

# 1. 파이썬 기본문법

---

# 1. Python 2.x VS Python 3.x

---

파이썬 2.x에서 3으로의 변경 사항 중에 특히 주목해야 할 것은 파이썬3이 2.x 버전들과 하위 호환성을 전혀 없음

파이썬을 처음 배우시는 분들은 파이썬 3.x로 학습

그러나, 파이썬 2.x와 차이점을 알아 둘 필요가 있음

# 1. Python 2.x VS Python 3.x

---

print 함수

**2.X style :**

```
print "welcome to", "python3k"
```

Welcome to python3k

**3 style :**

```
print("welcome to", "python3k")
```

Welcome to python3k

# 1. Python 2.x VS Python 3.x

---

Long형이 없어지고 int형으로 통

**2.X style :**

```
type(2**31)
```

```
<type 'long'>
```

**3 style :**

```
type(2**31)
```

```
<class 'int'>
```

- 2의 31제곱이 2.x에서는 long형이었는데, 3에서는 2의 31제곱은 물론 2의 100제곱도 int형 인 것을 확인할 수 있습니다.
- 즉 2.x에서는 sys.maxint이하의 값은 int로 처리되고 그 이상의 값은 long으로 처리되었지만 3에서는 모두 int로 처리됩니다.

# 1. Python 2.x VS Python 3.x

---

"int/int"의 결과는 float

**2.X style :**

```
1/2
```

```
0
```

**3 style :**

```
3/2
```

```
1.5
```

```
type(2/2)
```

```
<class 'float'>
```

- 2.x에서는 int/int의 결과가 int로만 나와서 파이썬에 익숙하지 않은 사용자에게 예상 밖의 결과가 많이 나왔지만 3에서는 그렇지 않다.

# 1. Python 2.x VS Python 3.x

String, Unicode체계가 바뀌었습니다.

파이썬 2.x에서는 string과 unicode로 구분

**2.X style :**

```
type('가')
```

```
<type 'str'>
```

```
type('가'.decode('utf-8'))
```

```
<type 'unicode'>
```

```
type(u'가')
```

```
<type 'unicode'>
```

파이썬 3에서는 string과 bytes로 구분

**3 style :**

```
type('가')
```

```
<type 'str'>
```

```
type('가'.decode('cp949'))
```

```
<type 'bytes'>
```

```
type(u'가')
```

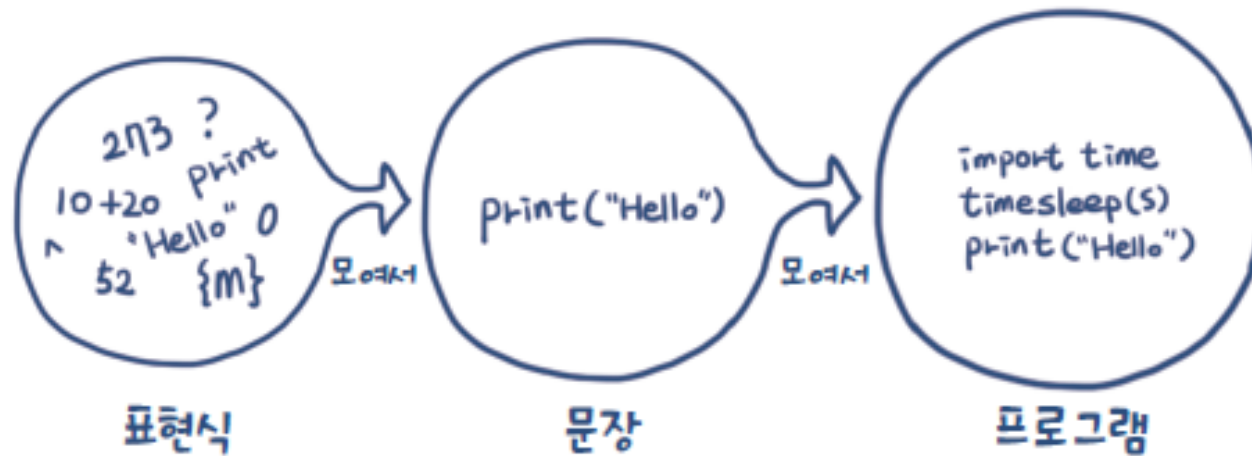
```
SyntaxError : invalid syntax
```

파이썬 2.x에서는 일반 문자열이 인코딩이 있는 문자열이었고 유니코드가 따로 있었는데, 파이썬3에서는 유니코드를 따로 지정하지 않고, 일반 문자열이 기존의 유니코드와 동일하며, 인코딩이 있는 문자열은 바이트로 표현됩니다.

