b) => black (0, 0, 0), white (255, 255, 255)

- 글자색 변경은 'W033[38;2;r;g;bm'
- 배경색 변경은 'W033[48;2;r;g;bm'
- 원상태 리셋은 'W033[0m'

```
1 print("Hello~")
2 ## True color (RGB) 색 지정
3 print('\033[38;2;255;0;0m' + '빨간 색' + '\033[0m')
4 print('\033[38;2;0;255;0m' + '초록 색' + '\033[0m')
5 print('\033[38;2;0;0;255m' + '파란 색' + '\033[0m')
6
7 print('\033[38;2;255;0;0m' + '\033[48;2;0;255;0m' + "Welcome!" + '\033[0m')
8
```

Run: Test

```
Hello~
빨간 색
초록 색
파란 색
Welcome!

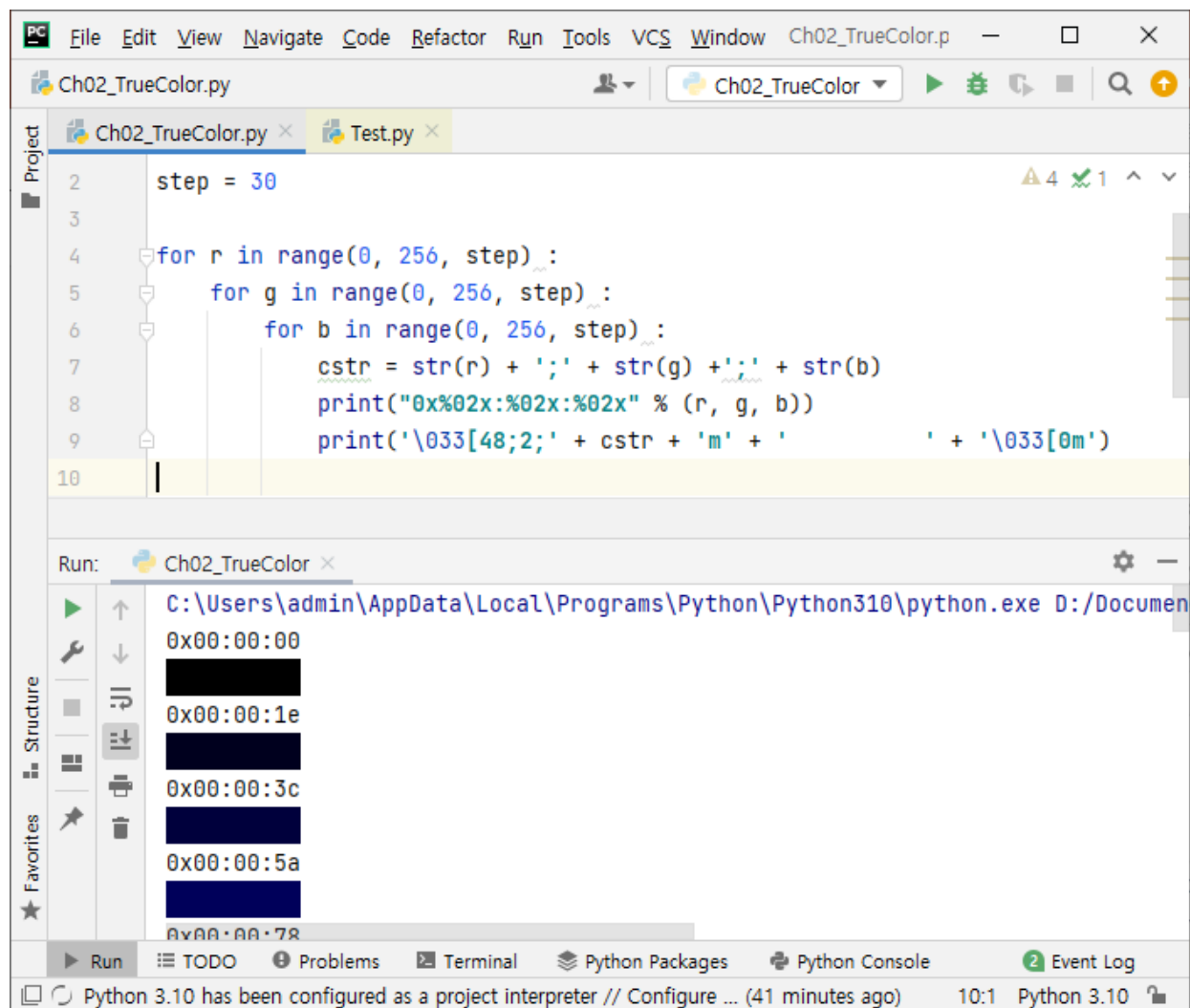
Process finished with exit code 0
```

Python 3.10 has been configured as a project interpreter // Configure a ... (today 오전 11:17) 1:1 Python 3.10

# [용어] 픽셀

## [Python: 픽셀(Pixel) 값 혼합 확인]

Ch03\_TrueColor.py



The screenshot shows an IDE window titled 'Ch02\_TrueColor.p' with a menu bar (File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window) and a toolbar. The editor displays a Python script in 'Ch02\_TrueColor.py' with the following code:

```
2 step = 30
3
4 for r in range(0, 256, step):
5     for g in range(0, 256, step):
6         for b in range(0, 256, step):
7             cstr = str(r) + ';' + str(g) + ';' + str(b)
8             print("0x%02x:%02x:%02x" % (r, g, b))
9             print('\033[48;2;' + cstr + 'm' + ' ' + '\033[0m')
```

The Run console shows the output of the script, displaying hexadecimal color values and their corresponding visual representations as colored squares:

```
Run: Ch02_TrueColor x
C:\Users\admin\AppData\Local\Programs\Python\Python310\python.exe D:/Documen
0x00:00:00
0x00:00:1e
0x00:00:3c
0x00:00:5a
0x00:00:78
```

The status bar at the bottom indicates 'Python 3.10 has been configured as a project interpreter // Configure ... (41 minutes ago)' and '10:1 Python 3.10'.

