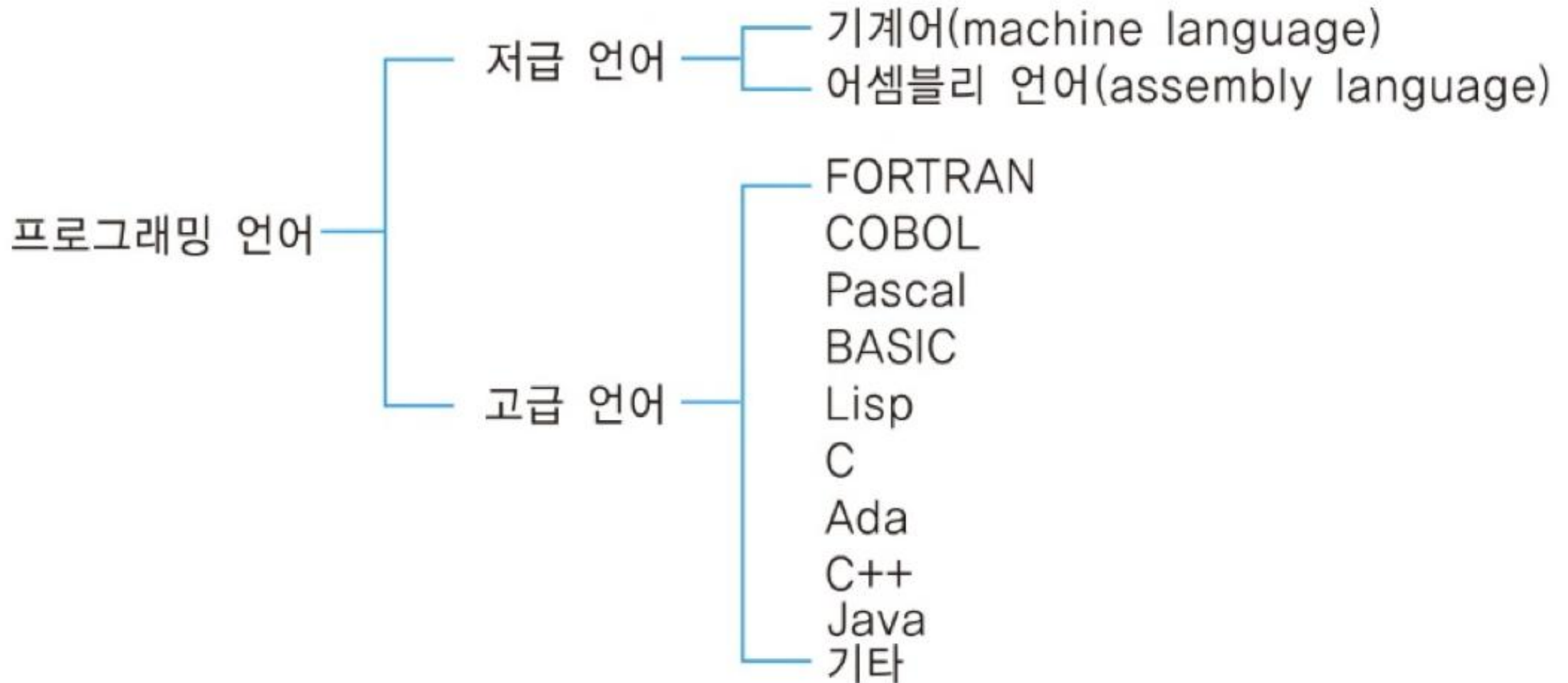




파이썬 프로그래밍

충북온라인학교 박정진

프로그래밍 언어의 분류

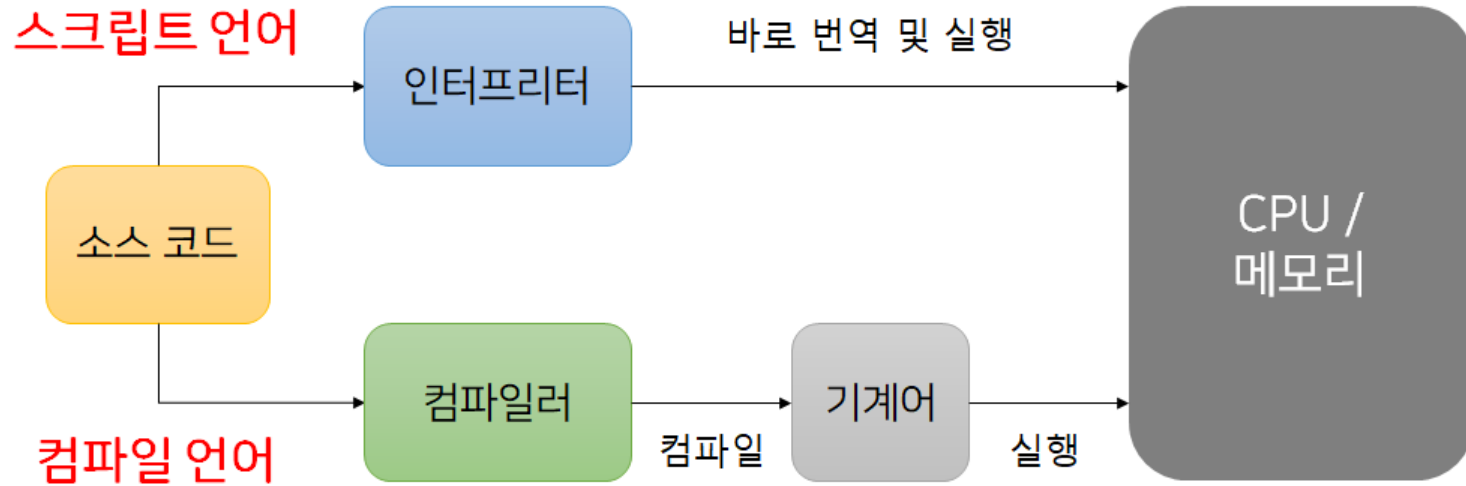




소개

➤ 파이썬은 ?

- 누가? 네덜란드의 프로그래머 귀도 반 로섬이 1991년도에 개발
- 언제? 1994년에 정식 출시한 인터프리터형 프로그래밍 언어
- 특징
 - 직관적이고 쉬운 단순한 문법
 - 다양하고 풍부한 라이브러리들을 바탕으로 한 강력한 생태계
 - 프로그래밍 교육, 인공지능, 데이터 분석 등 다양한 분야에서 사용
 - 학습 난이도 역시 초보자가 배우기 쉽기에 프로그래밍 입문용으로 많이 추천됨
 - 앱을 만드는 데는 부적합하지만 생산성이 높고 실무에서의 사용률 높음
- TIOBE 인덱스 기준 프로그래밍 언어 순위 1위



	컴파일러	인터프리터
개발 편의성	코드를 수정하고 실행하려면 컴파일을 다시 해야 한다.	코드를 수정하고 즉시 실행할 수 있다.
실행 속도	빠르다.	느리다.
보안	프로그램의 코드가 유출되지 않는다.	프로그램의 코드가 유출될 수 있다.
파일 용량	프로그램의 실행 파일 전체를 전송해야 하므로, 용량이 크다.	프로그램의 코드만 전송하면 실행이 되므로 용량이 작다.
프로그래밍 언어	C, C++처럼 비교적 저수준에 가까운 언어	Python, Ruby처럼 비교적 고수준에 가까운 언어

TIOBE Index for March 2026

Mar 2026	Mar 2025	Change	Programming Language	Ratings	Change
1	1		 Python	21.25%	-2.59%
2	4	▲	 C	11.55%	+2.02%
3	2	▼	 C++	8.18%	-2.90%
4	3	▼	 Java	7.99%	-2.37%
5	5		 C#	6.36%	+1.49%
6	6		 JavaScript	3.45%	-0.01%
7	9	▲	 Visual Basic	2.50%	-0.02%
8	8		 SQL	2.00%	-0.57%
9	16	▲▲	 R	1.88%	+0.94%
10	10		 Delphi/Object Pascal	1.80%	-0.36%
11	24	▲	 Perl	1.75%	+1.05%
12	12		 Scratch	1.63%	-0.03%

파이썬 기본 출력

➤ print() 함수를 이용한 출력

```
# 인사 출력
print('Hello world1')
print('Hello world2')
print('Hello world3', end=' ')
print('Hello world4')
print('Hello python1\nHello python2\n')

# 숫자 출력
print(100)
print(1,2,3,4,5)
print(1,2,3,4,5, sep=',')
print(12,35,20, sep=':')
```

파이썬 기본 출력



활동

출력

3.01.004 출력하기

3.01.004를 출력하는 프로그램을 작성해 보자.

코드 작성

01

02

03

04

05

06

변수

➤ 변수의 개념과 특징

- 변수 = '변하는 수'라는 의미
- (반대말) ↔ 상수 = '변하지 않고 항상 같은 수'
- 프로그램을 만들 때 컴퓨터에게 무언가를 기억시키려면 변수를 사용
- 변수는 데이터가 저장된 위치를 가리키는 특별한 이름
- 이름은 프로그래머가 지정
- 변수명 지정 규칙
- 예약어 사용 불가

규칙	가능	불가능
영문자(A-Z, a-z), 숫자(0-9), '_' 문자만 사용한다.		
영문자 또는 '_' 문자로 시작하고, 숫자로 시작하지 않는다.	a, B1, _c, n, sum, i	!, a+, a b, 1, 2.0, 'c', "abc", 2a
영문자 대문자와 소문자를 구분해 작성한다.		

변수


- 변수의 생성
 - 변수에 값을 저장하면 자동 생성됨.

- 형식

```
변수명 = 값
```

- 예시

```
level = 1  
hp = 100  
mp = 50.5  
name = 'diablo'
```

 먼저 해 보기

코드

```
01 s = "Hello"  
02 print(s)  
03 print(s)  
04 print(s)
```

설명

01 변수 s에 문자열 Hello를 저장한다.
02~04 변수 s에 저장된 값을 3회 출력한다.

실행 결과

```
Hello  
Hello  
Hello
```

변수의 출력

➤ 변수의 내용을 출력하는 3가지 방법

```
name = "파이썬"  
age = 16  
  
# 1. % 연산자 방식 (데이터 타입을 알아야 함)  
print("내 이름은 %s이고, 나이는 %d살이야." % (name, age))  
age = 17  
  
# 2. .format() 방식 (뒤에 변수를 한 번 더 써야 함)  
print("내 이름은 {}이고, 나이는 {}살이야.".format(name, age))  
age = 17  
  
# 3. f-string 방식 (압도적으로 간결하고 직관적!)  
print(f"내 이름은 {name}이고, 나이는 {age}살이야.")
```

- **f-string이란?** 문자열 따옴표 앞에 알파벳 `f`를 붙여서, 문자열 안에 변수를 직접 집어넣는 직관적인 문법

자료형

➤ 자료형의 개념

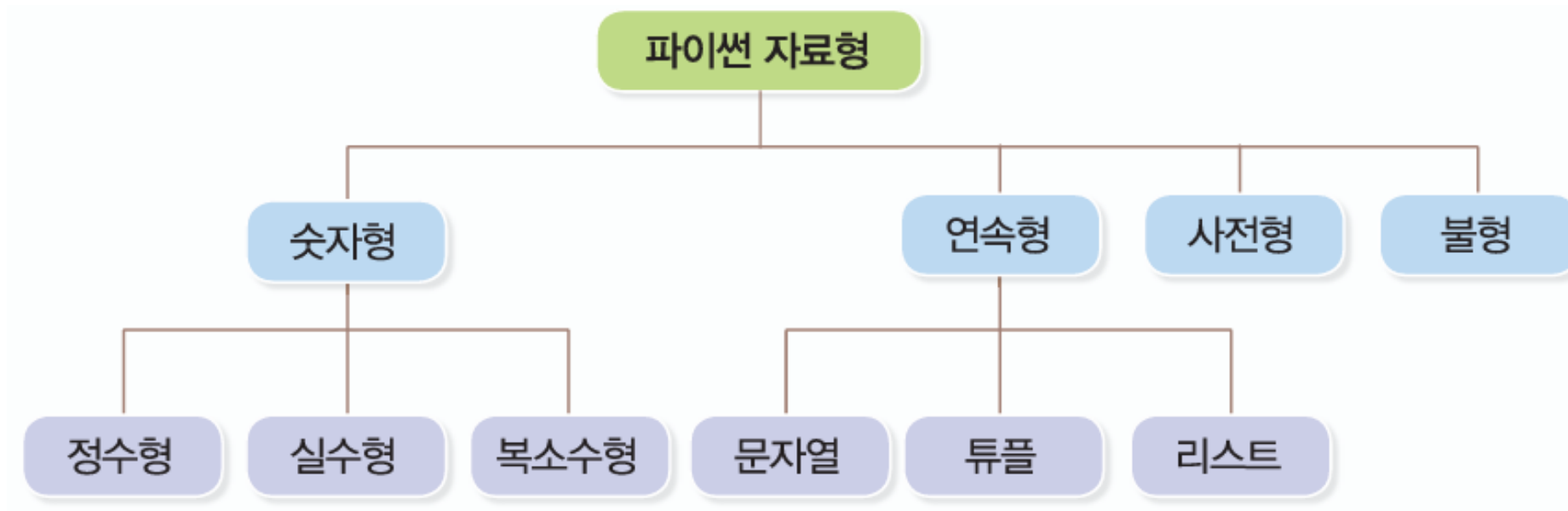
- 내부에서 데이터를 저장하는 방법과 형태, 크기를 정의
- 컴퓨터는 모든 것이 0과 1로 저장됨
- 같은 데이터도 자료형에 따라 다르게 해석 가능
 - 메모리에 나란히 저장된 72과 105라는 숫자의 의미
 - 이진수: 0100 1000 0110 1001
 - 문자열: Hi
 - 정수: 26,952
 - 실수: 8.8203125

DEC	HEX	OCT	Char	DEC	HEX	OCT	Char
43	2B	053	+	86	56	126	V
44	2C	054	,	87	57	127	W
45	2D	055	-	88	58	130	X
46	2E	056	.	89	59	131	Y
47	2F	057	/	90	5A	132	Z
48	30	060	0	91	5B	133	[
49	31	061	1	92	5C	134	\
50	32	062	2	93	5D	135]
51	33	063	3	94	5E	136	^
52	34	064	4	95	5F	137	_
53	35	065	5	96	60	140	`
54	36	066	6	97	61	141	a
55	37	067	7	98	62	142	b
56	38	070	8	99	63	143	c
57	39	071	9	100	64	144	d
58	3A	072	:	101	65	145	e
59	3B	073	;	102	66	146	f
60	3C	074	<	103	67	147	g
61	3D	075	=	104	68	150	h
62	3E	076	>	105	69	151	i
63	3F	077	?	106	6A	152	j
64	40	100	@	107	6B	153	k
65	41	101	A	108	6C	154	l
66	42	102	B	109	6D	155	m
67	43	103	C	110	6E	156	n
68	44	104	D	111	6F	157	o
69	45	105	E	112	70	160	p
70	46	106	F	113	71	161	q
71	47	107	G	114	72	162	r
72	48	110	H	115	73	163	s
73	49	111	I	116	74	164	t
74	4A	112	J	117	75	165	u
75	4B	113	K	118	76	166	v
76	4C	114	L	119	77	167	w
77	4D	115	M	120	78	170	x
78	4E	116	N	121	79	171	y
79	4F	117	O	122	7A	172	z
80	50	120	P	123	7B	173	{
81	51	121	Q	124	7C	174	
82	52	122	R	125	7D	175	}
83	53	123	S	126	7E	176	~
84	54	124	T	127	7F	177	DEL
85	55	125	U	made by Lee Jae-wook			

자료형

➤ 자료형의 개념

- 파이썬의 자료형



자료형의 특성

① 숫자형

숫자형은 우리가 일반적으로 사용하는 숫자 데이터를 의미하며, 파이썬에서 사용되는 숫자형의 종류에는 정수, 실수, 복소수 등이 있다.