## 2. 용어 정리

### 표현식

- 값을 만들어내는 간단한 코드
  - 273
  - $10 + 20 + 30 * 10$
  - "Python Programming"
- 하나 이상 표현식 모여 문장 구성
- 문장이 모여 프로그램 구성



## 2. 용어 정리

---

### 키워드 (Keyword)

- 특별한 의미가 부여된 단어
- 파이썬은 대소문자 구분 사용

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			



## 2. 용어 정리

### 식별자 (Identifier)

- 프로그래밍 언어에서 이름 붙일 때 사용하는 단어
- 알파벳 사용이 관례
- 사용 기본규칙
  - 키워드 사용 불가
  - 특수문자는 \_ 만 허용
  - 숫자로 시작 불가
  - 공백 사용 불가

사용 가능한 단어	사용 불가능한 단어
alpha	break ← 키워드라서 안 됩니다.
alpha10	273alpha ← 숫자로 시작해서 안 됩니다.
_alpha	has space ← 공백을 포함해서 안 됩니다.
AlpHa	
ALPHA	

## 2. 용어 정리

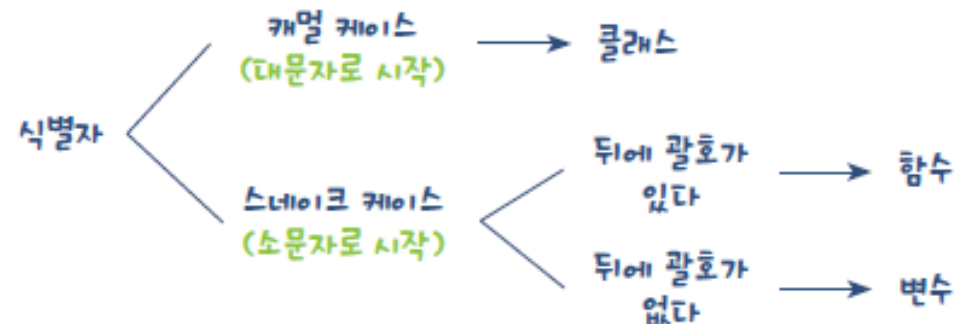
- 종류와 구분

- 스네이크 케이스 (snake\_case)

- 단어 사이에 \_ 기호 사용하여 식별자 형성

- 캐멀 케이스 (CamelCase)

- 단어 첫 글자를 대문자로 하여 식별자 형성



스네이크 케이스	캐멀 케이스
item_list	ItemList
login_status	LoginStatus
character_hp	CharacterHp
rotate_angle	RotateAngle

## 2. 용어 정리

---

- 함수 (Function)
  - 글자 뒤 괄호 있는 경우
  - 괄호에 문자열 등 자료 입력
    - Ex. print()
- 클래스, 변수, 함수를 구분해보자
  - print()
  - list()
  - soup.select()
  - math.pi
  - math.e
  - class Animal:
  - BeautifulSoup

## 2. 용어 정리

---

### 주석 (Comment)

- 프로그램 설명하는 코드
- 프로그램 진행에 영향 주지 않음
- 앞에 # 기호 붙여 주석 처리

# 간단히 출력하는 예입니다.

`print("Hello Python Programming...!")` # 문자열을 출력합니다.