# 01. 정보의 표현

## II. 다양한 정보의 표현

### ■ 사운드

- 컴퓨터의 입력장치인 마이크로 수집되는 사운드는 디지털 신호로 변환하는 과정인 ADC(Analog-to-Digital Converter)을 진행
  - 음의 크기 및 음질 등에 대한 정보를 시간 차로 샘플링하여 그 정도를 2진 값으로 변환
- 컴퓨터에 저장된 사운드를 재생하려면 역으로 변환하는 과정인 DAC(Digital-to-Analog Converter)와 스피커 등의 출력장치를 통해 원래의 아날로그 신호 형태로 전달

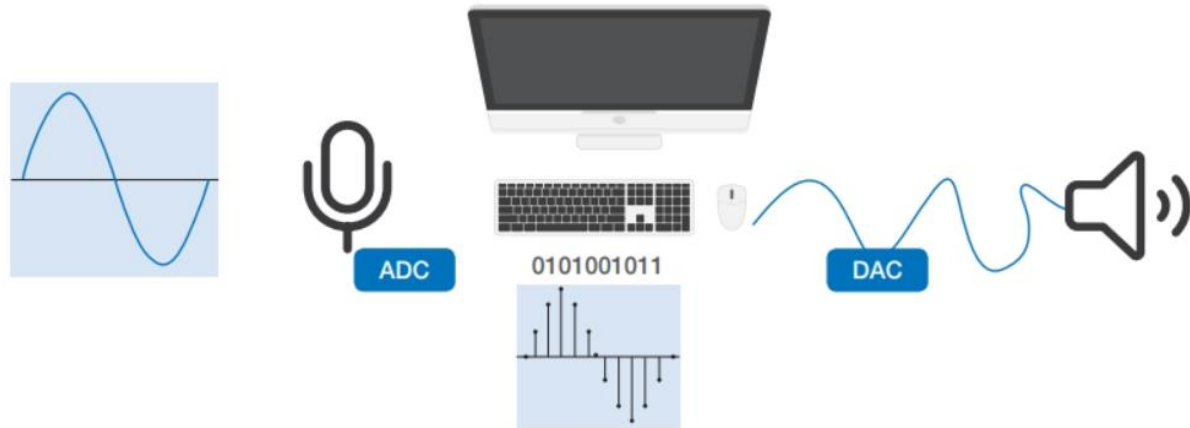


그림 3-9 사운드의 변환 과정

02

## 변수의 개념과 활용

## 02. 변수의 개념과 활용

### I. 변수의 개념과 사용

- 메모리 주소와 변수명
  - **변수(Variable)**는 프로그램이 실행되는 동안 필요한 데이터를 잠시 저장하는 공간
  - 변수의 이름(변수명)은 결국 메모리의 특정 위치에 이름을 붙이는 것

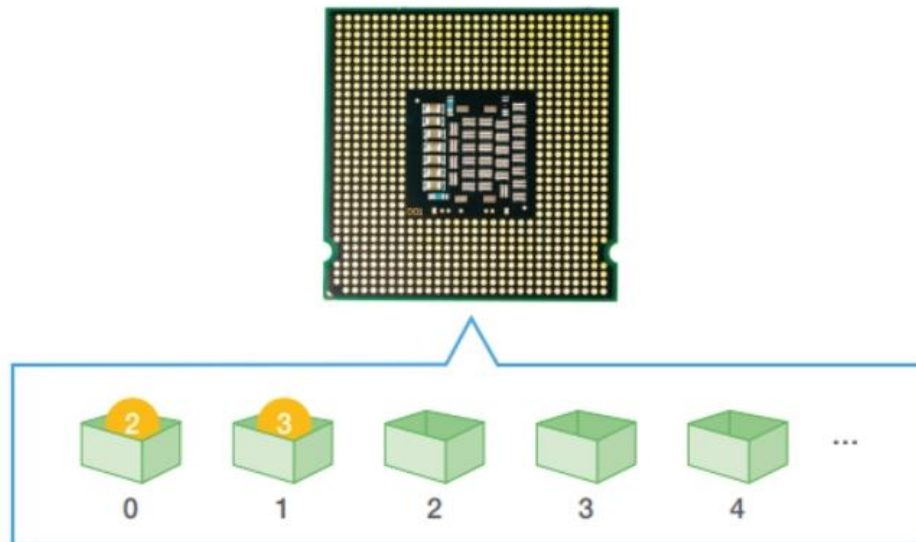


그림 3-11 물건을 담은 상자와 같이 값을 저장하는 변수

## 02. 변수의 개념과 활용

### I. 변수의 개념과 사용

- 변수 생성과 값 저장 (Write)

- 파이썬에서는 변수명을 대입 연산자(=)를 왼쪽에 명시하여 변수명에 해당하는 메모리를 만들게 하고, 오른쪽 값을 저장

```
>>> 변수명 = 저장할 값
>>> sum = 3
```

오른쪽 값을 왼쪽 변수에 저장(대입)

- 변수 값 사용 (Read)

- 대입 연산자(=) 오른쪽에 명시하거나 단독으로 사용하면, 변수명에 해당하는 메모리에 저장된 값을 읽어서 변수명 대신해 치환한다.

➤ a = 3	#write(=)
➤ b = 4	#write(=)
➤ sum = a + b	#read(a) > read(b) > + > write(sum)
➤ print("%d + %d = %d" %(a, b, sum))	#read(a) > read(b) > read(sum) > 배치 > 함수호출



## 02. 변수의 개념과 활용

### I. 변수의 개념과 사용

- 변수 생성과 값 저장

#### 실습 3-1

변수를 만들고 값 저장하기

①

```
>>> x = 3
>>> y = 5
>>> z = "안녕"
```

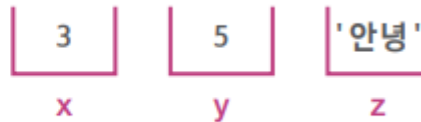


그림 3-13 변수 생성과 값 저장

②

```
>>> x
3
>>> y
5
>>> z
'안녕'
```

```
>>> sum = x + y
>>> print(sum)
8
```

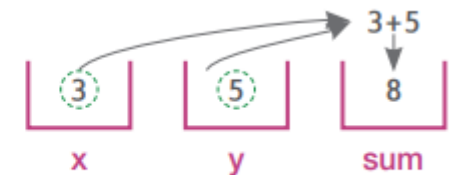


그림 3-14 변수 값 저장 시, 수식 사용

## 02. 변수의 개념과 활용

### I. 변수의 개념과 사용

- 변수명 생성 시 주의 사항
  - 문자(A-z)와 숫자(0-9), \_(underscore)를 사용할 수 있음
  - '2x'처럼 변수명이 숫자로 시작하면 안 됨
  - 대소문자를 구분하므로, 'data'와 'Data'는 다른 변수로 취급
  - '합계'와 같은 한글 변수명도 사용할 수 있음
  - 파이썬의 키워드(keyword)인 True, if, for 등은 사용할 수 없음