예시

```
orange = 100    # 정숫값
pi = 3.14159    # 실숫값
cpx = 10 + 5j   # 복소숫값
```

숫자형 데이터는 대부분 산술 연산자와 함께 사용된다. 숫자형 자료형에서 사용할 수 있는 연산자는 7가지가 있다.

표 III-7 숫자형의 연산자

숫자형	예시	연산자	연산자 사용 예시
정수	-808, 1023, 0	+, -, *, /, %, //, **	$10+2*3+4 = 20$
실수	-3.14159, 2.78		$1.5**2 = 2.25$
복소수	$2+3j$, <code>complex(4, -5)</code>		$(2+3j)-(4-2j) = -2+5j$

자료형의 특성

② 연속형

연속형(sequence)은 여러 개의 데이터가 연속으로 저장되는 자료형을 말하며, 크게 문자열, 튜플, 리스트의 3가지 자료형이 포함된다.

문자열 문자열 자료형은 0개 이상의 문자로 구성된 데이터를 의미한다. 문자열 데이터를 표현하기 위해서는 작은따옴표(' ') 혹은 큰따옴표(" ") 를 사용한다.

예시

```
alpha = 'A'  
name = "information"
```

문자열에서 사용할 수 있는 연산자로는 '+'와 '*'가 있다. 숫자형 변수에서 사용한 연산자와 동일한 형태이지만 실제 작동하는 방식은 다르다.

표 III-8 문자열의 연산자

연산자	예시	연산 결과
+	"Hello" + "World!"	HelloWorld!
*	"pi!"*3	pi!pi!pi!

튜플 리스트와 유사하게 여러 데이터를 나열해서 저장하지만 괄호를 사용하여 표현하며, 한 번 만들어진 튜플의 데이터는 수정, 삭제, 생성이 불가능하다.

예시 num = (1, 2, 3)

표 III-9 튜플의 연산자

연산자	예시	연산 결과
+	(1, 2, 3) + (4, 5, 6)	(1, 2, 3, 4, 5, 6)
*	(0, 10, 2)*3	(0, 10, 2, 0, 10, 2, 0, 10, 2)

리스트 숫자, 문자열 등 여러 데이터를 나열해서 저장할 수 있는 자료형이다. 각각의 데이터는 콤마(,)로 구분하고 전체는 대괄호로 묶어서 표현한다. 하나의 변수에 여러 개의 데이터를 저장할 때 사용하는 자료형이다.

예시 num = [1, 2, 3, 4, 5, 6, 7] # 리스트

표 III-10 리스트의 연산자

연산자	예시	연산 결과
+	[1, 0, 2, 3] + [0, 8, 0, 8]	[1, 0, 2, 3, 0, 8, 0, 8]
*	[1, 2, 4]*3	[1, 2, 4, 1, 2, 4, 1, 2, 4]

③ 사전형

사전형(dictionary)은 연속형과 달리 키(key)와 값(value)이 쌍으로 이루어진 구조로 데이터를 저장하는 자료형이다. 키와 값은 콜론(:)으로 구분하며, 전체는 중괄호로 묶어서 표현한다.

예시 `dic = {"apple": "사과", "orange": "오렌지"}`

④ 불형

불형(bool)은 참(True, 1)과 거짓(False, 0)을 표현하는 자료형이다.

예시 `check = True`

표 III-11 불형의 연산자

연산자	예시	연산 결과
and	x and y	x와 y가 모두 참이라면 참, 아니라면 거짓
or	x or y	x가 참 또는 y가 참이라면 참, 아니라면 거짓
not	not x	x가 거짓이라면 참, 아니라면 거짓

자료형의 특성

② 연속형

연속형(sequence)은 여러 개의 데이터가 연속으로 저장되는 자료형을 말하며, 크게 문자열, 튜플, 리스트의 3가지 자료형이 포함된다.

문자열 문자열 자료형은 0개 이상의 문자로 구성된 데이터를 의미한다. 문자열 데이터를 표현하기 위해서는 작은따옴표(' ') 혹은 큰따옴표(" ") 를 사용한다.

예시

```
alpha = 'A'  
name = "information"
```

문자열에서 사용할 수 있는 연산자로는 '+'와 '*'가 있다. 숫자형 변수에서 사용한 연산자와 동일한 형태이지만 실제 작동하는 방식은 다르다.

표 III-8 문자열의 연산자

연산자	예시	연산 결과
+	"Hello" + "World!"	HelloWorld!
*	"pi!"*3	pi!pi!pi!

자료형의 특성

➤ 자료형 확인

- `type()` 이라는 명령어를 사용

```
hp = 100
mp = 50.5
name = 'diablo'
alive = True
items = []

print('hp =', hp, type(hp))
print('mp =', mp, type(mp))
print('name =', name, type(name))
print('alive =', alive, type(alive))
print('items =', items, type(items))
```

```
hp = 100 <class 'int'>
mp = 50.5 <class 'float'>
name = diablo <class 'str'>
alive = True <class 'bool'>
items = [] <class 'list'>
```

자료형의 특성

➤ 파이썬 자료형이 표현 바뀐 미 예시

▼ [표 II-2] 파이썬 자료형의 표현 방법 및 예시

자료형	표현 방법	파이썬 자료형 표현 예시
정수	소수점 없이 부호와 숫자를 붙여 쓴다.	-1, 0, 1, 2
문자열	<ul style="list-style-type: none">• 큰따옴표(")로 묶어 표현한다.• 작은따옴표(')로 묶어 표현할 수도 있다.	<ul style="list-style-type: none">• "hello", "1", "123"• 'a', 'A', '1'
실수	소수점과 함께 부호와 숫자를 붙여 쓴다.	1.0, 1.414, -2.718
논릿값	True, False만 사용한다.	True, False

TIP

파이썬에서는 문자와 문자열을 따로 구분하여 표현하지 않는다. 문자도 문자열로 저장한 후 처리한다.

+

C/C++ 언어에서 문자 1개는 작은따옴표(')로 묶어 표현하고, 문자열은 큰따옴표(")로 묶어 서로 다르게 표현한다.

프로그래밍 언어에 따라 내부적으로 데이터를 저장하고 처리하는 방법이 다르다.

자료형의 특성

➤ 연산과 자료형

- 컴퓨터는 같은 자료형의 데이터만 연산이 가능

```
age = 16
txt = '세에 입학'
string = age + txt
print(string)
```

TypeError: unsupported operand type(s) for
+: 'int' and 'str'

자료형 차이에 따른 계산 오류 메시지

- 자료형이 다른 데이터 간에 연산을 하려면 영 변환 필요

```
age = 16
txt = '세에 입학'
string = str(age) + txt
print(string)
```

```
age = "16"
age = int(age) + 10
print(age)
```

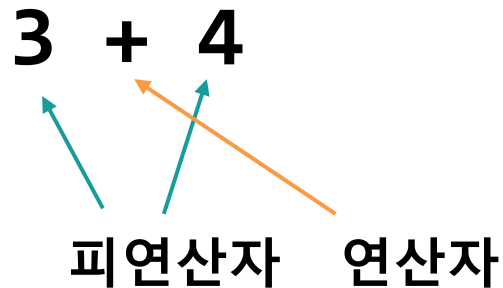
자료형 변환

➤ 자료형 변환 함수

자료형 변환 함수	자료형 변환 내용
<code>int()</code>	수 또는 문자열을 정숫값으로 변환한다.
<code>chr()</code>	정숫값을 유니코드 문자로 변환한다.
<code>ord()</code>	유니코드 문자를 정숫값으로 변환한다.
<code>str()</code> , <code>repr()</code>	어떤 값을 문자열로 변환한다.
<code>float()</code>	수 또는 문자열을 실수로 변환한다.
<code>bool()</code>	수 또는 문자열을 논릿값으로 변환한다.

연산자(operator)

➤ 연산자와 피연산자



➤ 이항연산자

- 피연산자가 두 개인 연산자
- +, -, ×, ÷

➤ 단항연산자

- 피연산자가 한 개인 연산자
- -, +

➤ 파이썬의 연산자 표기

의미	수학	파이썬	비고
덧셈	+	+	
뺄셈	-	-	
곱셈	×	*	√
나눗셈	÷	/	√
나머지		%	
같다	=	==	√
다르다		!=	√
크다	>, ≥	>, >=	
작다	<, ≤	<, <=	

산술 연산자

연산자	의미	예시	결괏값
+	합하기	2+3	5
		3+2	5
-	빼기	2-3	-1
		3-2	1
*	곱하기	2*3	6
		3*2	6
/	나누기	2/3	0.6666666666666666
		3/2	1.5
%	나눈 나머지	2%3	2
		3%2	1
//	나눈 몫	2//3	0
		3//2	1
**	거듭제곱	2**3	8
		3**2	9

산술 연산자

➤ 산술 연산자 실습 코드

```
# 에러가 발생하는 코드
# 그 이유는?
a = input()
b = input()
print(f'{a} + {b} =', a+b)
print(f'{a} - {b} =', a-b)
print(f'{a} * {b} =', a*b)
print(f'{a} / {b} =', a/b)
print(f'{a} % {b} =', a%b)
print(f'{a} ** {b} =', a**b)
print(f'{a} // {b} =', a//b)
```

➤ 점검

```
a = input()
b = input()
print(type(a))
print(type(b))
print(a+b)
print(a*100)
```

➤ 문자열의 덧셈과 곱셈

산술 연산자

➤ 산술 연산자 실습 코드

```
# 수정된 코드
a = int(input())
b = int(input())

print(f'{a} + {b} =', a+b)
print(f'{a} - {b} =', a-b)
print(f'{a} * {b} =', a*b)
print(f'{a} / {b} =', a/b)
print(f'{a} % {b} =', a%b)
print(f'{a} ** {b} =', a**b)
print(f'{a} // {b} =', a//b)
```

➤ 입력과 출력

```
3
4

3 + 4 = 7
3 - 4 = -1
3 * 4 = 12
3 / 4 = 0.75
3 % 4 = 3
3 ** 4 = 81
3 // 4 = 0
```

파이썬 표준 입출력

➤ input() 함수의 사용

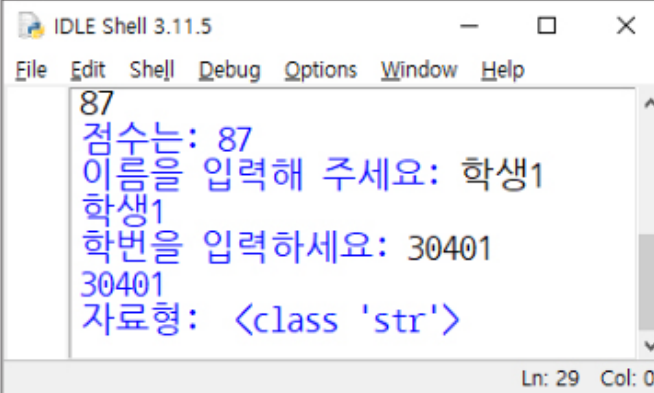
예시

```
# 표준 입출력
t = input()
print("점수는:",t)

#input() 함수에는 문구를 추가할 수 있음
name = input("이름을 입력해 주세요: ")
print(name)

#input() 함수는 입력값을 문자열로 저장
num = input("학번을 입력하세요: ")
print(num)
print("자료형: ", type(num))
```

결과



```
IDLE Shell 3.11.5
File Edit Shell Debug Options Window Help
87
점수는: 87
이름을 입력해 주세요: 학생1
학생1
학번을 입력하세요: 30401
30401
자료형: <class 'str'>
Ln: 29 Col: 0
```



스스로 해보기

빈칸에 올바른 명령어를 채워 이름과 성적을 입력받아 출력하는 프로그램을 완성해 보자.

소스 코드

```
# 입력 형식에 맞게 입력받기
# 평균 계산을 위해 정수형으로 변경
name = input("이름을 입력하세요: ")
```

exam1 = ①

exam2 = ②

정수형으로 변경

exam1 = ③

exam2 = ④

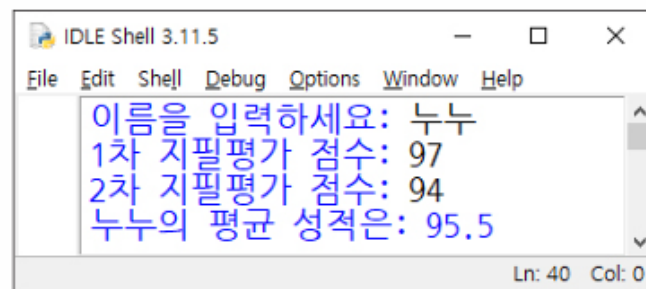
평균 구하기

ave = ⑤

출력 형식에 맞게 출력하기

⑥

결과



input()으로 입력받은 값은 문자열 형태로 저장되므로 올바른 산술 계산을 위해서는 정수형(int) 또는 실수형(float)으로 변환이 필요하다.

리스트

➤ 리스트의 개념

- 여러 개의 데이터를 순서대로 한 줄로 저장할 수 있는 자료형
- 내용물의 자료형은 서로 달라도 무방

➤ 예시

```
rainbow = ['빨', '주', '노', '초', '파', '남', '보']  
print(rainbow)
```

```
hakbun = [2101, 2102, 2103, 2104, 2025]  
print(hakbun)
```

```
#      학년,반,번호,   이름,  남자?, 재학중?  
student = [2, 1, 13, '홍길순', True, True]
```

리스트

➤ 인덱싱

```
# 0번부터 시작함에 유의
weekdays = ['일', '월', '화', '수', '목', '금', '토']
print(weekdays)
print(weekdays[0])
print(weekdays[1])
print(weekdays[2])
print(weekdays[3])

# 리스트의 길이를 구할 때 사용하는 len()
print(len(weekdays))
print(weekdays[len(weekdays)-1])

# 특정 원소만 교체
weekdays[5] = 'Friday'
# 변경 확인
print(weekdays)
```

인덱스	데이터
0	일
1	월
2	화
3	수
4	목
5	금
6	토

리스트

➤ 리스트 조작 함수

표 III-12 리스트에서 자주 사용하는 메서드

* 리스트(num)의 초기값: [1, 0, 2, 3]

메서드의 종류	동작	사용 예시	리스트(num) 상태
append()	리스트의 끝에 데이터 추가	num.append(8)	[1, 0, 2, 3, 8]
insert(a, b)	리스트의 a 번째 위치에 b 값 추가	num.insert(2, 9)	[1, 0, 9, 2, 3]
pop(t)	리스트의 t번 인덱스 값 호출 후 제거	num.pop(3)	[1, 0, 2]
sort()	리스트의 데이터 정렬	num.sort()	[0, 1, 2, 3]
reverse()	리스트의 값을 역순으로 배치	num.reverse()	[3, 2, 0, 1]

리스트

➤ 리스트 조작 메서드 실습

<pre>lst = [4,6,8] # 리스트 초기값 lst.append(10) # 원소 10 추가 print(lst)</pre>	<pre>[4, 6, 8, 10]</pre>
<pre>lst = [0,2,4] + lst # 리스트 [0,2,4]에 lst를 합침 print(lst)</pre>	<pre>[0, 2, 4, 4, 6, 8, 10]</pre>
<pre>a = lst.pop() # 맨 마지막 원소 인출 print(a, lst)</pre>	<pre>10 [0, 2, 4, 4, 6, 8]</pre>
<pre>a = lst.pop(0) # 0번째 원소 인출 print(a, lst)</pre>	<pre>0 [2, 4, 4, 6, 8]</pre>
<pre>lst.remove(4) # 원소 4를 제거(한번만 제거됨) print(lst)</pre>	<pre>[2, 4, 6, 8]</pre>
<pre>lst.insert(2,5) # 2번째 위치에 5를 삽입 print(lst)</pre>	<pre>[2, 4, 5, 6, 8]</pre>
<pre>lst.reverse() # 리스트 순서 뒤집기 print(lst)</pre>	<pre>[8, 6, 5, 4, 2]</pre>
<pre>lst.sort() # 리스트 정렬 print(lst)</pre>	<pre>[2, 4, 5, 6, 8]</pre>
<pre>lst.clear() # 리스트 초기화 print(lst)</pre>	<pre>[]</pre>

리스트



리스트를 활용한 프로그램 작성하기

스스로

해보기

다음은 1학년 0반 학생들의 중간고사 성적이 저장된 리스트이다.