# 기호 뒷부분이  
주석 처리됩니다.

# 3. 출력

현재 작업을 알 수 있도록 메시지 출력

코드 입력

- `print()` 함수 사용
- 괄호 안에 출력할 내용 입력
  - 여러 개 입력 가능
- 입력 후 `output.py`로 저장

```
# 하나만 출력합니다.
print("# 하나만 출력합니다.")
print("Hello Python Programming...!")
print()

# 여러 개를 출력합니다.
print("# 여러 개를 출력합니다.")
print(10, 20, 30, 40, 50)
print("안녕하세요", "저의", "이름은", "윤인성입니다!")
print()

# 아무 것도 입력하지 않으면 단순히 줄바꿈합니다.
print("# 아무 것도 출력하지 않습니다.")
print("--- 확인 전용선 ---")
print()
print()
print("--- 확인 전용선 ---")
```

# 3. 출력

---

## 출력

- 위 파일 저장한 폴더로 이동
- [Shift] + 마우스 오른쪽 버튼 클릭 → [여기서 명령 창 열기(w)] 클릭
- 명령 프롬프트에 python output.py 명령어 입력 실행

```
# 하나만 출력합니다.  
Hello Python Programming...!
```

```
# 여러 개를 출력합니다.  
10 20 30 40 50  
안녕하세요 저의 이름은 윤인성입니다!
```

```
# 아무 것도 출력하지 않습니다.  
--- 확인 전용선 ---
```

```
--- 확인 전용선 ---
```

# 4. 문자열

---

## 문자의 나열 (文字列, String)

- "Hello Python Programming...!"
- "안녕하세요"

## 큰따옴표/작은따옴표 사용

- "<글자>" / '<글자>'
- `print("안녕하세요")` / `print('안녕하세요')`
  - 안녕하세요 출력

# 5. 이스케이프 문자

문자열 내부에 따옴표 넣기

- ""안녕하세요"라고 말했습니다"
- print("""안녕하세요"라고 말했습니다")
  - 구문 오류 (Syntax Error) 발생
  - 여러 자료(문자열)를 단순히 나열할 수 없음

File "<stdin>", line 1

```
print("""안녕하세요"라고 말했습니다")
```

^

SyntaxError: invalid syntax

""안녕하세요"라고 말했습니다"

문자열로 인식되는 부분



## 5. 이스케이프 문자

---

- 큰따옴표를 문자열 내부에 넣는 경우
  - `print("안녕하세요"라고 말했습니다)`  
→ "안녕하세요"라고 말했습니다
- 작은따옴표를 문자열 내부에 넣는 경우
  - `print("'배가 고픈다'라고 생각했습니다")`  
→ '배가 고픈다'라고 생각했습니다

## 5. 이스케이프 문자

### 이스케이프 문자 (Escape Character)

- \ (백슬래시) 기호와 함께 문자 조합하여 사용
- 문자열 만드는 기호 아닌 단순 따옴표로 인식

이스케이프 문자	설명
\"	큰따옴표를 의미합니다.
\'	작은따옴표를 의미합니다.

```
print("\"안녕하세요\"라고 말했습니다")
```

```
print('\배가 고픈다\'라고 생각했습니다')
```



"안녕하세요"라고 말했습니다

'배가 고픈다'라고 생각했습니다

## 5. 이스케이프 문자

- 그 외 다양한 기능

이스케이프 문자	설명
\n	줄바꿈을 의미합니다.
\t	탭을 의미합니다.

<pre>print("이름\t나이\t지역") print("윤인성\t25\t강서구") print("윤아린\t24\t강서구") print("구름\t3\t강서구")</pre>	→	<table><tr><td>이름</td><td>나이</td><td>지역</td></tr><tr><td>윤인성</td><td>25</td><td>강서구</td></tr><tr><td>윤아린</td><td>24</td><td>강서구</td></tr><tr><td>구름</td><td>3</td><td>강서구</td></tr></table>	이름	나이	지역	윤인성	25	강서구	윤아린	24	강서구	구름	3	강서구
이름	나이	지역												
윤인성	25	강서구												
윤아린	24	강서구												
구름	3	강서구												

이스케이프 문자	설명
\\	\를 의미합니다.