## 02. 변수의 개념과 활용

### I. 변수의 개념과 사용

- 변수 생성과 값 저장

실습 3-2

파이썬의 키워드 확인하기

- ① 예약어를 가지고 있고 이를 변수 이름으로 사용할 수 없음

```
>>> import keyword
>>> print(keyword.kwlist)
['False', 'None', 'True', '__peg_parser__', 'and', 'as', 'assert', 'async', 'await',
'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']
```

## 02. 변수의 개념과 활용

### II. 변수 값의 변경과 입력

#### 실습 3-3

#### 변수 값을 변경하고 입력받기

- ② 변수 number와 name에 각각 숫자와 문자열을 저장하고 출력

```
>>> number = 5
>>> name = "Kim"
>>> print(number, name)
5 Kim
```



그림 3-15 변수 생성과 값 저장

- ③ 대입 연산자 우측에 바꾸려는 값이나 수식을 입력하면 변수의 값이 변경

```
>>> number = 3
>>> name = "Kim" * number
>>> print(number, name)
3 KimKimKim
```



그림 3-16 변수 값 변경

- ❖ 문자열에서 '+'와 '\*' 연산자는 각각 문자열의 '연결'과 '반복' 의미로 사용

## 02. 변수의 개념과 활용

### II. 변수 값의 변경과 입력

#### 실습 3-3

#### 변수 값을 변경하고 입력받기

- ③ input() 함수를 이용해 name 변수 값을 키보드로 직접 입력받아 바꾸고 출력

```
>>> name = input("이름 입력 : ")
```

```
이름 입력 : 홍길동
```

```
>>> print(name)
```

```
홍길동
```

input() 함수의 괄호() 안에 적은  
메시지가 입력할 때 표시됨

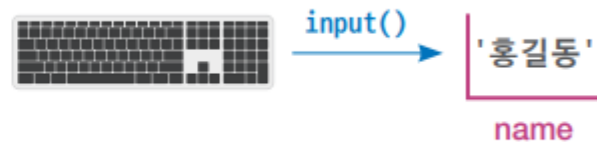


그림 3-17 변수 값 입력받기

- ④ number 변수에는 input() 함수로 입력한 정수를 저장하고, 수식을 만들어 출력

```
>>> number = int(input("번호 입력 : "))
```

```
번호 입력 : 5
```

```
>>> print(name * number)
```

```
홍길동홍길동홍길동홍길동홍길동
```

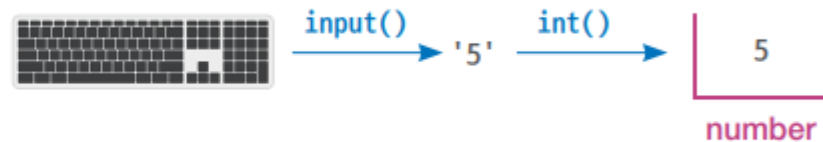


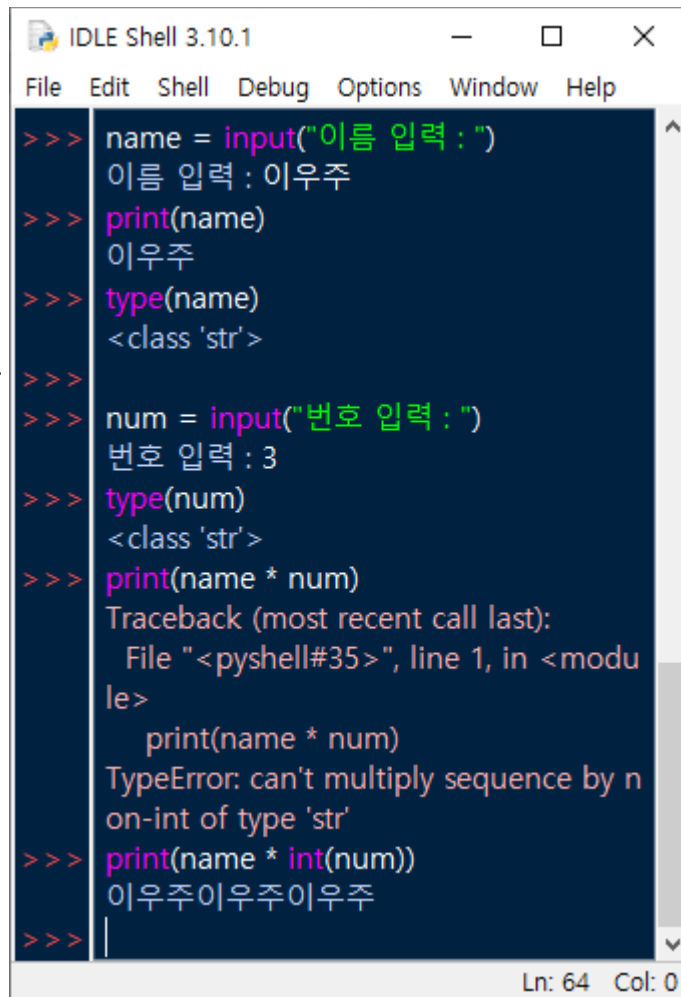
그림 3-18 정수 입력과 수식의 결과 출력

## 02. 변수의 개념과 활용

### II. 변수 값의 변경과 입력

#### [Python실습]

- 키보드로 입력되는 값은 문자열로 인식한다.
- int()는 대상 값을 정수형으로 형변환을 해준다.
  - 형변환을 하면 저장하는 방식이 원하는 형으로 바뀜
  - 저장된 형식과 동일한 형식으로 사용해야 문제가 없음



```
IDLE Shell 3.10.1
File Edit Shell Debug Options Window Help
>>> name = input("이름 입력 : ")
이름 입력 : 이우주
>>> print(name)
이우주
>>> type(name)
<class 'str'>
>>>
>>> num = input("번호 입력 : ")
번호 입력 : 3
>>> type(num)
<class 'str'>
>>> print(name * num)
Traceback (most recent call last):
  File "<pyshell#35>", line 1, in <module>
    print(name * num)
TypeError: can't multiply sequence by non-int of type 'str'
>>> print(name * int(num))
이우주이우주이우주
>>>
```

Ln: 64 Col: 0

## 02. 변수의 개념과 활용

### [변수] 문제

1. 다음 코드 중에서 오류가 발생하는 것을 모두 고르세요.