리스트명 score	100	22	13	12	43	87	97	23	91	29
---------------	-----	----	----	----	----	----	----	----	----	----

1. 리스트의 맨 마지막 학생 점수를 29점에서 92점으로 수정하는 소스 코드를 작성해 보자.
2. 리스트의 함수를 사용하여 점수를 내림차순으로 정렬하고 리스트 전체를 출력하는 소스 코드를 작성해 보자.

리스트

➤ split()

- 하나의 긴 문자열을 특정 기준 (구분자)으로 잘라서 여러 개의 조각으로 나누고, 그 조각들을 리스트로 반환
- 마치 긴 가래떡을 일정한 간격으로 썰어서 접시(리스트)에 담아주는 역할이라고 생각하면 이해하기 쉬움

➤ 기본문법

```
문자열.split(sep=None, maxsplit=-1)
```

➤ 매개변수

- sep (구분자): 문자열을 나눌 기준이 되는 문자
 - 괄호 안에 아무것도 넣지 않으면(split()), 띄어쓰기(공백), 탭(Tab), 줄바꿈(Enter) 등을 기준으로 자동으로 나눕니다.
- maxsplit (최대 분할 횟수): 문자열을 최대 몇 번 자를 것인지 지정
 - 기본값은 -1이며, 이는 자를 수 있을 만큼 모두 자른다는 뜻입니다.

리스트

- 예제 1: 기본 사용법 (공백을 기준으로 분리)

```
text = "파이썬 프로그래밍은 정말 재미있습니다"
result = text.split()
print(result)

# 출력: ['파이썬', '프로그래밍은', '정말', '재미있습니다']
```

- 예제 2: 특정 문자를 기준으로 분리 (데이터 분석 활용)

```
# 주식 종목명, 종목코드, 현재가가 쉼표(,)로 연결된 데이터
stock_info = "삼성전자,005930,85000"
result = stock_info.split(",")

print(result)
# 출력: ['삼성전자', '005930', '85000']

# 리스트 인덱싱을 통해 원하는 값만 추출 가능
name = result[0]
price = result[2]
print(f"{name}의 현재가는 {price}원 입니다.")
# 출력: 삼성전자의 현재가는 85000원 입니다.
```

리스트

➤ 예제 3: 특정 문자를 기준으로 분리

```
h, m, s = input("현재 시간을 입력하세요.\nhh:mm:ss ").split(':')  
  
print(f'현재 시간은 {h}시 {m}분 {s}초 입니다.')
```

➤ 예제 4:

```
# 분리된 자료가 문자열  
nums = input('숫자 세 개를 입력하세요: ').split()  
print(nums[0]+nums[1]+nums[2]) # nums 에는 문자열이 들어있다.  
  
# 분리된 자료를 숫자로 변환  
nums = list(map(int, input('숫자 세 개를 입력하세요: ').split()))  
print(nums[0]+nums[1]+nums[2])
```

```
숫자 세 개를 입력하세요: 1 2 3  
123  
숫자 세 개를 입력하세요: 1 2 3  
6
```

2차원 리스트

➤ 개념

- 리스트의 원소가 리스트인 리스트

➤ 예시

학년	반	번호	이름	남녀
2	1	1	홍길동	남
2	1	2	홍길순	녀
2	1	3	홍길남	남
2	1	4	홍길녀	녀
2	1	5	홍길만	남

➤ 코드

```
# 방법1
stu1 = [2, 1, 1, '홍길동', True]
stu2 = [2, 1, 2, '홍길순', False]
stu3 = [2, 1, 3, '홍길남', True]
stu4 = [2, 1, 4, '홍길녀', False]
stu5 = [2, 1, 5, '홍길만', True]
students = [stu1, stu2, stu3, stu4, stu5]
print(students)

# 방법2
people = [
    [2, 1, 1, '홍길동', True],
    [2, 1, 2, '홍길순', False],
    [2, 1, 3, '홍길남', True],
    [2, 1, 4, '홍길녀', False],
    [2, 1, 5, '홍길만', True],
]
print(people)
```

2차원 리스트

➤ 인덱싱

- 더블 인덱싱 [][] 사용
- [행][열]

➤ 예시

index	0	1	2	3	4
0	2	1	1	홍길동	남
1	2	1	2	홍길순	녀
2	2	1	3	홍길남	남
3	2	1	4	홍길녀	녀
4	2	1	5	홍길만	남

➤ 코드

```
# 2차원 리스트 선언
st = [
    [2, 1, 1, '홍길동', True],
    [2, 1, 2, '홍길순', False],
    [2, 1, 3, '홍길남', True],
    [2, 1, 4, '홍길녀', False],
    [2, 1, 5, '홍길만', True],
]

print(st[0])      # 첫 번째 학생
print(st[0][3])  # 첫 번째 학생의 이름
print(st[1][3])  # 두 번째 학생의 이름
print(st[4][3])  # 4행 3열 (0-based)
```

다차원



정보 실습 활동

2차원 리스트를 이용한 프로그램 작성하기



개인

☁ 학생의 이름과 성적을 data 리스트에 저장하여 출력하는 프로그램을 작성해 보자.

이름	루미	호두	누룽	초코	루루	코코	바방
성적	100	58	41	87	97	23	91

1 출력 결과에 맞게 소스 코드를 완성해 보자.

소스 코드	결과
<pre># 이름과 성적 저장하기 data = [['루미', 100], ['호두', 58], ['누룽', 41], ['초코', 87], ['루루', 97], ['코코', 23], ['바방', 91]] # 이름과 성적 출력하기 print("3번 학생:", ①, "점") print("마지막 학생: ", ②, "점")</pre>	

2 data 리스트에 새로운 학생의 이름과 성적이 나오도록 소스 코드를 완성해 보자.

소스 코드	결과
<pre># 새로운 학생 추가하기 name = ① score = ② data.append(③) # 전체 데이터 출력하기 print(④)</pre>	

다차원 리스트

➤ 개념

- 3차원 이상의 n차원 리스트
- 연속된 인덱싱 가능 [][][...]

➤ 예시

1반	0	1	2	3	4
2반	0	1	2	3	4
3반	0	1	2	3	4
0	2	3	1	홍상동	남
1	2	3	2	홍상순	녀
2	2	3	3	홍상남	남

➤ 예제

```
# 2차원 리스트 선언
st = [ #1반
      [[2, 1, 1, '홍길동', True],
       [2, 1, 2, '홍길순', False],
       [2, 1, 3, '홍길남', True]],

      #2반
      [[2, 2, 1, '홍만동', True],
       [2, 2, 2, '홍만순', False],
       [2, 2, 3, '홍만남', True]],

      #3반
      [[2, 3, 1, '홍상동', True],
       [2, 3, 2, '홍상순', False],
       [2, 3, 3, '홍상남', True]]
    ]

print(st[1][2][3]) # 2반 3번 이름
```

비교 연산자

➤ 개요

- 좌변과 우변을 비교하는 연산
- 결과는 불리언(boolean) 값으로 True, False 둘 중 하나

▼ [표 II-5] 비교 연산에 사용하는 연산자

연산자	의미	예시	결괏값
<	(왼쪽 값이) 작다.	2<2	False
		2<3	True
		3<2	False
>	(왼쪽 값이) 크다.	2>2	False
		2>3	False
		3>2	True
==	(값이 서로) 같다.	2==2	True
		2==3	False
		3==2	False
!=	(값이 서로) 다르다.	2!=2	False
		2!=3	True
		3!=2	True
<=	(왼쪽 값이) 작거나 같다.	2<=2	True
		2<=3	True
		3<=2	False
>=	(왼쪽 값이) 크거나 같다.	2>=2	True
		2>=3	False
		3>=2	True

비교 연산자

➤ 비교 연산자 실습 코드

```
# 비교 연산자 테스트
txt = input('숫자 세 개를 공백으로 분리하여 입력:')
a, b, c = map(int, txt.split())
print(f'a={a}, b={b}, c={c} \n')

print('a>b, a>=b, a>c, a>=c')
print(a>b, a>=b, a>c, a>=c, '\n')

print('a<b, a<=b, a<c, a<=c')
print(a<b, a<=b, a<c, a<=c, '\n')

print('a==b, a==c, a!=b, a!=c')
print(a==b, a==c, a!=b, a!=c, '\n')

n, m = input('이름 두 개를 입력:').split()
print('n==m, n!=m, n>m, n<m')
print(n==m, n!=m, n>m, n<m, '\n')
```

숫자 세 개를 공백으로 분리하여 입력:1 1 2
a=1, b=1, c=2

a>b, a>=b, a>c, a>=c
False True False False

a<b, a<=b, a<c, a<=c
False True True True

a==b, a==c, a!=b, a!=c
True False False True

이름 두 개를 입력:홍길동 홍길순
n==m, n!=m, n>m, n<m
False True False True

논리 연산자

➤ 개요

- 논리 연산자를 사용하면, 여러 가지 복잡한 조건을 간단히 표현할 수 있다.
- 논리 연산자를 사용한 결과는 참 또는 거짓의 불(boolean)으로 계산된다.

▼ [표 II-6] 논리 연산에 사용하는 연산자

연산자	의미	예시	결괏값
not	논리 부정(반대)	not True	False
		not False	True
or	논리합(한 개 이상 참이면 참)	False or False	False
		False or True	True
		True or False	True
		True or True	True
and	논리곱(모두 참이면 참)	False and False	False
		False and True	False
		True and False	False
		True and True	True



하나 더 알기

참, 거짓과 그 값에 대한 논리 연산인 논리 부정(not), 논리합(or), 논리곱(and) 등은 수학자 조지 불(George Boole)이 처음 만들었다. 조지 불이 만들었다는 의미로 True, False를 불값, not, or, and를 불 연산자라고 부르기도 한다.

논리 연산자를 사용하면 참, 거짓의 논릿값이 계산된 후, 그 결과로 새로운 참, 거짓의 논릿값이 만들어진다.

논리 연산자

➤ 논리 연산자 실습 코드

```
# 논리 연산자 테스트
a, b, c = map(int, input('숫자 세 개를 입력: ').split())
print(f'a={a}, b={b}, c={c} \n')

print('a>b and b>c :', a>b and b>c)
print('a<b and b<c :', a<b and b<c)

print('a>b or b>c :', a>b or b>c)
print('a<b or b<c :', a<b or b<c)

# 오직 0만 False이고, 나머지 모든 수는 True이다
print('bool(a), bool(b), bool(c):', bool(a), bool(b), bool(c))
print('not(a), not(b), not(c):', not(a), not(b), not(c))
```

숫자 세 개를 공백으로 분리하여 입력: -1 0 1
a=-1, b=0, c=1

a>b and b>c : False
a<b and b<c : True
a>b or b>c : False
a<b or b<c : True

bool(a), bool(b), bool(c): True False True
not(a), not(b), not(c): False True False

파이썬의 비교 연산자 체이닝(Chained Comparisons)

➤ 개념

- 파이썬은 사람이 수학 수식을 읽는 흐름 그대로 연산자를 이어서 쓸 수 있도록 지원함.
- 파이썬 내부에서는 연속된 비교를 다음과 같이 and 연산자로 묶어서 해석
 - $c > b > a \rightarrow (c > b) \text{ and } (b > a)$
 - $30 > 20 > 10 \rightarrow (30 > 20) \text{ and } (20 > 10)$
 - 결과: True and True \rightarrow True

➤ 테스트 코드

```
a=10; b=20; c=30;

print(c>b>a) # 결과는?

# C나 Java에 익숙한 사람들은
# 30>20>10
# True>10 즉 False를 상상하겠지만...

# 파이썬의 내부 작동
# (c>b) and (b>a)
```

논리 연산자

논리 연산자 시스템 코드



활동

논리 연산자 2, 3, 5의 공배수 판별하기

한 개의 정숫값을 입력받아 그 값이 2의 배수이면서 3의 배수이고 또한 5의 배수인 경우, 즉 입력받은 값이 2, 3, 5의 공배수인 경우에만 True를 출력하고 그 외의 경우는 False를 출력하는 프로그램을 작성해 보자.

코드 작성

01

02

03

연산자 우선순위

우선 순위	연산자	① () 괄호	설명
높음 ↑ ↓ 낮음	**	②	거듭제곱
	+x, -x, ~x		단항 +, 단항 -, 비트 단위 not
	*, /, //, %	③	곱, 나눗셈, 몫, 나머지
	+, -	④	덧셈, 뺄셈
	<<, >>		비트 시프트
	&		비트 단위 and
	^		비트 단위 xor
			비트 단위 or
	<, <=, >, >=, !=, ==	⑤	비교
	not x	⑥	논리 부정(not)
and	⑦	논리곱(and)	
or	⑧	논리합(or)	

참고: <https://docs.python.org/3/reference/expressions.html> 6.17. Operator precedence

연산자 우선순위

➤ 복잡한 수식의 계산

다음 수식의 계산 결과는?

```
a = 10+4*3-1
```

```
print(a)
```

```
b = 10+4*(3-1)
```

```
print(b)
```

```
r=4+5*6/(2+1)+15-5*2**2
```

```
print(r)
```

우선 순위	연산자	설명	
높음 ↑	**	거듭제곱	
	+x, -x, ~x	단항 +, 단항 -, 비트 단위 not	
	*, /, //, %	곱, 나눗셈, 몫, 나머지	
	+, -	덧셈, 뺄셈	
	<<, >>	비트 시프트	
	&	비트 단위 and	
	^	비트 단위 xor	
		비트 단위 or	
	↓ 낮음	<, <=, >, >=, !=, ==	비교
		not x	논리 부정(not)
and		논리곱(and)	
or		논리합(or)	

특수 연산자

➤ 거듭제곱 연산자 **

파이썬의 거듭제곱 연산자 **을 사용하면, 다른 프로그래밍 언어로는 계산하기 까다로운 매우 큰 수도 쉽게 만들어 낼 수 있다.