<pre>print("\\ \\ \\ \\")</pre>	→	<pre>\\ \\ \\ \\</pre>
---------------------------------	---	------------------------

# 5. 여러 줄 문자열 만들기

## 여러 줄 문자열

- \n 대신 사용
- 큰따옴표/작은따옴표 3번 반복

```
"""<문자열>
```

```
<문자열>
```

```
<문자열>"""
```

```
print("동해물과 백두산이 마르고 닳도록\n하느님이 보우하사 우리나라 만세\n무궁화 삼천리 화려강산 대한사람\n대한으로 길이 보전하세")
```



```
print("""동해물과 백두산이 마르고 닳도록
하느님이 보우하사 우리나라 만세
무궁화 삼천리 화려 강산
대한사람 대한으로 길이 보전하세""")
```



동해물과 백두산이 마르고 닳도록  
하느님이 보우하사 우리나라 만세  
무궁화 삼천리 화려강산 대한사람  
대한으로 길이 보전하세

## 6. 문자열 연산자

각각의 자료에 적합한 연산자 사용

### 문자열 연결 연산자 (+)

- 두 문자열 연결하여 새로운 문자열 형성

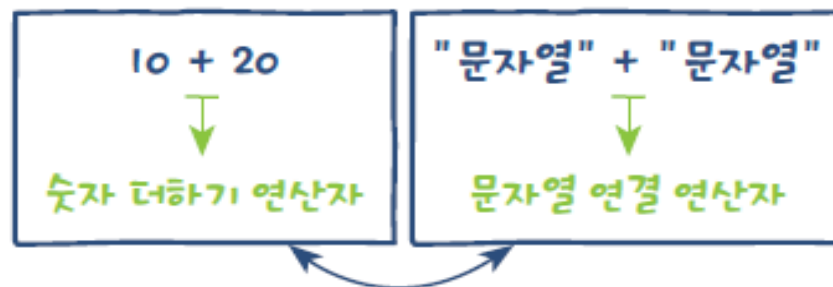
〈문자열〉 + 〈문자열〉

# 출력합니다.

`print("안녕하세요" + "...!")`



안녕하세요...!



## 6. 문자열 연산자

### 문자열 반복 연산자 (\*)

- 문자열을 숫자와 \* 로 연결하여 반복

〈문자열〉 \* 〈숫자〉 또는 〈숫자〉 \* 〈문자열〉

문

```
# 출력합니다.
```

```
print("안녕하세요" * 3)
```



안녕하세요안녕하세요안녕하세요

```
# 출력합니다.
```

```
print("# 문자 선택 연산자")
```

```
print("안녕하세요"[0])
```

```
print("안녕하세요"[1])
```

```
print("안녕하세요"[2])
```

```
print("안녕하세요"[3])
```

```
print("안녕하세요"[4])
```



〈문자열〉[〈숫자〉]

```
# 문자 선택 연산자
```

```
안
```

```
녕
```

```
하
```

```
세
```

```
요
```

## 6. 문자열 연산자

### 문자열 범위 선택 연산자

- 문자열의 특정 범위 선택
- 대괄호 안 숫자는 둘 중 하나 생략 가능
- 범위 적용 숫자에 주의

<문자열>[<숫자>:<숫자>]

# 출력합니다.

```
print("# 문자 범위 선택 연산자")
print("안녕하세요"[0:2]:', "안녕하세요"[0:2])
print("안녕하세요"[1:3]:', "안녕하세요"[1:3])
print("안녕하세요"[2:4]:', "안녕하세요"[2:4])
print()
```

```
print("# 문자 범위 선택 연산자 : 한쪽 숫자 생략")
print("안녕하세요"[3:]:', "안녕하세요"[3:])
print("안녕하세요"[:3]:', "안녕하세요"[:3])
```

# 문자 범위 선택 연산자

"안녕하세요"[0:2]: 안녕  
"안녕하세요"[1:3]: 녕하  
"안녕하세요"[2:4]: 하세

# 문자 범위 선택 연산자 : 한쪽 숫자 생략

"안녕하세요"[3:]: 세요  
"안녕하세요"[:3]: 안녕하

## 7. 문자열 길이 구하기

## len() 함수 사용

- 괄호 안 문자열의 글자 수 표시

len(<문자열 또는 리스트>)

```
print(len("안녕하세요"))
```

→ 5

```
print(len("안녕하세요"))
```

```
print(5)
```

5



## 8. 자료형 확인하기

---

`type()` 함수 사용

- 자료의 형태 확인

`type(<자료>)`

```
print(type("안녕하세요"))  
print(type(len("안녕하세요")))
```



```
<class 'str'>  
<class 'int'>
```