- ① num1 = 100
- ② 100 = num1
- ③ num1 = num2 = 100
- ④ num1 = 100 = num2 = 100
- ⑤ num1 = num2 = num3 = 100

2. 다음 각 문항에서 잘못된 변수명을 모두 고르시오.

- |             |        |        |       |       |
|-------------|--------|--------|-------|-------|
| ① TrueValue | ② ab!  | ③ aB2  | ④ 2ab | ⑤ if  |
| ⑥ _ab       | ⑦ AB_C | ⑧ ab 2 | ⑨ 에이비 | ⑩ if2 |

## 02. 변수의 개념과 활용

### [변수] 문제

3. 다음의 아래 코드 이후에 실행될 코드 중에 오류 발생여부를 O, X로 표시하시오.

```
➤ num = input("0~9 중에 하나만 입력: ")
```

- ① `print("%d" %num)` \_\_\_\_\_
- ② `print("%c" %num)` \_\_\_\_\_
- ③ `print("%d" %int(num))` \_\_\_\_\_

4. 다음 코드의 실행 결과를 적으시오.

```
print(" " * 4, "*" * 1)
print(" " * 3, "*" * 2)
print(" " * 2, "*" * 3)
print(" " * 1, "*" * 4)
print(" " * 0, "*" * 5)
```



## 02. 변수의 개념과 활용

### [변수] 알고리즘에서의 변수 활용 (1)

- [변수의 유용성] 동일 알고리즘으로 가능한 모든 값을 처리 가능

#### [Python실습]

- I. ○ ○ ○ 란에 키보드로 입력받은 문자열이 출력되도록 코드를 수정하시오.

```
print("당신의 이름은요?")  
print("아~ ○ ○ ○ 씨군요.")  
print("○ ○ ○ 씨 반가워요.")
```

- II. 키보드로 나이를 입력받은 다음 자신의 나이와 차이를 출력하도록 코드를 수정하시오.

```
print("당신의 나이는요?")  
print("아~ ○ ○ 살 이군요.")  
print("나하고 ○ ○ 살 차이가 나네요.")
```

## 02. 변수의 개념과 활용

### [변수] 알고리즘에서의 변수 활용 (1)

[Python실습]

#### III. 키보드로 나이를 입력받은 다음 누가 나이가 많은지를 출력하는 코드 (반복 가능)

```
while True :  
    age = int(input("<나이 입력> "))  
    print("아~ %d살..." % age)  
    myage = 30  
    print("나하고 %d살 차이가 나네요." % abs(myage - age))  
  
    if myage > age :  
        print("그럼 내가 위네요.")  
    elif myage < age :  
        print("그럼 당신이 위네요.")  
    else :  
        print("그럼 동갑이네요.")  
  
    yn = input(">>그만 하려면 x를 입력? ")  
    if yn == 'x' or yn == 'X' :  
        break
```

## 02. 변수의 개념과 활용

### [변수] 알고리즘에서의 변수 활용 (2)

[Python실습] 아래 출력 형태를 알고리즘으로 해결

#### I. 반복 패턴 찾기

- ① ' ' 패턴-1: 4, 3, 2, 1, 0
- ② '\*' 패턴-2: 1, 2, 3, 4, 5

```
print(" " * 4 + "*" * 1)
print(" " * 3 + "*" * 2)
print(" " * 2 + "*" * 3)
print(" " * 1 + "*" * 4)
print(" " * 0 + "*" * 5)
```

```
      *
     **
    ***
   ****
  *****
```

#### II. 패턴 사이의 관계 찾기

- ① 패턴-1은 1씩 감소, 패턴-2는 1씩 증가
- ② 패턴-1과 패턴-2의 숫자 합은 5
- ③ 패턴은 1~5행 까지 진행
  - 패턴-2는 행의 수와 동일하게 변화
  - 패턴-1은 (5 - 행값)으로 진행

#### III. 적용할 변수 찾기

- 총 행의 수는 5
- 행의 수를 변수(i)로 사용
- 패턴-2는 i 순으로 진행
- 패턴-1은 (5 - i) 순으로 진행

## 02. 변수의 개념과 활용

### [변수] 알고리즘에서의 변수 활용 (2)

[Python실습] 아래 출력 형태를 알고리즘으로 해결

#### IV. 변수 활용

- 총 행의 수는 변수 n으로 사용
- 현재 행의 수를 변수 i로 사용
- 패턴-2는 i 순으로 진행
- 패턴-1은 (5 - i) 순으로 진행
- i-행을 n-행까지 진행

```
for i in range(1, 5+1, 1):  
    print(" " * (5-i) + "*" * i)
```

```
for 현재 값 in range(시작 값, 종료 값, 증가 값):  
    반복문
```

```
print(" " * 4 + "*" * 1)  
print(" " * 3 + "*" * 2)  
print(" " * 2 + "*" * 3)  
print(" " * 1 + "*" * 4)  
print(" " * 0 + "*" * 5)
```

```
*  
**  
***  
****  
*****
```



```
n = int(input("층 수(0~100): "))  
for i in range(1, n+1, 1):  
    print(" " * (n-i) + "*" * i)
```

## 02. 변수의 개념과 활용

### [변수] 문제

5. 아래와 같은 패턴으로 출력되도록 코드를 수정하여 보시오.

```
  *  
 ***  
*****  
*****  
*****
```

```
n = int(input("층 수(0~100): "))  
for i in range(1, n+1, 1):  
    print(" " * (n-i) + "*" * i)
```

## 02. 변수의 개념과 활용

### III. 데이터형과 형 변환

- 정수와 실수

표 3-3 데이터형의 종류

데이터형	설명	예시
int	정수(integer)	5 -1 100 -13123
float	실수(floating-point)	5.23 -0.024 2349012.412
str	문자열(string)	"python" '문자' '123'
bool	불린(boolean)	True False