코드

01

```
print(2**1000)
```

설명

01

2^{1000} 출력

실행
결과

```
10715086071862673209484250490600018105614048117055336074437503883703510511249361  
22493198378815695858127594672917553146825187145285692314043598457757469857480393  
45677748242309854210746050623711418779541821530464749835819412673987675591655439  
46077062914571196477686542167660429831652624386837205668069376
```

거듭제곱 연산자 **을 사용하면, 제곱근도 매우 간단히 계산할 수 있다.

코드

01

```
print(2**(0.5))
```

설명

01

$2^{\frac{1}{2}}$ 값, 즉 $\sqrt{2}$ 값 출력

실행
결과

```
1.4142135623730951
```

특수 연산자

➤ 바다코끼리 연산자 :=

파이썬 버전 3.8부터는 바다코끼리의 눈과 이빨처럼 생긴 바다코끼리 연산자 :=도 제공된다. 바다코끼리 연산자를 대입 연산자처럼 사용하면, 어떤 값을 변수에 저장한 후 대입식 전체를 그 값으로 바꿀 수 있다.

코드

```
01 print(n := int(input()))  
02 print(n**(0.5))
```

설명

```
01 입력받은 값을 정수로 변환하여 변수 n에 저장하고, 그 값을 출력  
02  $\sqrt{n}$  값 출력
```

입력

3

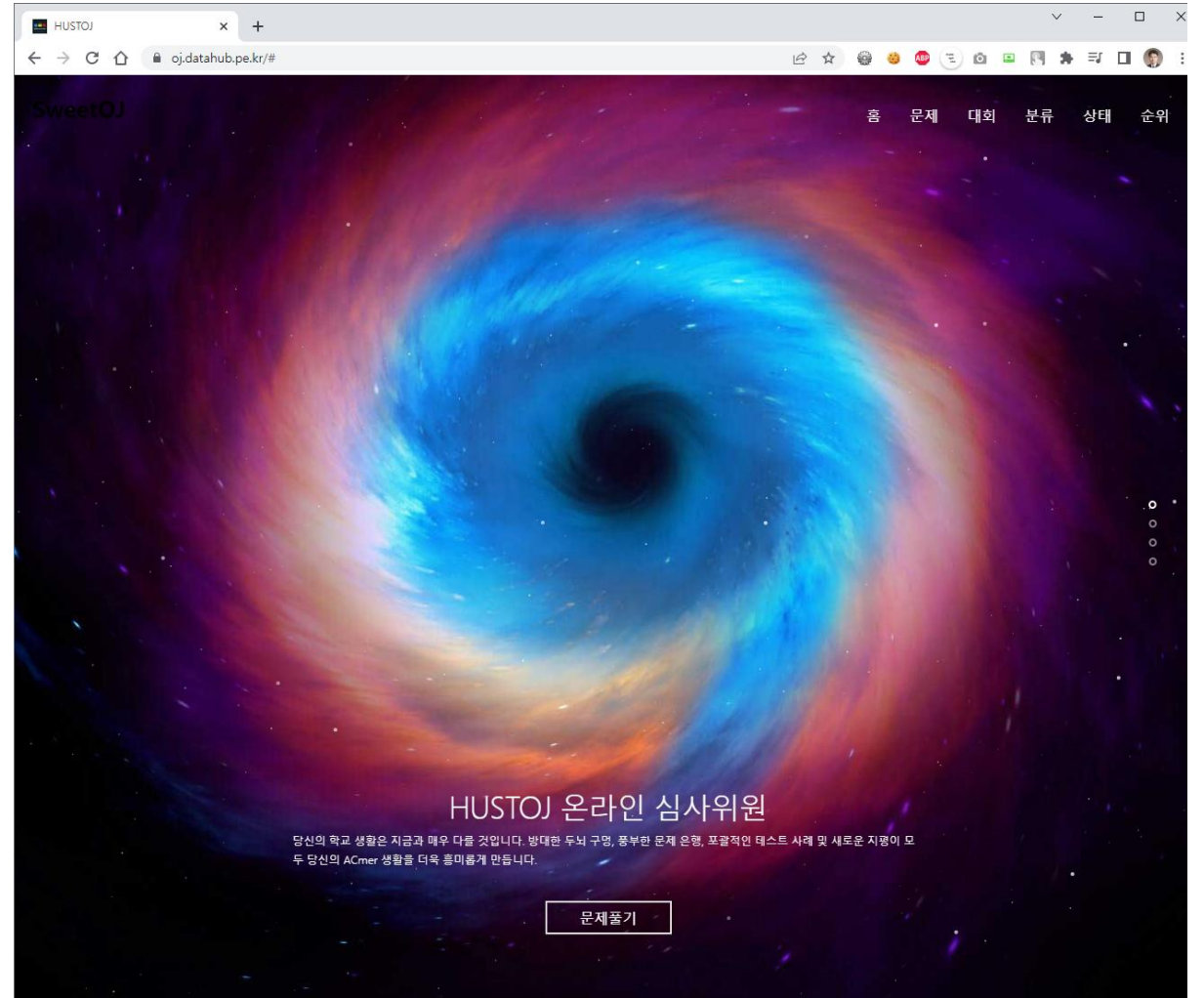
실행
결과

```
3  
1.7320508075688772
```

01 사용법

➤ 0J(온라인 저지)란?

- 프로그래밍 대회에서 프로그램들을 시험할 목적으로 만들어진 온라인 시스템이다. 대회 연습용으로 사용되기도 한다.
- 본 수업용 0J
 - <https://soj.datahub.pe.kr/>

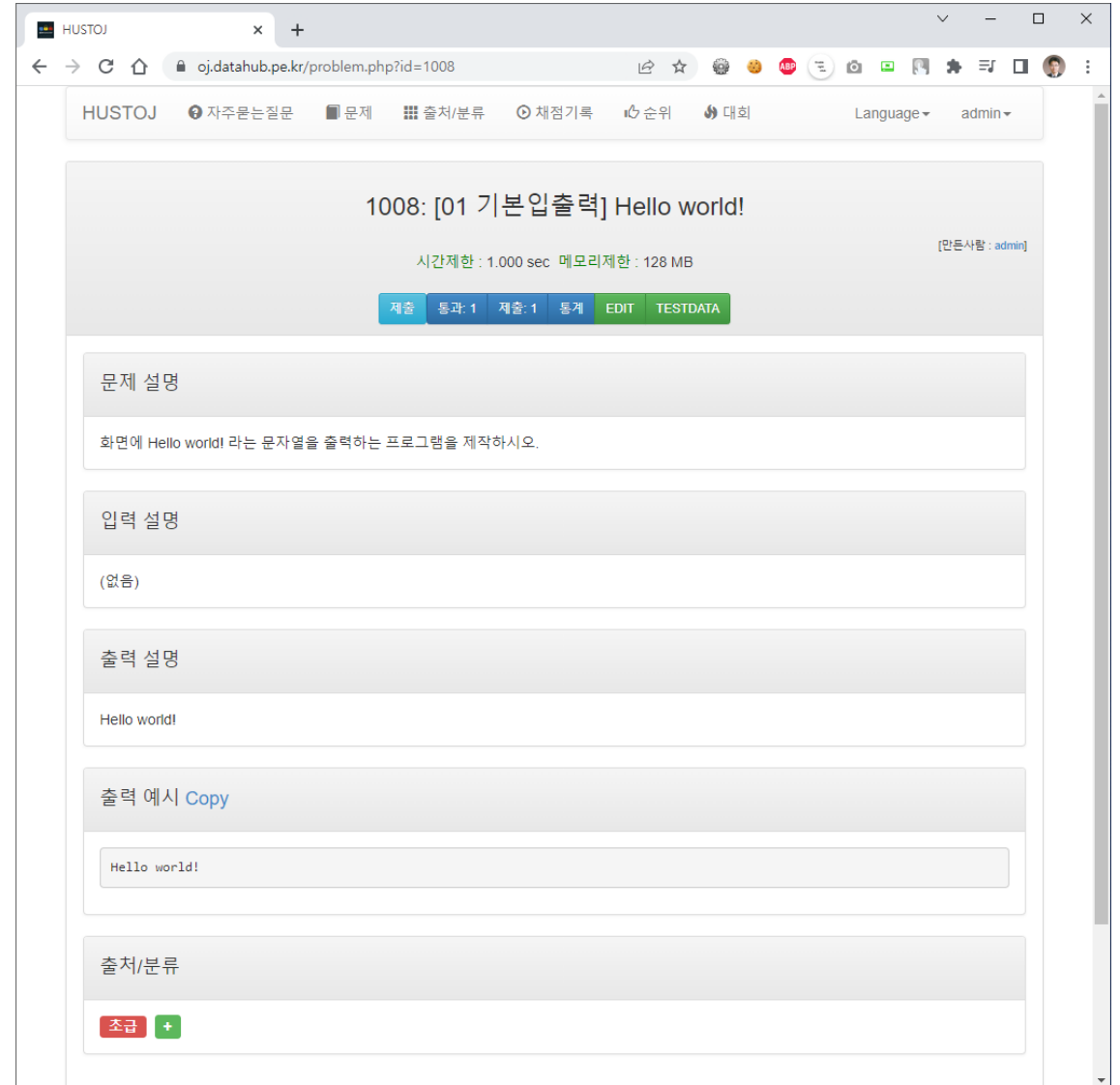


01 사용법

➤ 파일제출

- CodeBlock 등 IDE에서 프로그램 작성
- 완성된 프로그램을 01에 업로드 후
- 채점 결과 확인

- 채점 결과 종류
 - 모두 맞춤
 - 틀림 | 정확도: __%
 - 실행중 에러 | 정확도: __%
 - 컴파일 에러



제어문

➤ 선택문

- 단순 if
- if..else
- if..elif
- if..elif..elif..else

➤ 반복문

- for
 - for in range
- while

➤ 기타

- break
- continue

선택문

① if 문의 이해

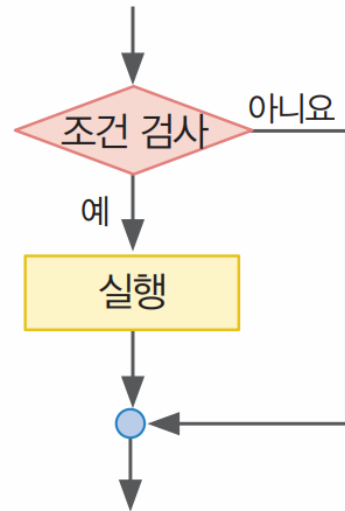
선택 구조는 주어진 조건에 따라 다른 명령을 선택하여 실행하는 경우에 사용하는 제어 구조이다. 선택 구조를 프로그래밍하기 위해서는 if 문을 사용할 수 있으며, if 문과 관련된 예약어에는 elif 문과 else 문이 있다.

유형	if 문	if ~ else 문	if ~ elif ~ else 문
형식	<pre>if 조건식: 명령어1 명령어2 명령어3</pre>	<pre>if 조건식: 명령어1 else: 명령어2 명령어3</pre>	<pre>if 조건식1: 명령어1 elif 조건식2: 명령어2 else: 명령어3</pre>
순서도	<pre> graph TD A{조건식을 만족하는가?} -- 예 --> B[명령어1] B --> C[명령어2] C --> D[명령어3] A -- 아니요 --> D </pre>	<pre> graph TD A{조건식을 만족하는가?} -- 예 --> B[명령어1] B --> C[명령어3] A -- 아니요 --> D[명령어2] D --> C </pre>	<pre> graph TD A{조건식 1을 만족하는가?} -- 예 --> B[명령어1] A -- 아니요 --> C{조건식 2를 만족하는가?} C -- 예 --> D[명령어2] C -- 아니요 --> E[명령어3] </pre>

선택문

➤ 단순 if

- 조건이 만족되면 할 일이 있을 때 사용



- ex)
 - 100점이면, '축하축하' 출력

➤ 코드

```
score = int(input('점수는? '))

if score == 100:
    print('축하축하')
print('수고했어요')
```

선택문



활동

단순 선택

작은 값 출력하기

두 개의 정수를 한 번에 한 줄씩 입력받아 작은 값을 출력하는 프로그램을 작성해 보자.

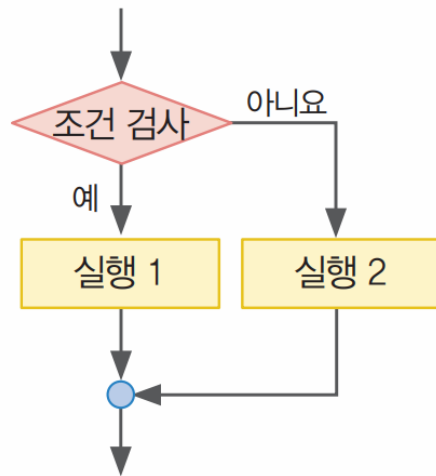
코드 작성

01
02
03
04
05
06
07
08

선택문

➤ if ... else

- 조건이 만족되면 A를 하고, 아니면 B를 해야 할 때



- ex)
 - 60점 이상 이면 '합격', 아니면
 '불합격'

➤ 코드

```
score = int(input('점수는? '))

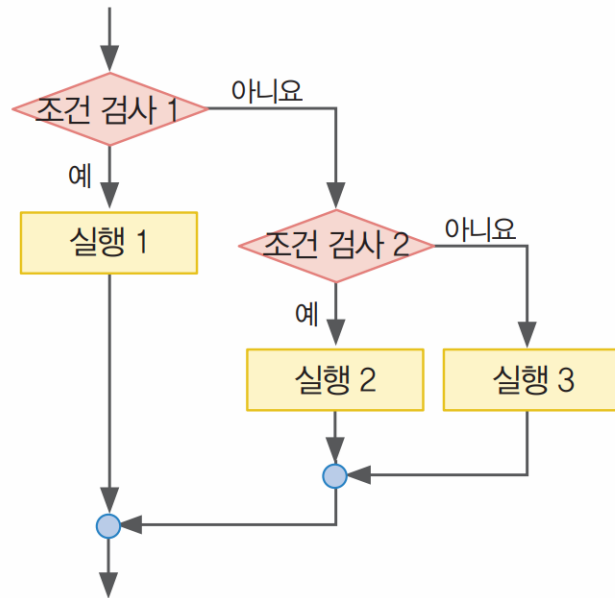
if score >= 60:
    print('합격!')
    print('축하드려요')
else:
    print('불합격!')
    print('재도전하세요')

print('조회 완료')
```

선택문

➤ if ... elif ... else

- 다중비교



- ex)

- 90점 이상: A
- 80점 이상: B
- 70점 이상: C
- 60점 이상: D
- 모두 아니면: E

➤ 코드

```
score = int(input('점수는? '))

if score >= 90:
    print('A')
elif score >= 80:
    print('B')
elif score >= 70:
    print('C')
elif score >= 60:
    print('D')
else:
    print('E')
```



if ~ elif ~ else 문을 이용한 프로그램 작성하기

지식
정보 처리

컴퓨팅
사고력



개인

제시된 기준에 따라 과일의 당도를 측정하여 등급을 분류하는 프로그램을 작성해 보자.