# 9. 숫자

## 숫자 만들기

- 숫자를 입력

숫자  
◦

```
print(273)
print(52.273)
```

→ 273  
52.273

숫자

단어	의미
int(integer)	정수
float(floating point)	부동 소수점

```
print(type(52))
print(type(52.273))
```


→ <class 'int'>  
<class 'float'>

# 10. 숫자 연산자

## 사칙 연산자

연산자	설명
+	숫자 덧셈 연산자
-	숫자 뺄셈 연산자
*	숫자 곱셈 연산자
/	숫자 나눗셈 연산자

```
print(5 + 7)      12
print(5 - 7)      -2
print(5 * 7)      35
print(5 / 7)      0.7142857142857143
```



### ⚠️ 단점

- 숫자 나누고 소수점 이하 자릿수 삭제

```
print("3 / 2:", 3 / 2)      3 / 2: 1.5
print("3 // 2:", 3 // 2)    3 // 2: 1
```



# 10. 숫자 연산자

## ❖ 나머지 연산자

- $A \% B = A$ 를  $B$ 로 나누었을 때의 나머지

`print(5 % 2)`  1

## ❖ 제곱 연산자

- $\langle \text{숫자A} \rangle ** \langle \text{숫자B} \rangle = \langle \text{숫자A} \rangle^{\langle \text{숫자B} \rangle}$

# 출력합니다.

`print("# 제곱 연산자")`

`print("2 ** 1 =", 2 ** 1)`

`print("2 ** 2 =", 2 ** 2)`

`print("2 ** 3 =", 2 ** 3)`

`print("2 ** 4 =", 2 ** 4)`



# 제곱 연산자

2 \*\* 1 = 2

2 \*\* 2 = 4

2 \*\* 3 = 8

2 \*\* 4 = 16

# 10. 숫자 연산자

## 연산자의 우선순위

- 곱셈과 나눗셈을 덧셈과 뺄셈보다 우선시
- 같은 우선순위는 왼쪽에서 오른쪽 순서로 계산

```
print("2 + 2 - 2 * 2 / 2 * 2 = ", 2 + 2 - 2 * 2 / 2 * 2)
print("2 - 2 + 2 / 2 * 2 + 2 = ", 2 - 2 + 2 / 2 * 2 + 2)
```



- 괄호 사용하여 우선순위를 지정
- 우선순위 확실한  $2 + 2 - 2 * 2 / 2 * 2 = 0.0$
- 문자열 연산자  $2 - 2 + 2 / 2 * 2 + 2 = 4.0$

# 11. 변수

## 값을 저장하는 식별자

- 숫자 및 모든 자료형 저장 가능

```
pi = 3.14159265  
print(pi)
```



3.14159265

# 변수에 값 넣기

<변수 이름> = 값

# 변수 사용하기

<변수 이름>

- 같은 변수에 여러 종류의 자료형 넣을 수 있음
  - 실행 중 TypeError 발생할 확률
  - 되도록 하나의 자료형 넣어 활용

## 12. 복합 대입 연산자

변수 활용하여 기존 연산자와 다른 연산자 조합

- “자료형에 적용하는 기본 연산자” & “= 연산자”
- Ex)  $a += 10 \rightarrow a = a + 10$
- 숫자에 적용

연산자 이름	설명
<code>+=</code>	숫자 덧셈 후 대입
<code>-=</code>	숫자 뺄셈 후 대입
<code>*=</code>	숫자 곱셈 후 대입
<code>/=</code>	숫자 나눗셈 후 대입
<code>%=</code>	숫자 나머지 구한 후 대입
<code>**=</code>	숫자 제곱 후 대입

- 문자열에 적용

연산자 이름	설명
<code>+=</code>	문자열 연결 후 대입
<code>*=</code>	문자열 반복 후 대입

## 12. 복합 대입 연산자

◦ Ex)

```
# 숫자 복합 대입 연산자
```

```
number = 100
```

```
number += 10
```

```
number += 20
```

```
number += 30
```

```
print("number:", number)
```



```
number: 160
```

```
string: 안녕하세요!!
```

```
# 문자열 복합 대입 연산자
```

```
string = "안녕하세요"
```

```
string += "!"
```

```
string += "!"
```

```
print("string:", string)
```