- [파이썬] 변수 정의 특징

➤ 변수의 자료 저장 형식을 미리 지정하지 않아도 된다.

```
① >>> x = 85
    >>> y = 92
    >>> z = 76.5
    >>> sum = x+y+z
```

## 02. 변수의 개념과 활용

### III. 데이터형과 형 변환

#### ■ 정수와 실수

실습 3-4

점수의 합계와 평균을 계산해서 출력하기

- ① 3개의 변수에 각각 점수를 저장하고 합계를 계산

```
>>> x = 85
>>> y = 92
>>> z = 76.5
>>> sum = x+y+z
```

- ② type() 함수로 각 변수의 데이터형을 확인

```
>>> type(x)
<class 'int'>
>>> type(sum)
<class 'float'>
```

- ③ 합계를 출력하고, 평균도 계산해서 출력

```
>>> print("합계 = ", sum)
합계 = 253.5
>>> print("평균 = ", sum/3)
평균 = 84.5
```

## 02. 변수의 개념과 활용

### III. 데이터형과 형 변환

#### [Python실습]

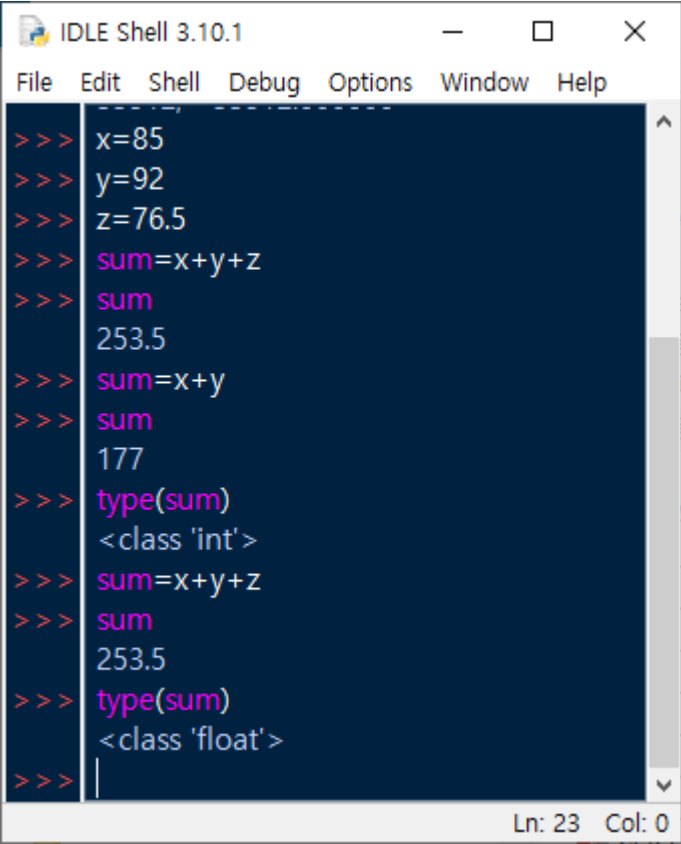
- 변수에 값을 저장하는 형식(데이터형)은 저장하려는 값의 표현 형태에 의해 정해짐
  - ❖ 만약 num 변수에 10을 저장한다면
    - >>> num = 10; 정수로 저장
    - >>> num = 10.0; 실수로 저장
    - >>> num = '10'; 문자열로 저장
- 복수의 자료형이 포함된 연산식으로 저장한다면 저장할 값에 손실이 되지 않는 자료형으로 저장
  - >>> num = 10 + 5; 정수로 저장, 결과가 **항상** 정수이므로
  - >>> num = 10 + 5.0; 실수로 저장, 연산에 **실수가 포함되어** 있으므로
  - >>> num = 10 / 5; 실수로 저장, '/' 연산 결과가 **실수가 나올 수** 있으므로



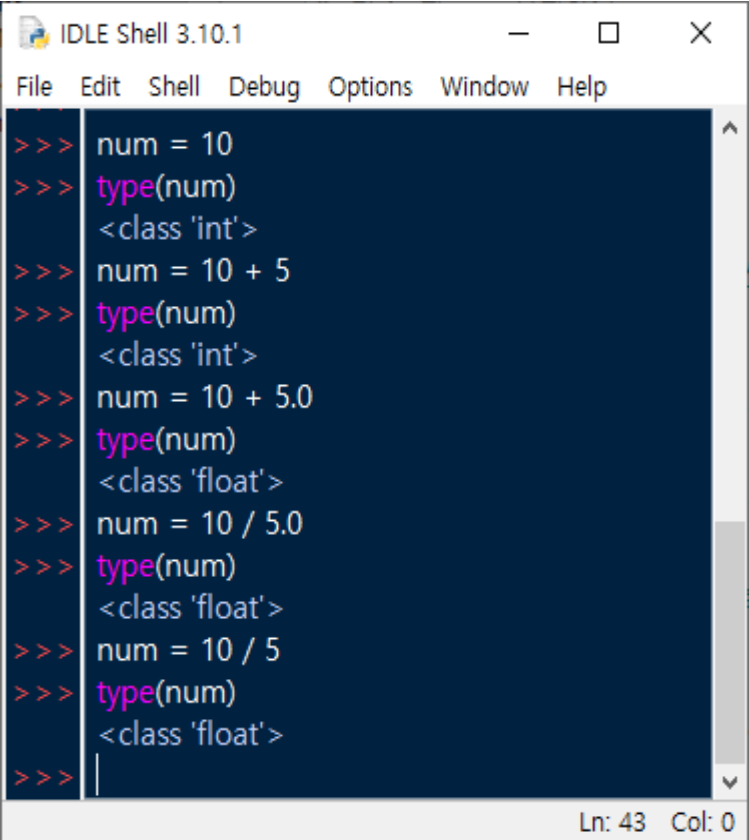
## 02. 변수의 개념과 활용

### III. 데이터형과 형 변환

#### [Python실습]

- 

```
>>> x=85
>>> y=92
>>> z=76.5
>>> sum=x+y+z
>>> sum
253.5
>>> sum=x+y
>>> sum
177
>>> type(sum)
<class 'int'>
>>> sum=x+y+z
>>> sum
253.5
>>> type(sum)
<class 'float'>
>>>
```

Ln: 23 Col: 0
- 

```
>>> num = 10
>>> type(num)
<class 'int'>
>>> num = 10 + 5
>>> type(num)
<class 'int'>
>>> num = 10 + 5.0
>>> type(num)
<class 'float'>
>>> num = 10 / 5.0
>>> type(num)
<class 'float'>
>>> num = 10 / 5
>>> type(num)
<class 'float'>
>>>
```

Ln: 43 Col: 0

## 02. 변수의 개념과 활용

### [변수] 문제

6. 아래 코드 이후에 실행될 코드의 실행 결과에 대해 데이터 형식으로 답하시오.