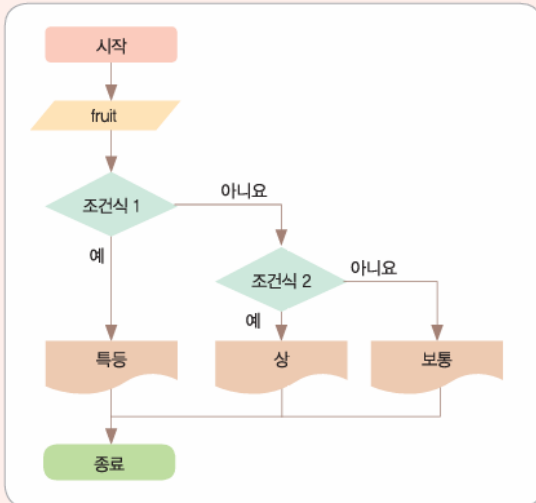
▼ 당도 등급 기준

과일의 당도	등급
11 이상	특등
9 이상 11 미만	상
9 미만	보통

▼ 입력 및 출력 예시

입력 예시	출력 예시
12	특등
7	보통
10	상

1 알고리즘을 보고 순서도의 조건식에 들어갈 내용을 적어 보자.



조건식 1

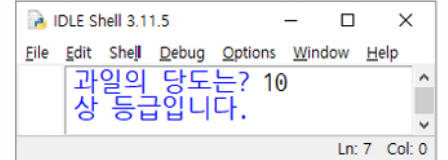
조건식 2

2 입력한 당도에 따른 등급이 출력되도록 프로그램을 완성해 보자.

소스 코드

```
fruit = int( input( '과일의 당도는? ' ) )
```

결과



선택문

➤ BMI 계산

- 체질량지수는 자신의 몸무게(kg)를 키의 제곱(m)으로 나눈 값입니다.
- 몸무게(kg 단위)와 키(cm 단위)를 입력받아 BMI를 계산하여 소수점 둘째 자리까지 출력하고,
- BMI 수치에 따른 결과를 출력하시오.
 - 18.5 미만이면 '저체중'
 - 18.5 ~ 23미만이면 '정상'
 - 23.0 ~ 25 미만이면 '과체중'
 - 25.0 이상부터는 '비만'

➤ 코드



선택문

- 필기/실기 모두 60점 이상이어야 합격

```
pilgi = int(input('필기점수는? '))
silgi = int(input('실기점수는? '))

if pilgi>=60 and silgi>=60:
    print('합격')
else:
    print('불합격입니다.')
    print('다시 도전하세요')
```

- 필기/실기 둘 중 하나만 60점 이상이면 합격

```
pilgi = int(input('필기점수는? '))
silgi = int(input('실기점수는? '))

if pilgi>=60 or silgi>=60:
    print('합격')
else:
    print('불합격입니다.')
    print('다시 도전하세요')
```

선택문

- 점수가 60점 미만이면 합격

```
pilgi = int(input('필기점수는? '))

if not(pilgi < 60):
    print('합격')
else:
    print('불합격입니다.')
    print('다시 도전하세요')
```

- 필기가 60점 미만이고, 실기도 60점 미만이면 합격

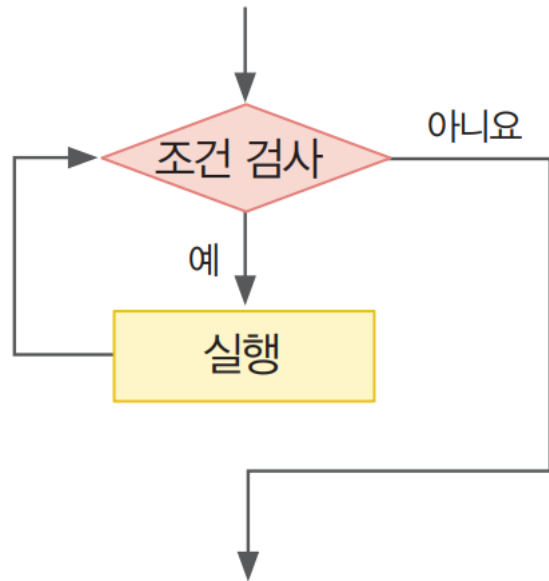
```
pilgi = int(input('필기점수는? '))
silgi = int(input('실기점수는? '))

if not(pilgi<60) and not(silgi<60):
    print('합격')
else:
    print('불합격입니다.')
    print('다시 도전하세요')
```

반복문

➤ while 반복문

- 조건이 만족되는 동안 반복하여 실행




▲ [그림 II-7] while ~ 반복 구조

➤ 형식

형식	설명
<pre>while 조건식: 실행 1 실행 2 ...</pre>	조건식의 계산 결과(논릿값)가 True인 동안 코드 블록으로 들여쓰기한 명령을 반복해서 실행하고, 조건식의 계산 결과(논릿값)가 False인 경우는 반복 실행을 중단한다.

예시	설명
<pre>n = 1 while n<=3: print(n) n = n+1</pre>	n<=3 비교 계산 결과가 True인 동안, 반복해서 n값을 출력하고, n=n+1을 실행한다.

 **하나 더 알기**

while ~ 반복 구조는 if ~ 선택 구조에서 if를 while로 바꾼 것으로 생각할 수 있다. if ~ 선택 구조는 조건식의 결과값이 True인 경우에 한 번만 실행되지만, while ~ 반복 구조는 조건식의 결과값이 True인 동안 계속 반복해서 실행된다.

반복문

➤ 형식

```
while 반복조건 :  
    반복할 문장1;  
    반복할 문장2;  
    반복할 문장3;  
    :
```

반복과 무관한 코드

➤ 작동 순서

- 반복조건 확인하여 False이면 탈출하고 아니면 반복 문장들 실행

➤ 예시

```
n = 1  
while n < 5 :  
    print(n)  
    n=n+1
```

n

5

1
2
3
4

반복문

➤ 5회 반복 방법 1

```
c = 0
while (c < 5):
    print(c)
    c=c+1
```

```
0
1
2
3
4
```

➤ 5회 반복 방법 2

```
c = 1
while (c <= 5):
    print(c)
    c=c+1
```

```
1
2
3
4
5
```

반복문



활동

단순 반복 n 부터 1까지 한 줄에 하나씩 출력하기

한 개의 정수(n)를 입력받아 그 수부터 1까지 1만큼씩 작아지는 값을 한 줄에 하나씩 출력하는 프로그램을 작성



활동

단순 반복 1부터 n 까지 한 줄로 출력하기

한 개의 정수(n)를 입력받아 1부터 그 수까지 스페이스를 사이에 두고 한 줄로 출력하는 프로그램을 작성해 보자.

코드 작성

01
02
03
04
05
06
07
08
09

반복문

➤ 퀴즈

```
num = 10    # 10부터 시작

print("Rocket lunch countdown..")
while (num > 0) : # 0보다 크면
    print(f"{num:2d}")
    num=num-1; # 1씩 감소하면서...

print(f"last num:{num}\n")
```

- 카운트 다운 숫자는 얼마까지 출력될까?
- 종료 직전 num 값은? →

➤ 결과

```
Rocket lunch countdown..
10
 9
 8
 7
 6
 5
 4
 3
 2
 1
```

0

반복문 활용

➤ 1부터 10까지 누계 구하기

$$1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10$$

sum	0	1	3	6	10	15	21	28	36	45	55
c	1	2	3	4	5	6	7	8	9	10	11

Diagram illustrating the calculation of the sum of numbers from 1 to 10. The 'sum' row shows the cumulative sum at each step, and the 'c' row shows the current number being added. Arrows indicate the addition of 'c' to 'sum' and the increment of 'c' by 1.

[초기값]

sum = 0

c = 1

```
while (c < 11):
```

```
    sum = sum + c
```

```
    c = c + 1
```

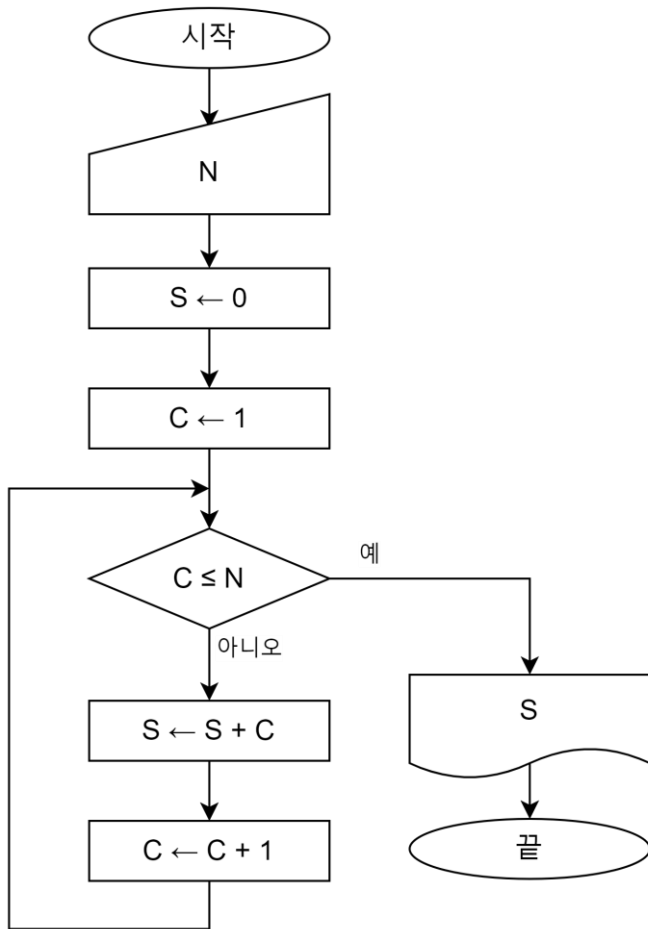
```
while (c <= 10):
```

```
    sum = sum + c
```

```
    c = c + 1
```

반복문 활용

➤ 1부터 10까지 누계 구하기



➤ 소스코드

```
sum=0
c=1

while (c <= 10) :
    sum=sum+c
    c=c+1

print(sum)
```

➤ 출력

55

STEP	sum	c
1	0	1
2	1	2
3	3	3
4	6	4
5	10	5
6	15	6
7	21	7
8	28	8
9	36	9
10	45	10
11	55	11

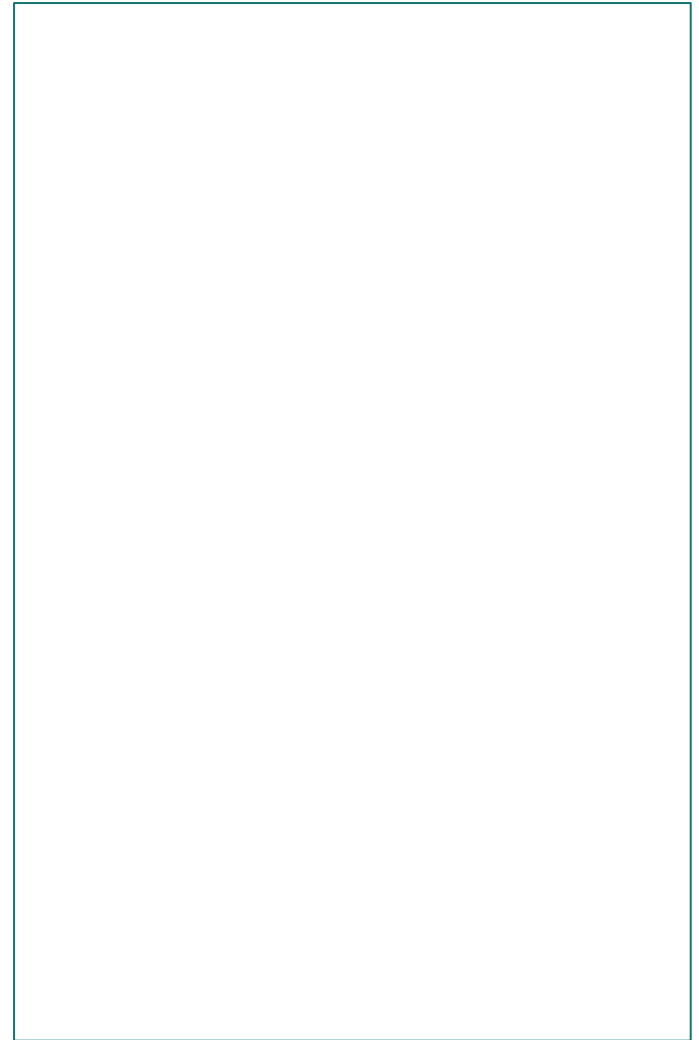
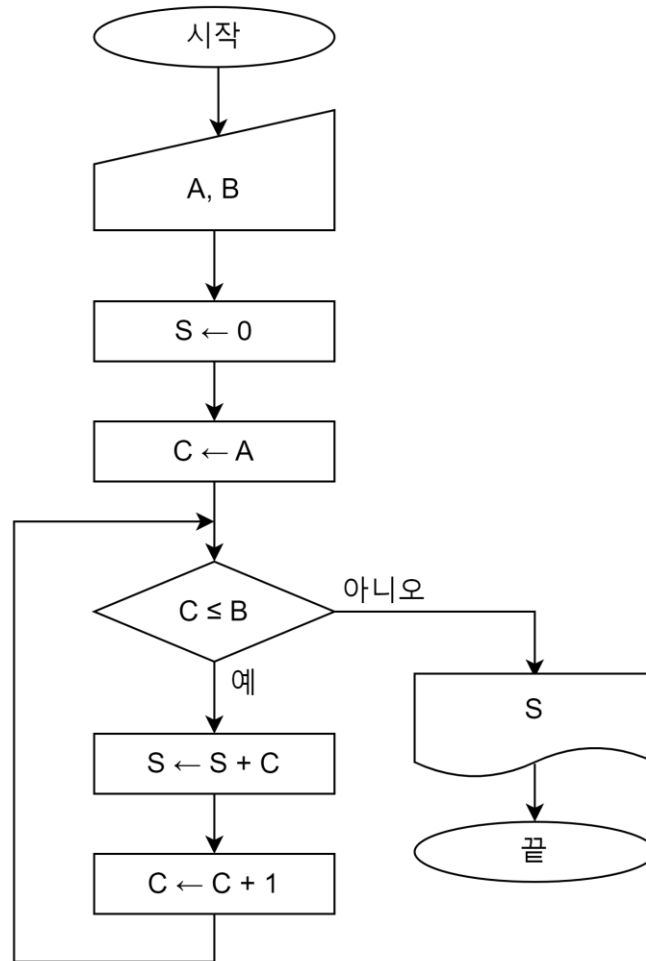
연습문제

➤ while 문을 사용하여 두 정수 a와 b를 입력 받아 a부터 b까지 누계를 구하는 프로그램을 작성하시오.

(a < b 라고 가정)

■ 예

1
10
55



소인수로 분해하기

➤ 문제

양의 정수 한 개가 입력되었을 때, 그 수를 소인수로 분해하여 출력하는 프로그램을 작성하시오.

➤ 입력

양의 정수 한 개가 입력된다. (2이상)

➤ 출력

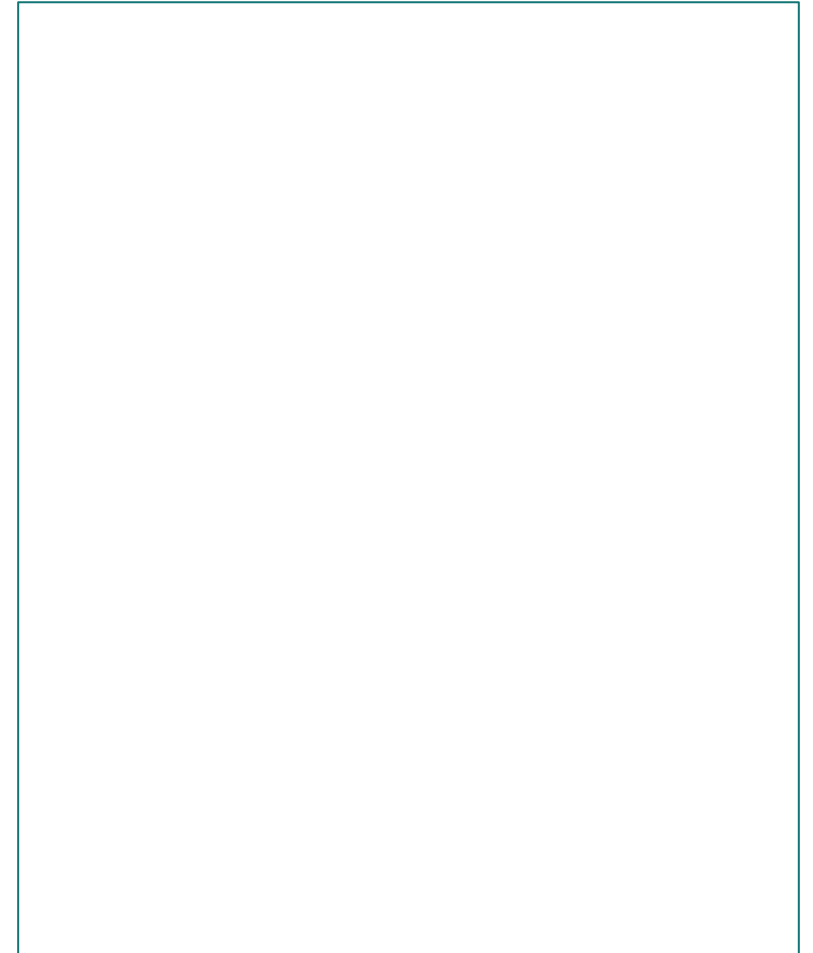
그 수를 소인수로 분해하여 오름차순으로 출력한다.

➤ 입력과 출력의 예

입력 예	출력 예
420	2 2 3 5 7

➤ 프로그램

```
2 | 420
2 | 210
3 | 105
5 | 35
7 | 7
  | 1
```



중첩된 while

➤ 중첩된 while

```
while(반복조건1) :  
    외부반복문1  
    while(반복조건2) :  
        내부반복문1  
        내부반복문2  
    외부반복문2
```

- 들여쓰기를 사용하여 반복 내용의 시작과 끝을 명확히 하자!

➤ 00:00 부터 ~ 05:59 까지 시간 출력

```
print("clock time")  
  
hour=0  
while(hour < 6) :  
    min = 0;  
    while(min < 60) :  
        print(f"{hour:02d}:{min:02d} ", end='');  
        min=min+1;  
    print()  
    hour=hour+1
```

for 반복문

① for 문

for 문은 반복을 어떻게 설정할 것인지에 따라 2가지 유형으로 나눌 수 있다.

■ for 문의 유형

유형 1	유형 2
<pre>for 변수 in range(N): 명령어1 명령어2</pre>	<pre>for 변수 in 리스트/튜플/문자열: 명령어1 명령어2</pre>

range() 함수

정수 수열을 만들어 주는 함수로 주로 for 문과 함께 사용된다.

range(N) 함수를 사용하면 0부터 (N-1)까지의 정수가 하나씩 변수에 대입된다. 만약 연속형 변수를 사용하면 첫 번째 변수부터 마지막 변수까지 차례로 대입된다. 그리고 대입된 변수마다 명령어 1, 2를 실행하게 되며, 반복하는 횟수는 대입되는 횟수와 동일하다.

반복문은 조건문과 마찬가지로 for 문 끝이 콜론(:)으로 끝나야 하며, 반복할 명령어와 반복하지 않을 명령어를 들여쓰기를 통해 구분해야 한다.

for 반복문



하나 더 알기

while ~ 반복 구조는 주어진 조건이 True인 동안에 계속하여 반복 실행되지만, for ~ 반복 구조는 주어진 수열이나 리스트 안에 들어 있는 데이터의 개수만큼만 반복 실행된다.

range() 함수는 일정한 차이값을 가진 정수 수열을 만든다.

- range(n)은 0, 1, 2, 3, ..., n-2, n-1의 정수 수열을 뜻한다.

예 range(3) ➔ 0, 1, 2의 정수 수열

- range(a, b)는 a, a+1, a+2, ..., b-2, b-1의 정수 수열을 뜻한다.

예 range(1, 4) ➔ 1, 2, 3의 정수 수열

- range(a, b, k)는 a, a+k, a+2*k, ..., a 이상 b보다 작은 차이값이 k인 정수 수열을 뜻한다.

예 range(1, 10, 2) ➔ 1, 3, 5, 7, 9의 정수 수열

range(5, -1, -1) ➔ 5, 4, 3, 2, 1, 0의 정수 수열

range 수열

range(stop)

range(start, stop)

range(start, stop, step)

start: 시작값(포함)

stop: 중단값(불포함)

step: 증감값

for ... in range() 예시

1) 기본편: range(start)

```
# 0부터 4까지 총 5번 반복합니다.
```

```
for i in range(5):  
    print("현재 숫자:", i)
```

```
# 출력 결과:  
# 현재 숫자: 0  
# 현재 숫자: 1  
# 현재 숫자: 2  
# 현재 숫자: 3  
# 현재 숫자: 4
```

2) 응용편: range(start, stop)

```
# 1부터 5까지 반복합니다.  
(6은 포함되지 않음)
```

```
for i in range(1, 6):  
    print("번호:", i)
```

```
# 출력 결과:  
# 번호: 1  
# 번호: 2  
# 번호: 3  
# 번호: 4  
# 번호: 5
```

for ... in range() 예시

3) 심화편: range(start, stop, step)

```
# 2부터 시작해서 10까지 2씩 증가합니다.  
(짝수 출력)
```

```
for i in range(2, 11, 2):  
    print("짝수:", i)
```

```
# 출력 결과:
```

```
# 짝수: 2
```

```
# 짝수: 4
```

```
# 짝수: 6
```

```
# 짝수: 8
```

```
# 짝수: 10
```

4) 역순편

```
# 5부터 1까지 1씩 감소하며 반복합니다.  
(0은 포함되지 않음)
```

```
for i in range(5, 0, -1):  
    print(i, "초 전...")  
print("발사!")
```

```
# 출력 결과:
```

```
# 5 초 전...
```

```
# 4 초 전...
```

```
# 3 초 전...
```

```
# 2 초 전...
```

```
# 1 초 전...
```

```
# 발사!
```

for ... in range() 예시

5) 활용편

```
total = 0
# 1부터 10까지의 숫자를 total 변수에 계속 누적
for i in range(1, 11):
    total = total + i

print("1부터 10까지의 합:", total)

# 출력 결과:
# 1부터 10까지의 합: 55
```

for ... in 리스트/튜플/문자열 예시

1) 기본편

```
# 과일 바구니(리스트)에서 과일을 하나씩 인출
fruits = ["사과", "바나나", "포도", "귤"]

for fruit in fruits:
    print("내가 좋아하는 과일:", fruit)

# 출력 결과:
# 내가 좋아하는 과일: 사과
# 내가 좋아하는 과일: 바나나
# 내가 좋아하는 과일: 포도
# 내가 좋아하는 과일: 귤
```

2) 응용편

```
# 학생들의 시험 점수 리스트입니다.
scores = [85, 90, 78, 100, 88]
total = 0

# 점수를 하나씩 꺼내서 total에 더합니다.
for score in scores:
    total = total + score

print("총점:", total)
print("평균:", total / 5)

# 출력 결과:
# 총점: 441
# 평균: 88.2
```

for ... in 리스트/튜플/문자열 예시

3) 문자열 순회

```
# 단어의 스펠링을 한 글자씩 출력합니다.  
word = "PYTHON"  
  
for letter in word:  
    print(letter)  
  
# 출력 결과:  
# P  
# Y  
# T  
# H  
# O  
# N
```

4) 심화편: enumerate()

```
# 출석부 리스트입니다.  
students = ["유재석", "김종국", "송지효"]  
  
# enumerate를 사용하면 (인덱스, 데이터)  
# 두 개를 동시에 꺼내줍니다.  
# 번호는 0부터 시작하므로 1을 더해줍니다.  
for i, name in enumerate(students):  
    print(f"{i + 1}번 학생: {name}")  
  
# 출력 결과:  
# 1번 학생: 유재석  
# 2번 학생: 김종국  
# 3번 학생: 송지효
```

for 반복문



활동

단순 반복

정수 데이터 한 줄로 입력받아 여러 줄로 출력하기

스페이스를 사이에 두고 한 줄로 입력받은 각각의 정숫값을 리스트에 저장한 후, 한 줄에 하나씩 출력하는 프로그램을 작성해 보자.

코드 작성

01
02
03
04
05
06



활동

단순 반복

정수 데이터 여러 줄로 입력받아 순서 바꾸어 한 줄로 출력하기

데이터 개수와 그 개수만큼의 정숫값을 한 줄에 하나씩 입력받아 리스트에 저장한 후, 순서를 바꾸어 마지막 값부터 처음 값까지 스페이스를 사이에 두고 한 줄로 출력하는 프로그램을 작성해 보자.

코드 작성

01
02
03
04
05

3의 배수 게임

➤ 문제

3의 배수 게임을 하던 정올이는 3의 배수 게임에서 작은 실수를 계속해서 벌칙을 받게 되었다.

3의 배수 게임의 왕이 되기 위한 수련 프로그램을 작성해 보자.

** 3의 배수 게임이란?

여러 사람이 순서를 정해 순서대로 수를 부르는 게임이다.

만약 3의 배수를 불러야 하는 상황이라면, 그 수 대신 "박수"를 친다.

➤ 입력

첫 줄에 하나의 정수 n 이 입력된다.
(n 은 50미만의 자연수이다)

➤ 출력

1부터 n 까지 순서대로 공백을 두고 수를 출력하는데, 3의 배수(3, 6, 9 ...)인 경우 수 대신 영문 대문자 X 를 출력한다.

➤ 입력과 출력의 예

입력 예	출력 예
7	1 2 X 4 5 X 7

약수의 합 구하기

➤ 문제

한 정수 n 을 입력 받아서 n 의 모든 약수의 합을 구하는 프로그램을 작성하시오.

예를 들어 10의 약수는 1, 2, 5, 10이므로 이 값들의 합인 18이 10의 약수의 합이 된다.

➤ 입력

첫번째 줄에 정수 n 이 입력된다.
(단, $1 \leq n \leq 100,000$)

➤ 출력

n 의 약수의 합을 출력한다

➤ 입력과 출력의 예

입력 예	출력 예
5	6

입력 예	출력 예
10	18

➤ 고찰

n 의 약수들을 어떻게 알아낼 수 있을까?

약수의 합 구하기

➤ 문제

한 정수 n 을 입력 받아서 n 의 모든 약수의 합을 구하는 프로그램을 작성하시오.

예를 들어 10의 약수는 1, 2, 5, 10이므로 이 값들의 합인 18이 10의 약수의 합이 된다.

➤ 입력

첫번째 줄에 정수 n 이 입력된다.
(단, $1 \leq n \leq 100,000$)

➤ 출력

n 의 약수의 합을 출력한다

➤ 코드

```
n = int(input())
ans = 0;

print(ans)
```

공약수 찾기

➤ 문제

- 입력된 두 자연수의 공약수를 모두 출력하는 프로그램을 작성하시오.

➤ 입력

- 첫 번째 줄에 두 자연수 a와 b가 공백으로 분리되어 입력된다.
($1 \leq a, b \leq 2,100,000,000$)

➤ 출력

- a와 b의 공약수를 작은 수부터 큰 수 순서로 공백으로 분리하여 출력한다.

➤ 예시

```
8 24
1 2 4 8
```

➤ 프로그램

```
a, b = map(int, input().split())
```

반복문 - 중첩된 for

1) 한 학급 1번 부터 30번까지 출력한다.

```
for n in range(1, 31):  
    print(f"{n:4d} ", end="")
```

2) 한 학급 1번 부터 30번 까지 열 개 학급에 대하여 출력한다.

```
for c in range(1, 11):  
    print(f"{c}반")  
    for n in range(1, 31):  
        print(f"{n:4d} ", end="")  
    print() # 줄내림
```

3) 세 개 학년에 대하여 출력한다.

반복문 - 중첩된 for

➤ 중첩된 for 문을 이용하여 구구단 출력하기

```
for d in range(1, 6):      # 1단부터 5단까지
    print(f"----{d}단----")
    for x in range(1, 10): # x1 부터 x9 까지
        print(f"{d} * {x} = {d*x}")
    print()
```

```
----1단----
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
1 * 5 = 5
1 * 6 = 6
1 * 7 = 7
1 * 8 = 8
1 * 9 = 9

----2단----
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18

----3단----
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
```

반복문 - 중첩된 for

➤ 연습문제

삼각형의 밑변 길이 정수 a 를 입력 받아 반복문을 이용하여 아래 그림과 같은 직각 삼각형 모양을 출력하는 프로그램을 작성 하시오.

*	1개
**	2개
***	3개
****	4개
*****	5개
*****	:
*****	a개

➤ 정답



반복문 - 중첩된 for

➤ 연습문제

삼각형의 밑변 길이 정수 a 를 입력 받아 중첩된 반복문을 이용하여 아래 그림과 같은 직각 삼각형 모양을 출력하는 프로그램을 작성 하시오.

* ** *** **** ***** ***** *****	공백?+별n개=a개 ∴ ? = a-n
---	-------------------------

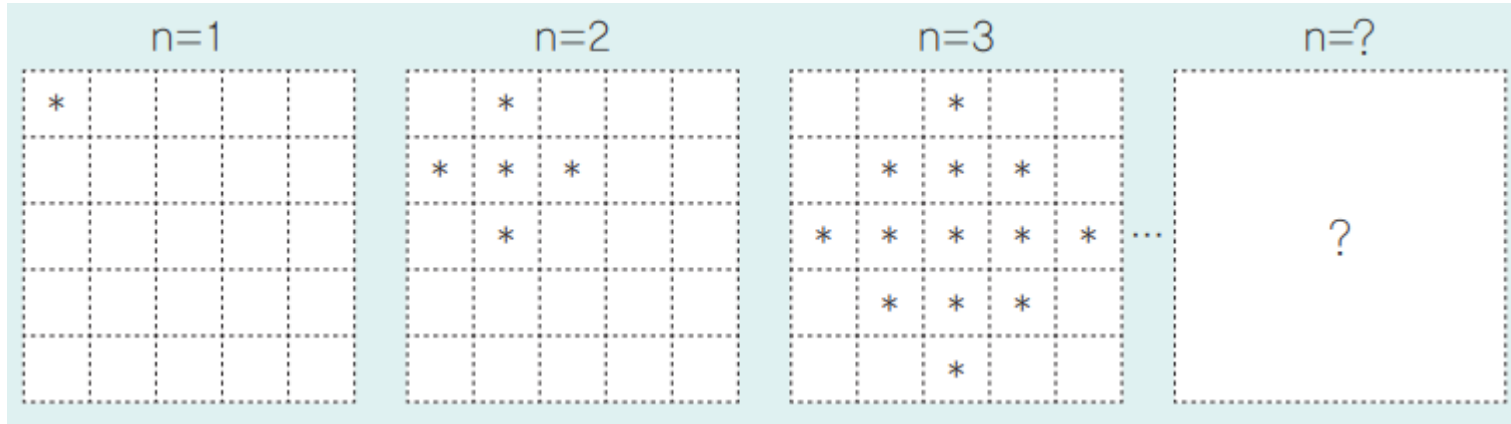
➤ 정답



중첩된 for문 활용

➤ 연습문제

삼각형의 밑변 길이 정수 a 를 입력 받아 중첩된 반복문을 이용하여 아래 그림과 같은 직각 삼각형 모양을 출력하는 프로그램을 작성 하시오.