(int, float, str 중에 선택)

```
➤ a = 3  
➤ b = 2.0  
➤ sum = a + b
```

- ① type(a)      \_\_\_\_\_
- ② type(b)      \_\_\_\_\_
- ③ type(sum)    \_\_\_\_\_

7. 위 문항의 변수 a, b, sum에 대한 연산결과가 아래와 같이 나오도록 print() 문을 완성하시오.

```
3 + 2.0 = 5.0
```

```
print("_____ " %(a, b, sum))
```

## 02. 변수의 개념과 활용

### III. 데이터형과 형 변환

#### ■ 문자열

- 하나 이상의 문자를 저장하는 데이터형이 문자열
  - 문자열의 각 문자는 ASCII코드로 저장, 각 문자를 배열 형태로 저장
- 파이썬에서는 큰따옴표(" ") 나 작은따옴표(' ')를 문자열 기호로 사용

```
>>> name = "Kim"
```

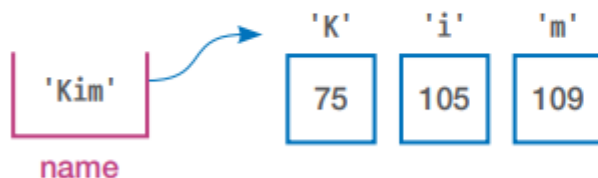


그림 3-19 문자열 처리

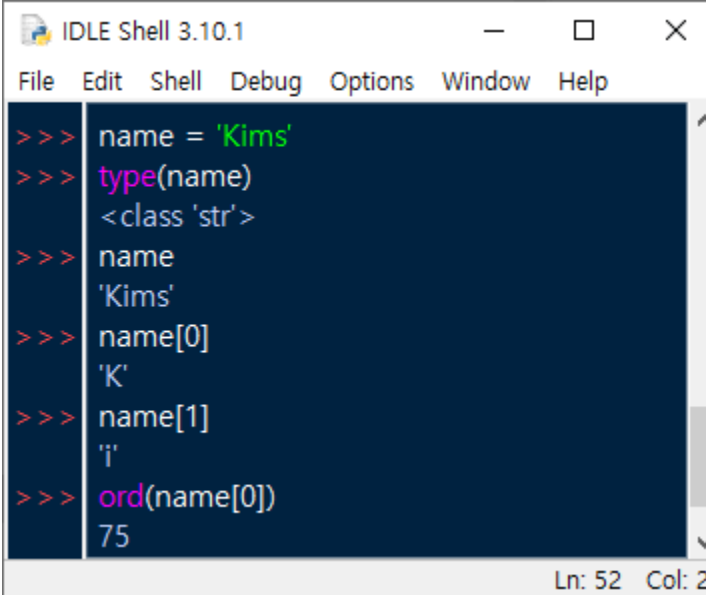
## 02. 변수의 개념과 활용

### III. 데이터형과 형 변환

#### ■ 문자열

- 하나 이상의 문자를 저장하는 데이터형이 문자열
  - 문자열의 각 문자는 ASCII코드로 저장, 각 문자를 배열 형태로 저장

#### [Python실습]



```

IDLE Shell 3.10.1
File Edit Shell Debug Options Window Help
>>> name = 'Kims'
>>> type(name)
<class 'str'>
>>> name
'Kims'
>>> name[0]
'K'
>>> name[1]
'i'
>>> ord(name[0])
75
Ln: 52 Col: 2

```

## 02. 변수의 개념과 활용

### III. 데이터형과 형 변환

- 문자열

실습 3-5

입력한 문자의 실제 숫자 값 확인하기

- ① 키보드로 하나의 문자를 입력받아 변수 ch에 저장하고 값을 확인

```
>>> ch = input("문자 하나만 입력하세요 : ")
문자 하나만 입력하세요 : a
>>> ch
'a'
```

- ② ord( ) 함수를 이용해 입력한 문자 'a'가 실제로 처리되는 값이 97임을 확인

```
>>> ord(ch)
97
>>> bin(ord(ch))
'0b1100001'
```

## 02. 변수의 개념과 활용

### III. 데이터형과 형 변환

#### ■ 문자열

##### 실습 3-6

##### 문자열의 다양한 형태 알아보기

- ① 문자열 데이터형에서 '+' 연산은 연결 의미로, '\*' 연산은 반복 의미로 사용

```
>>> a = "Hello," + "students~"
>>> type(a)
<class 'str'>
>>> b = 3
>>> print(a * b)
Hello,students~Hello,students~Hello,students~
```

- ② 긴 문자열의 경우 여러 줄로 나눈 상태(줄바꿈)로 변수에 저장

```
>>> a = """Hello,
students~"""
>>> print(a)
Hello,
students~
>>> a
'Hello,\nstudents~'
```

두 줄로 나뉜 문자열이 변수에 저장됨  
3개의 따옴표를 사용하면 입력한 모양(Enter나 Tab 등이 적용) 그대로 저장

문자열 변수 a를 출력한 모양

문자열 변수 a에 실제 저장된 값

## 02. 변수의 개념과 활용

### III. 데이터형과 형 변환

#### ■ 문자열

- ❖ 'w'는 **escape 문자**
- ❖ 'w'는 다음 한 문자를 특수한 제어 문자로 인식하게 지시
- ❖ 'n'는 new line, 't'는 tab

#### 실습 3-6

#### 문자열의 다양한 형태 알아보기

- ③ 문자열 사이에 줄바꿈 문자나 탭 문자('w')를 넣어 변수에 저장하고 출력

```
>>> b = "안녕하세요,\t학생 여러분~\n반갑습니다!"
>>> print(b)
안녕하세요,      학생 여러분~
반갑습니다!
```