➤ 입력설명

첫 번째 줄에 자연수 n 이 입력된다. ($1 \leq n \leq 15$)

중첩된 for문 활용

		1	2	3	4	5	6	7	8	9	
	j	1				*					a=4
	j	2			*	*	*				공백개수=a-j
	j	3		*	*	*	*	*			별개수=2*j-1
	j	4	*	*	*	*	*	*	*		
	j	3		*	*	*	*	*			
	j	2			*	*	*				
	j	1				*					

제어문 - break

➤ break 문

- for, while 문의 영역을 빠져 나오기 위해 사용
- 가장 가까운 루프를 벗어난다.

➤ 사용 예

- $1+2+3+\dots+n$ 의 합이 처음으로 100이상이 될 때, 그 때의 합과 n을 구하는 프로그램을 작성하시오.

```
sum = 0
n = 0

while True:
    n = n + 1
    sum = sum + n
    if(sum>=100):
        break

print(f'{n}까지 누적하면 {sum}입니다.')
```

소수 판별

➤ 문제

3 이상의 자연수(n)가 입력되었을 때,
소수 여부를 판별하는 프로그램을
작성해 보자.

($3 \leq n \leq 1,000,000$)

➤ 입력과 출력 예시

입력 예	출력 예
41	prime
111	composite

7	1
	2
	3
	4
	5
	6
	7

➤ 프로그램

```
n = int(input())

isPrime = True
for d in range(2, n):
    if(n%d == 0):
```

제어문 - continue

➤ continue 문

- 반복문 내에서 사용되며, 남겨진 반복내용을 중단하고 다음 반복을 시작한다.

➤ 사용 예

- 1 부터 20 까지의 정수 중에서 홀수만을 출력하시오. (for, continue 문을 사용할 것.)

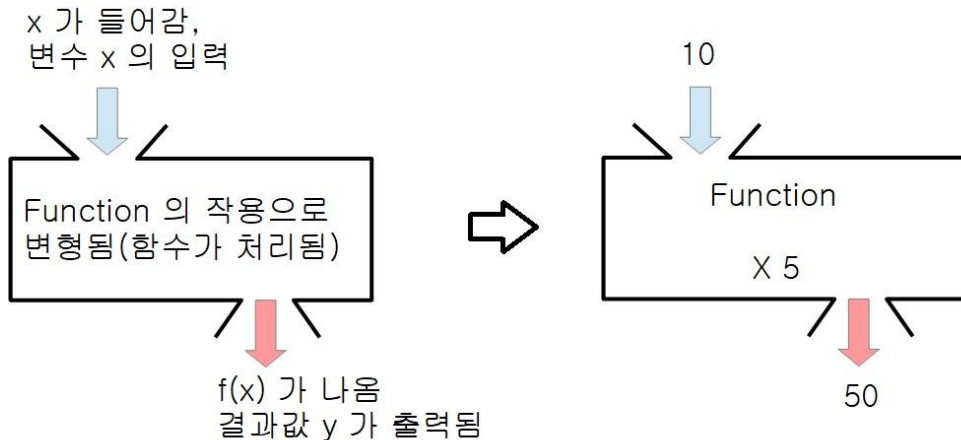
➤ 프로그램

```
for n in range(1, 21):  
    if(n%2 == 0):  
        continue  
  
    print(n)
```

함수

➤ 함수(function)란?

- 특정한 처리·기능을 수행하는 코드를 하나로 묶어 둔 것.
- 특정 인자를 받아 결과값을 반환.
- ‘식별자()’ 의 형태



➤ 함수 사용의 효과

- 코드들을 기능 단위로 묶을 수 있기 때문에 프로그램을 이해하고 만들기 쉽게 한다.

➤ 별명

- 서브루틴(subroutine)
- 프로시저(procedure)
- 메서드(method)

함수

➤ 함수의 종류

구분	설명	사용 방법
내장(built-in) 함수	프로그래밍 언어 자체에서 기본으로 제공되는 함수	특별한 준비 작업 없이 바로 사용 가능
라이브러리(library) 함수	라이브러리 파일을 통하여 제공되는 함수	함수들과 여러 가지 정의된 값을 함께 작성해 둔 관련 라이브러리 파일을 먼저 참조한 후 사용 가능
사용자 정의(user-defined) 함수	필요에 따라 사용자가 직접 정의하여 사용하는 함수	사용하기 전에 먼저 정의한 후 사용 가능

함수

➤ 내장 함수

- The Python interpreter has a number of functions and types built into it that are always available. They are listed here in alphabetical order.
- [Built-in Functions – Python 3.14.3 documentation](#)

Built-in Functions			
A abs() aiter() all() anext() any() ascii()	E enumerate() eval() exec()	L len() list() locals()	R range() repr() reversed() round()
B bin() bool() breakpoint() bytearray() bytes()	F filter() float() format() frozenset()	M map() max() memoryview() min()	S set() setattr() slice() sorted() staticmethod() str() sum() super()
C callable() chr() classmethod() compile() complex()	G getattr() globals()	N next()	T tuple() type()
D delattr() dict() dir() divmod()	H hasattr() hash() help() hex()	O object() oct() open() ord()	V vars()
	I id() input() int() isinstance() issubclass() iter()	P pow() print() property()	Z zip() __import__()

함수

➤ 라이브러리 함수

- 파이썬에서는 모듈(module)로 불리는 라이브러리 파일을 참조해서 그 안에 미리 정의되어 있는 함수와 상숫값을 사용할 수 있다.
- 여러 가지 수학 계산과 관련한 함수와 상숫값이 저장된 파이썬 표준 라이브러리 모듈은 math이다.

▪

▼ [표 II-10] 파이썬 모듈 참조 명령

명령 형식	예시	의미
import 모듈명	import math	math 모듈 내용 참조
import 모듈명 as 약어	import math as mt	math 모듈을 약어 mt로 참조
from 모듈명 import 정의	from math import *	math 모듈의 모든(*) 정의 참조

함수

➤ 라이브러리 함수 사용 예시

코드

```
01 import math
02
03 a, b = map(int, input().split())
04 print(math.gcd(a, b))
```

설명

```
01 math 모듈의 내용을 참조하도록 한다.
03 스페이스를 사이에 두고 2개의 정숫값을 한 줄로 입력받아 각각 변수 a,
   b에 저장한다.
04 math 모듈에 있는 gcd(a, b)에 의하여 얻은 최대 공약수 값을 출력한다.
```

입력

192 72

실행
결과

24

`input()`, `map()`은 내장 함수이고, `split()`은 문자열에 사용할 수 있는 메소드 함수이다. 또한 `gcd()`는 `math` 모듈을 참조하여 사용하는 라이브러리 함수이다. `math` 모듈을 `mt`라는 약어로 줄여서 참조하면, 다음과 같이 작성할 수 있다.

```
import math as mt
a, b = map(int, input().split())
print(mt.gcd(a, b))
```

`math` 모듈의 모든 정의를 참조하면, 다음과 같이 작성할 수도 있다.

```
from math import *
a, b = map(int, input().split())
print(gcd(a, b))
```

함수

➤ 라이브러리 함수



활동

라이브러리 함수

제곱근 출력하기

한 개의 정수를 입력받은 후, 그 수의 양의 제곱근을 출력하는 프로그램을 라이브러리 함수 `sqrt()`를 사용해 작성해 보자.

코드 작성

01
02
03
04

*`sqrt()` 함수에 값을 전달하면, 제곱근(square root)의 값을 계산해 준다.

함수

➤ 사용자 정의 함수

- 함수를 정의해서 사용하면, 복잡한 프로그램을 작은 단위의 작업들로 분해할 수 있다.
- 또한, 작성 코드를 편리하게 재사용할 수 있으므로 효율적으로 프로그램을 작성할 수 있을 뿐만 아니라 프로그램 실행 과정을 한층 쉽게 이해할 수 있다.

➤ 형식

형식	
def 함수 이름():	함수 정의
	실행 내용
:	실행 내용
	:
함수 이름()	함수 호출

예시	
def f():	함수 f() 정의
print("Hello")	Hello 문자열 출력
f()	함수 f() 호출

함수

➤ 함수의 네 가지 형태

구분		매개 변수의 여부	
		매개 변수 없음	매개 변수 있음
반환값 여부	반환값 없음	<pre>def f(): #정의 print("Hello")</pre>	<pre>def f(n): #매개 변수 n print(n) #n값 출력</pre>
		<pre>f() #호출</pre>	<pre>f(3) #정숫값 3 전달</pre>
	실행 결과		
	Hello	3	
반환값 있음	반환값 있음	<pre>def f(): return 3 #정숫값 3 반환</pre>	<pre>def f(a, b): #매개 변수 a, b return a+b #a+b값 반환</pre>
		<pre>print(f()) #()값 출력</pre>	<pre>print(f(3, 5)) #f(3, 5)값 출력</pre>
	실행 결과		
	3	8	

함수



함수 설계 3.141592 반환하기

실숫값 3.141592를 반환하는 함수 $g()$ 를 정의해 보자.

코드 작성

01

02



함수 설계 부호 바꾸어 출력하기

정숫값 한 개를 전달받아, 부호를 반대로 바꾼 값을 출력하는 함수 $g()$ 를 설계해 보자.

코드 작성

01

02

*참고: $-n$ 은 n 에 저장된 값의 부호를 바꾼 값을 의미한다.

함수

➤ 2개의 값을 전달받아 더한 값을 반환하는 함수

코드

```
01 def f(a, b):  
02     return a+b  
03  
04 print(f(3, 5))
```

설명

01~02 함수 f(a, b) 정의, 매개 변수 a, b
02 a+b를 계산한 결과값을 가져다 놓으며 복귀
04 f(a, b)에 3, 5를 전달하고 반환된 f(3, 5)값을 출력

실행
결과

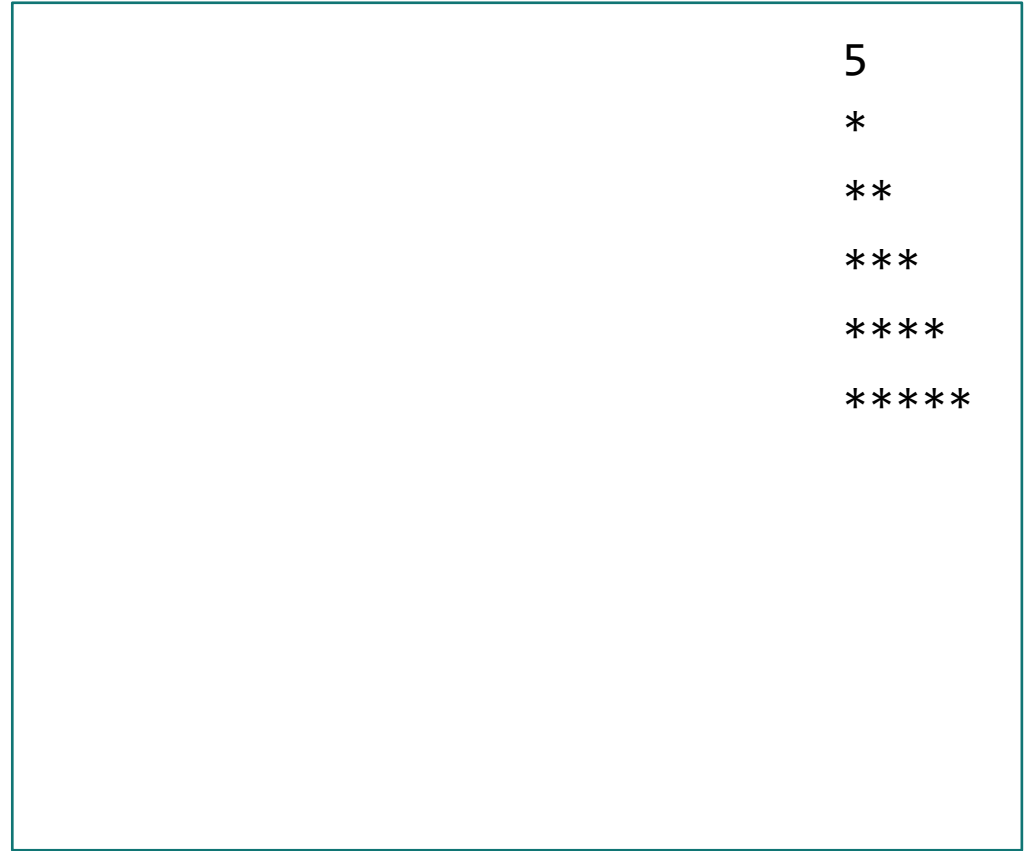
8

함수 만들기 연습

- 정수 k 를 넘겨받아 별(*)을 k 개 출력하는 함수 `kstars(k)`



- 왼쪽의 `kstars(k)`를 이용하여 *로 삼각형 그리기



함수 만들기 연습

- 두 실수 a , b 값을 받아 두 수의 차이(절대값)를 반환하는 함수

```
def diff(a, b):
```

- 두 자연수 a 와 b 를 입력 받아 a^b 를 계산하는 함수

```
def pow(a, b):
```

함수 만들기 연습

➤ 두 정수 a , b 값을 받아 큰 수를 반환하는 max 함수

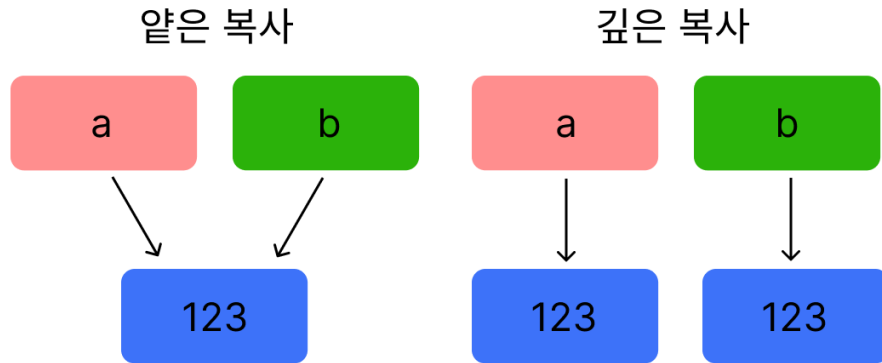


➤ 두 정수 a , b 값을 받아 작은 수를 반환하는 min 함수



얕은 복사 vs 깊은 복사

➤ 차이점



- 리스트는 대입연산자를 사용하면 얕은 복사가 이루어짐.
- 얕은 복사의 경우 사본에 가하는 변화가 원본에도 영향을 끼침.