#### 실습 3-7

#### 문자열 서식(formatting) 설정하기

- ① 좋아하는 과일을 변수에 저장해 두고, 하나의 문장으로 표현

```
>>> a = "사과"
>>> "I think " + a + " is the best fruit"
'I think 사과 is the best fruit'
>>> "I think %s is the best fruit" % a
'I think 사과 is the best fruit'
```

%s 자리가 변수 a의 값으로 교체

## 02. 변수의 개념과 활용

### III. 데이터형과 형 변환

#### ■ 문자열

##### 실습 3-7

##### 문자열 서식(formatting) 설정하기

- ② 여러 개의 값이 필요할 때는 % 기호 뒤에 괄호를 이용하고, 개수가 일치하도록 주의

```
>>> b = "배"
>>> "I think %s and %s are the best fruits" % (a, b)
'I think 사과 and 배 are the best fruits'
>>> c = 3
>>> "You ate %d %s today" % (c, a) ..... 변수 c는 정수형이라 %d를 사용
'You ate 3 사과 today'
```

- ③ 실수형에는 %f 포맷 코드를 사용하면 되는데, 마침표('.') 뒤에 숫자를 사용해 소수점 이하 자릿수를 정할 수 있음

```
>>> "반지름은 %d이고, 원주율은 %.2f입니다." % (10, 3.141592) ..... 소수점 이하 2자리까지만 표시하도록 설정
'반지름은 10이고, 원주율은 3.14입니다.'
```



## 02. 변수의 개념과 활용

### III. 데이터형과 형 변환

#### 여기서 잠깐    포맷 코드의 종류

표 3-4 포맷 코드의 종류와 사용 예시

포맷 코드	설명	예시	실행 결과
%s	문자열	"I like %s" %("meet")	'I like meet'
%d	정수	"키는 %d(cm)" %(180)	'키는 180(cm)'
%f	실수	"%f와 %5.1f" %(3.14, 3.14)	'3.140000와 3.1'
%c	문자	"90 이상은 %c 등급" %("A")	'90 이상은 A 등급'
%x	16진수	"100은 16진수로 %x" %(100)	'100은 16진수로 64'
%e	지수	"100은 %e" %(100)	'100은 1.000000e+02'

% 10.2 f

화면에서 10자리를 확보 ..... ↑    ↑ ..... 소수점 이하 둘째자리까지만 출력

## 02. 변수의 개념과 활용

### III. 데이터형과 형 변환

- 불린(Boolean)

실습 3-8

숫자의 크기 비교하기

- ① 정수형 변수를 만들고 값을 저장한 다음, '>' 연산자로 식을 만들어 결과를 확인

```
>>> x = 3
>>> y = 5
>>> x > y
False
```

- ② 비교 연산자를 사용한 수식의 결과를 변수 answer에 저장하고 데이터형과 저장된 값을 출력

```
>>> answer = x > y
>>> type(answer)
<class 'bool'>
>>> print(answer)
False
```

## 02. 변수의 개념과 활용

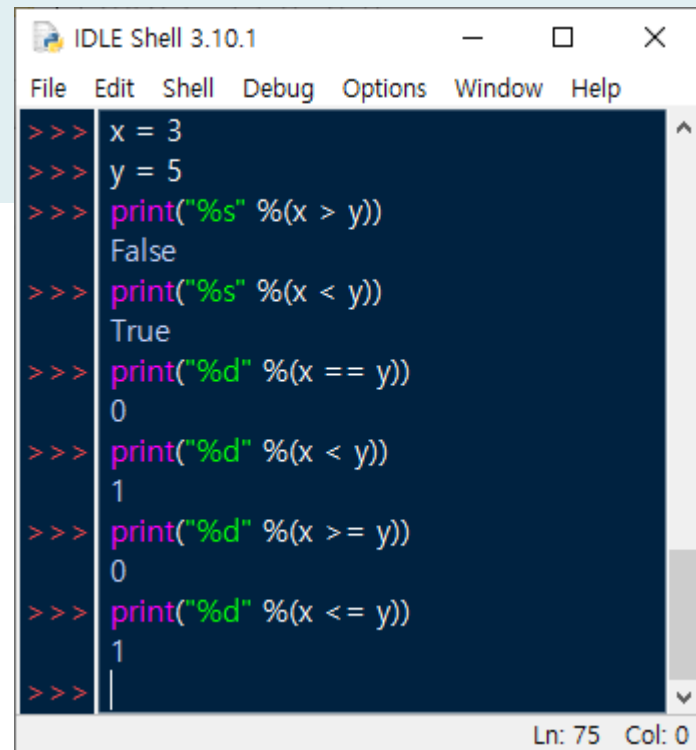
### III. 데이터형과 형 변환

#### ■ 불린(Boolean)

- ③ Boolean형의 연산 결과는 문자열로 'True' 또는 'False' 중에 하나로 표시
- ④ Boolean형의 연산 결과는 숫자로 1(True) 또는 0(False) 중에 하나로 표시

```
>>> x = 3
>>> y = 5
>>> x > y
False
```

- x = 3
- y = 5
- print("%s" %(x > y))
- print("%s" %(x < y))
- print("%d" %(x == y))
- print("%d" %(x < y))
- print("%d" %(x >= y))
- print("%d" %(x <= y))



```
IDLE Shell 3.10.1
File Edit Shell Debug Options Window Help
>>> x = 3
>>> y = 5
>>> print("%s" %(x > y))
False
>>> print("%s" %(x < y))
True
>>> print("%d" %(x == y))
0
>>> print("%d" %(x < y))
1
>>> print("%d" %(x >= y))
0
>>> print("%d" %(x <= y))
1
>>> |
Ln: 75 Col: 0
```

## 02. 변수의 개념과 활용

### III. 데이터형과 형 변환

#### ■ 형 변환

- 변수의 데이터형을 바꾸는 것이 형 변환
- `int()`, `float()`, `str()` 함수를 사용

표 3-5 데이터형에 따른 형 변환

함수	설명	예시	실행 결과
<code>int()</code>	다른 데이터형을 정수로 변환	<code>int("100")</code>	100
<code>float()</code>	다른 데이터형을 실수로 변환	<code>float(3)</code>	3.0
<code>str()</code>	다른 데이터형을 문자열로 변환	<code>str(12345)</code>	'12345'

## 02. 변수의 개념과 활용

### III. 데이터형과 형 변환

#### ■ 형 변환

실습 3-9

값을 입력받아 실수형으로 합계 출력하기

- ① 두 변수 x와 y에 입력한 값을 형 변환하여 실수형으로 저장

```
>>> x = float(input("첫번째 수 : "))
```

```
첫 번째 수 : 123.456
```

```
>>> y = float(input("두번째 수 : "))
```

```
두 번째 수 : 1000
```

정수로 입력해도 실수형으로 변환되어 저장

- ② 포맷 기호를 사용하는 경우와 문자열을 연결하는 경우, 쉼표(,)를 이용해 순서대로 출력하는 경우의 3가지 결과를 비교

```
>>> print("%f와 %f의 합은 %f" % (x, y, x+y))
```

```
123.456000와 1000.000000의 합은 1123.456000
```

포맷 기호 사용

```
>>> print(str(x) + "와 " + str(y) + "의 합은 " + str(x+y))
```

```
123.456와 1000.0의 합은 1123.456
```

숫자는 문자열로 변환한 후 연결(+)

```
>>> print(x, "와 ", y, "의 합은 ", x+y)
```

```
123.456 와 1000.0 의 합은 1123.456
```

순서대로 출력(,)

## 02. 변수의 개념과 활용

### [변수] 문제

8. 아래 코드의 실행 결과를 우측에 적으시오.

```
➤ x = 2  
➤ y = 3.1459  
➤ answer = x + y
```

- ① `print("%d" % answer)` \_\_\_\_\_
- ② `print("%d" % (x+int(y)))` \_\_\_\_\_
- ③ `print("%.2f" % (float(x)+y))` \_\_\_\_\_

9. 아래 코드 이후에 실행될 각 항의 코드 실행 결과를 우측에 적으시오.

```
➤ a = 3  
➤ b = 3.0  
➤ answer = a <= b
```

- ① `type(answer)` <class '\_\_\_\_\_'>
- ② `print(answer)` \_\_\_\_\_
- ③ `print("%d" % answer)` \_\_\_\_\_

## 02. 변수의 개념과 활용

### IV. 변수의 값 복사와 교환

- 변수  $x$ 에 10이 저장되어 있고, 다른 변수  $y$ 에  $x$ 의 값을 복사하려면  $y = x$  문장을 사용
- 대입 연산자(=)의 우 측에 특정 값이나 수식이 아닌 다른 변수명이 오면 그 변수에 저장되어 있던 값이 그대로 복사

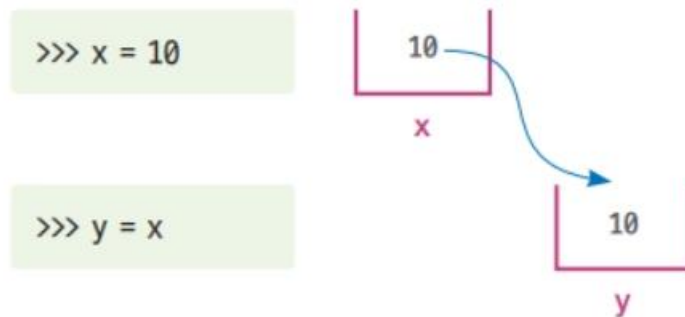


그림 3-20 변수의 값 복사

## 02. 변수의 개념과 활용

### IV. 변수의 값 복사와 교환

- x에 10, y에 20이 저장되어 있을 때 두 값을 교환하여 x는 20, y는 10을 갖도록 하는 경우라면?
- 파이썬에서는 `x, y = y, x` 라는 문장으로 간단하게 값을 교환할 수 있음

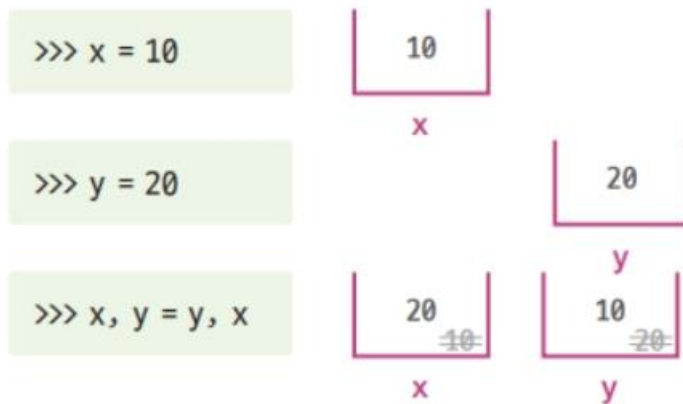


그림 3-21 두 변수의 값 교환

```
IDLE Shell 3.10.1
File Edit Shell Debug Options Window Help
>>> x = 10
>>> y = 20
>>> t = x
>>> x = y
>>> y = t
>>> print("x = %d, y = %d" % (x, y))
x = 20, y = 10
>>>
>>> x, y = y, x
>>> print("x = %d, y = %d" % (x, y))
x = 10, y = 20
>>>
```



## 02. 변수의 개념과 활용

### IV. 변수의 값 복사와 교환

- 값 교환

실습 3-10

변수의 값을 복사하고 교환하기

- ① 변수 x에 저장된 값을 y에 복사해서 출력

```
>>> x = 10
>>> y = x
>>> print("x = %d, y = %d" %(x, y))
x = 10, y = 10
```

- ② 변수의 값을 서로 교환한 후 출력

```
>>> x = "콜라"
>>> y = "주스"
>>> x, y = y, x
>>> print("x = %s, y = %s" %(x, y))
x = 주스, y = 콜라
```

## 02. 변수의 개념과 활용

### [변수] 문제

10. 아래 코드 이후에 실행될 각 항의 코드 실행 결과를 우측에 적으시오.

```
➤ x = 10  
➤ y = 20  
➤ temp = y  
➤ y = x  
➤ x = temp
```

① `print("%d" % x)` \_\_\_\_\_

② `print("%d" % y)` \_\_\_\_\_

③ `print("%d" % temp)` \_\_\_\_\_

# Thank You !

[Python]