➤ 예시

```
from copy import deepcopy

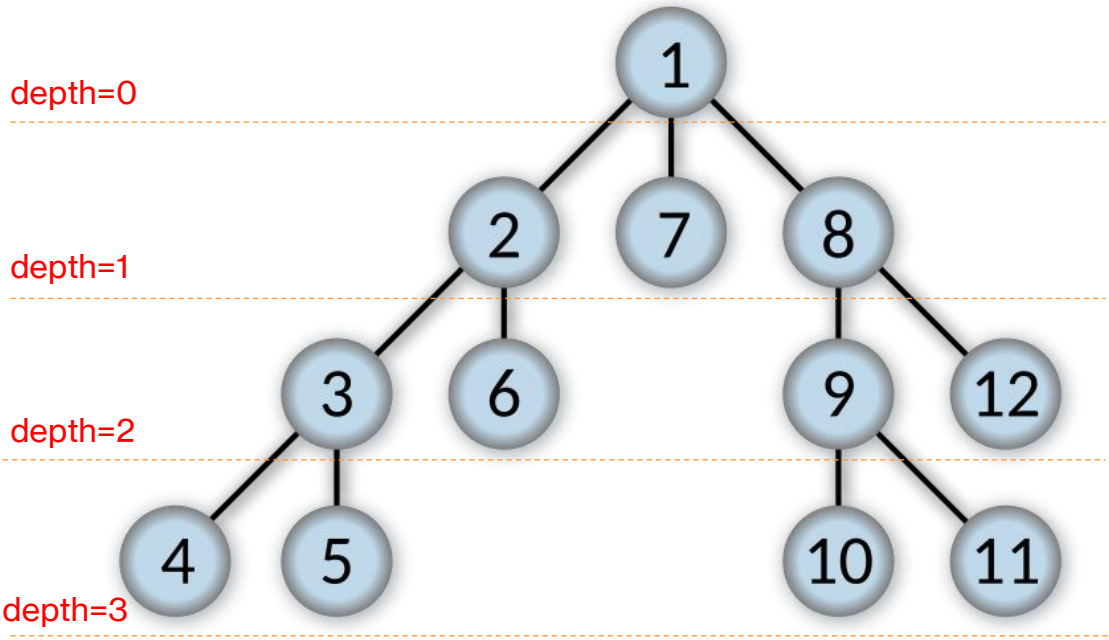
la = [1,2,3,4,5]
lb = la
print('la: ', la)
print('lb: ', lb, end='\n\n')

# lb를 수정하면 la까지 바뀜
lb[0] = 6
print('la: ', la)
print('lb: ', lb, end='\n\n')

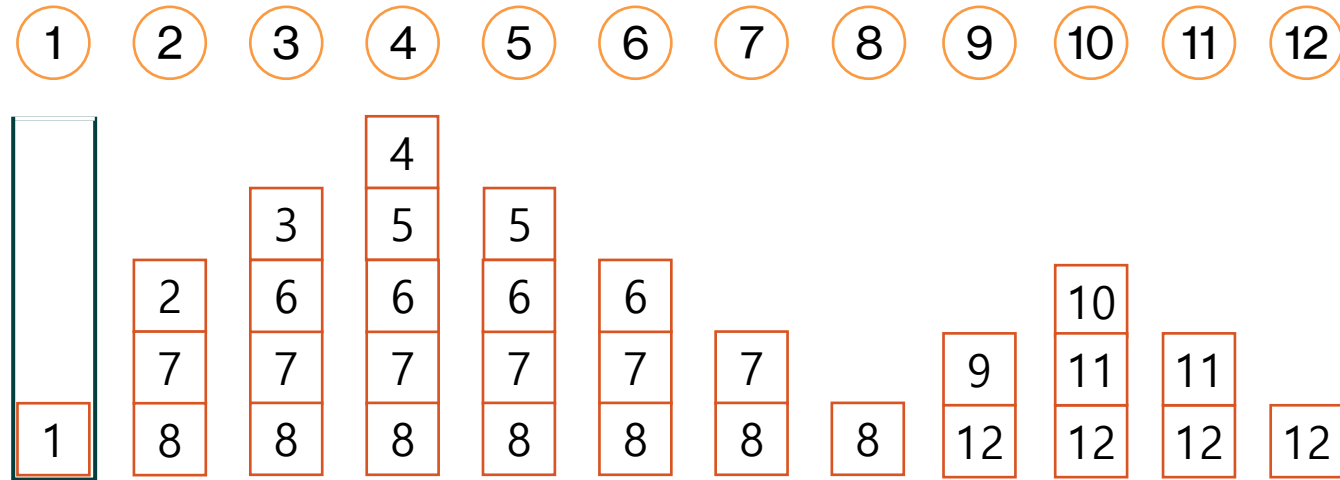
# lb를 수정해도 la는 영향없음.
la = [1,2,3,4,5]
lb = deepcopy(la)
lb[0] = 6
print('la: ', la)
print('lb: ', lb)
```

트리구조의 깊이우선탐색(DFS)

➤ 트리 구조



➤ 스택을 이용한 DFS 순회



- dfs(1)
- dfs(2), dfs(7), dfs(8)
- dfs(3), dfs(6), dfs(7), dfs(8)
- dfs(4), dfs(5), dfs(6), dfs(7), dfs(8)
- dfs(5), dfs(6), dfs(7), dfs(8)

계속 앞에 끼어든다

트리구조의 깊이우선탐색(DFS)

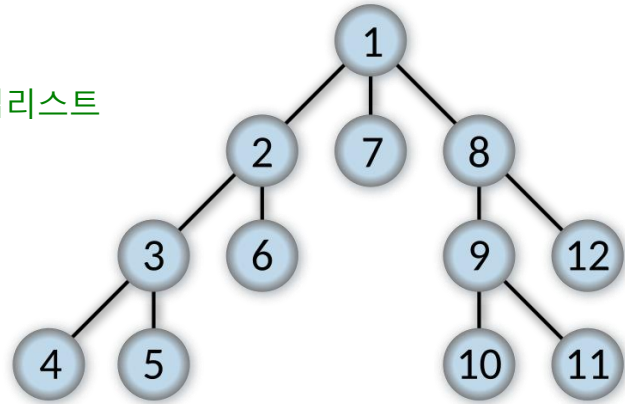
```
# stack 으로 구현하는 DFS
from copy import deepcopy

adjl = [] # 트리구조를 저장하는 인접리스트
for i in range(0,13):
    adjl.append(None)
adjl[1] = [2,7,8]
adjl[2] = [3,6]
adjl[3] = [4,5]
adjl[8] = [9,12]
adjl[9] = [10,11]
print(adjl)

stack = [1] # 🔦 탐색할 노드를 저장하는 스택 리스트 (초기 상태 삽입)
while stack: # 스택에 탐색할 노드가 남아있는 동안 반복
    print('stack: ', stack)
    # 🔦 스택의 맨 마지막(가장 깊은 곳) 노드를 꺼내어 현재 상태로 지정 (pop)
    currentnode = stack.pop()
    print(currentnode)

    # 자식 노드(다음 이동 가능한 상태) 생성
    if(adjl[currentnode] is not None):
        bnodes = reversed(deepcopy(adjl[currentnode]))

        # 🔦 자식 노드를 스택에 추가
        for child in bnodes:
            # 이미 방문했거나 스택에 대기 중인 상태는 중복 탐색하지 않음
            if child not in stack:
                stack.append(child)
```

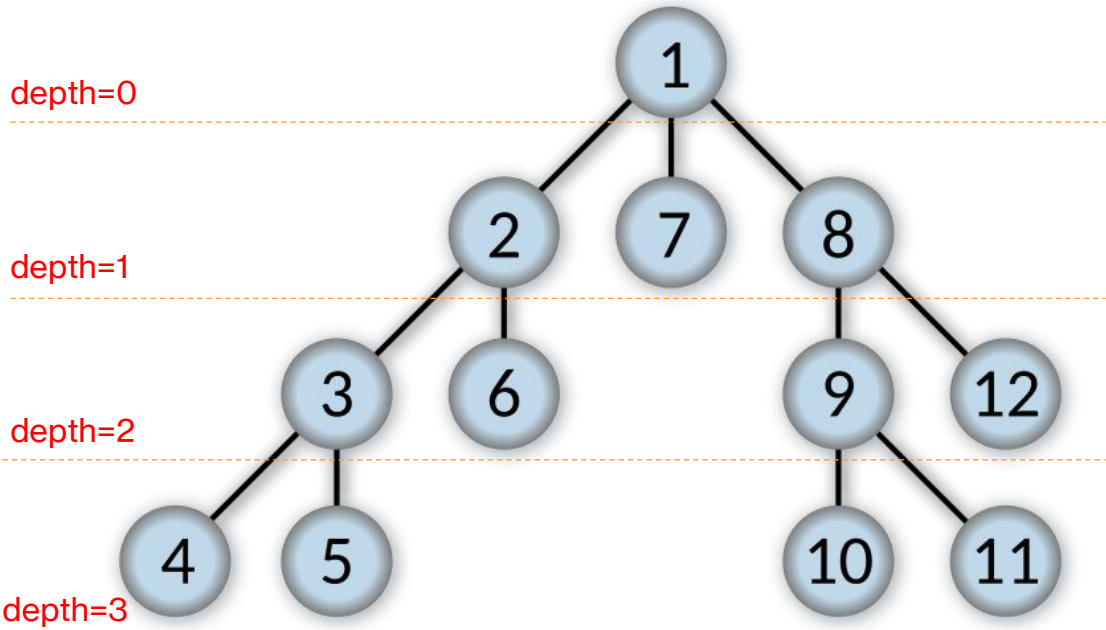


```
[None, [2, 7, 8], [3, 6], [4, 5], None, None, None,
None, [9, 12], [10, 11], None, None, None]
```

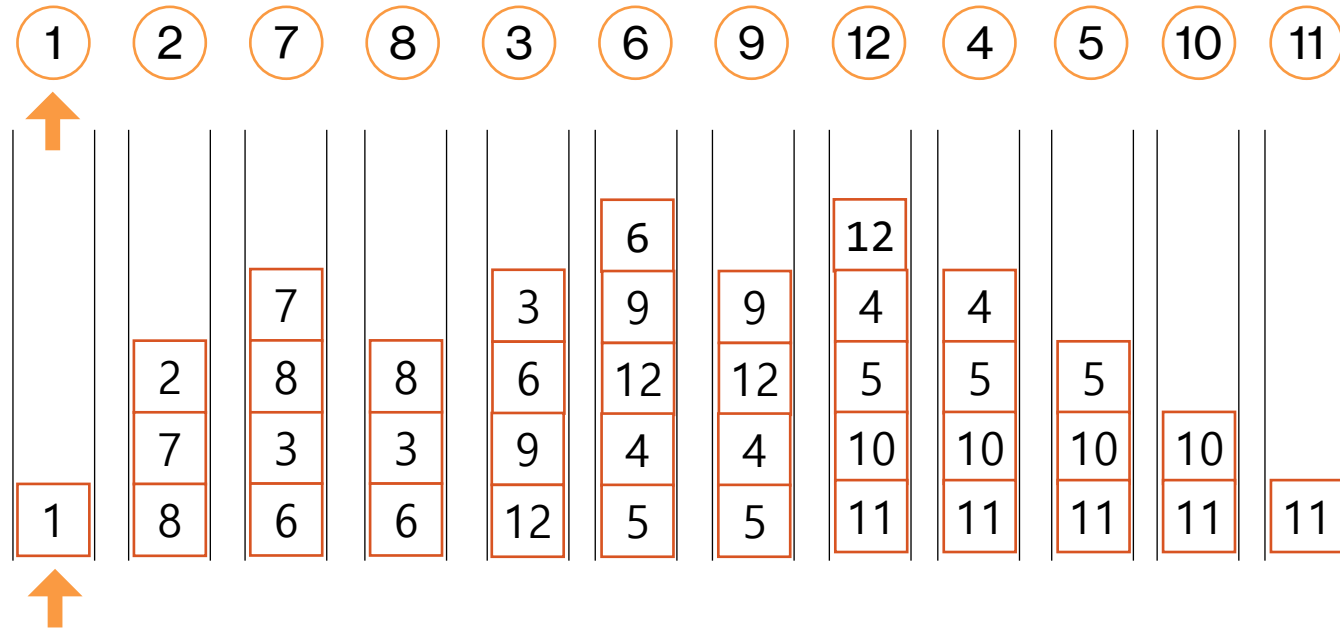
```
stack: [1]
1
stack: [8, 7, 2]
2
stack: [8, 7, 6, 3]
3
stack: [8, 7, 6, 5, 4]
4
stack: [8, 7, 6, 5]
5
stack: [8, 7, 6]
6
stack: [8, 7]
7
stack: [8]
8
stack: [12, 9]
9
stack: [12, 11, 10]
10
stack: [12, 11]
11
stack: [12]
12
```

트리구조의 너비우선탐색(BFS)

➤ 트리 구조



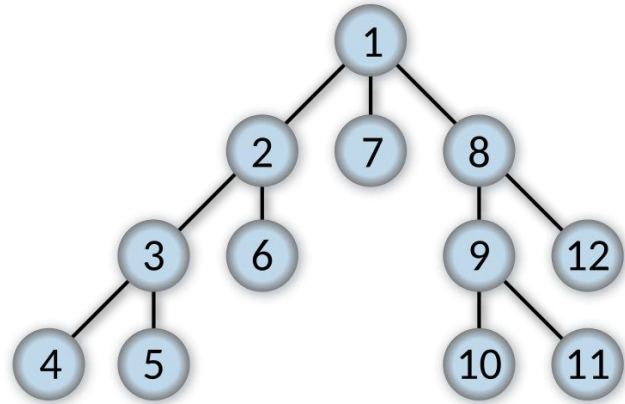
➤ 큐를 이용한 BFS 순회



- bfs(1)
- bfs(2), bfs(7), bfs(8)
- dfs(3), dfs(6), dfs(9), dfs(12)
- dfs(4), dfs(5), dfs(10), dfs(11)

트리구조의 너비우선탐색(BFS)

```
# queue 로 구현하는 BFS
from copy import deepcopy
from collections import deque
adjl = []
for i in range(0,13):
    adjl.append(None)
adjl[1] = [2,7,8]
adjl[2] = [3,6]
adjl[3] = [4,5]
adjl[8] = [9,12]
adjl[9] = [10,11]
print(adjl)
```



```
queue = deque([1]) # 탐색할 노드를 저장하는 큐 (초기 상태 삽입)
while queue: # 큐에 탐색할 노드가 남아있는 동안 반복
    print('queue: ', queue)
    # 큐의 맨 앞(가장 먼저 들어온 노드)을 꺼내어 현재 상태로 지정 (FIFO)
    currentnode = queue.popleft()
    print(currentnode)

    # 자식 노드(다음 이동 가능한 상태) 생성
    if(adjl[currentnode] is not None):
        bnodes = adjl[currentnode]

        # 자식 노드를 큐의 맨 뒤에 추가
        for child in bnodes:
            # 이미 방문했거나 큐에 대기 중인 상태는 중복 탐색하지 않음
            if child not in queue:
                queue.append(child)
```

```
[None, [2, 7, 8], [3, 6], [4, 5], None, None, None,
None, [9, 12], [10, 11], None, None, None]
```

```
queue: deque([1])
1
queue: deque([2, 7, 8])
2
queue: deque([7, 8, 3, 6])
7
queue: deque([8, 3, 6])
8
queue: deque([3, 6, 9, 12])
3
queue: deque([6, 9, 12, 4, 5])
6
queue: deque([9, 12, 4, 5])
9
queue: deque([12, 4, 5, 10, 11])
12
queue: deque([4, 5, 10, 11])
4
queue: deque([5, 10, 11])
5
queue: deque([10, 11])
10
queue: deque([11])
11
```