# 13. 입력

## 기본 입력

- `input()` 함수

- 코드 실행 후 나타나는 "입력>" 문자열 옆에 내용 입력
- 출력 후 프로그램 종료

```
# 입력을 받습니다.
```

```
string = input("입력> ")
```

```
# 출력합니다.
```

```
print(string)
```

<입력 받을 변수> = `input(<프롬프트 문자열>)`



```
0% 명령 프롬프트
C:\Users\hasat\sample>python test.py
입력> 안녕하세요
안녕하세요
```

# 13. 입력

## 입력 자료형

- input() 함수의 결과는 무조건 문자열 자료형
  - 숫자 입력해도 문자열로 들어옴

```
# 입력을 받습니다.  
string = input("입력> ")  
  
# 출력합니다.  
print("입력:", string)  
print("자료형:", type(string))
```



```
입력> 52273  
입력: 52273  
자료형: <class 'str'>  
  
입력> True  
입력: True  
자료형: <class 'str'>
```

# 13. 입력

- 입력받은 것과 숫자 더하는 코드 작성하려 할 경우

```
# 입력을 받습니다.
```

```
string = input("입력> ")
```

```
# 출력합니다.
```

```
print("입력 + 100:", string + 100)
```

문자열과 숫자를 더해버렸어요.



```
입력> 100
```

```
Traceback (most recent call last):
```

```
File "c:\Users\hasat\sample\test.py", line 4, in <module>
```

```
    print("입력 + 100:", string + 100)
```

```
TypeError: must be str, not int
```

# 14. 문자열을 숫자로 바꾸기

## 자료형 변환 (Cast)

- 하나의 자료형을 다른 자료형으로 바꾸는 것

함수	설명
int( )	문자열을 int 자료형으로 변환합니다.
float( )	문자열을 float 자료형으로 변환합니다.

```
output_a = int("52")
```

```
output_b = float("52.273")
```

```
print(type(output_a), output_a)
```

```
print(type(output_b), output_b)
```



```
<class 'int'> 52
```

```
<class 'float'> 52.273
```

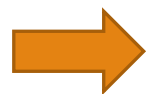
# 14. 문자열을 숫자로 바꾸기

숫자 연산 프로그램

- `input()` 함수와 `float()` 함수 조합

```
input_a = float(input("첫 번째 숫자> "))  
input_b = float(input("두 번째 숫자> "))
```

```
print("덧셈 결과:", input_a + input_b)  
print("뺄셈 결과:", input_a - input_b)  
print("곱셈 결과:", input_a * input_b)  
print("나눗셈 결과:", input_a / input_b)
```



```
첫 번째 숫자> 273  
두 번째 숫자> 52  
덧셈 결과: 325.0  
뺄셈 결과: 221.0  
곱셈 결과: 14196.0  
나눗셈 결과: 5.25
```

# 15. 숫자를 문자열로 바꾸기

---

`str()` 함수 사용

- 문자열로의 변환

```
output_a = str(52)
```

```
output_b = str(52.273)
```

```
print(type(output_a), output_a)
```

```
print(type(output_b), output_b)
```



```
<class 'str'> 52
```

```
<class 'str'> 52.273
```

# 16. 연습문제

---

다음 이스케이프 문자 중 줄바꿈에 사용되는 것은?

- \e
- \t
- \n
- \\\

다음 코드의 실행 결과를 예측하세요

```
print("0123456"[1:3])
```

inch 단위를 입력받아 cm 단위 구하는 코드를 작성하세요

킬로그램 단위 입력받아 파운드 단위 구하는 코드를 작성하세요

원의 반지름 입력받아 원의 둘레와 넓이를 구하는 코드를 작성하세요

## 2. 파이썬 기본

---

### 파이썬의 기본

- 변수 선언 필요 없음
- {} 블록이 없음
- if, 반복문 {} 가 없다 ---> 공백으로 블록 표시
- 주석
  - `""" """` : 여러줄 주석
  - `#` : 한줄 주석
- 콘솔 출력
- 콘솔 입력

`print()`

`input(문자열)`



## 2. 파이썬 기본

---

### 들여 쓰기

- 들여쓰기는 파이썬의 가장 큰 특징 중 하나입니다.
- 가독성을 높이는 유용한 수단이면서 치명적이고 발견하기 어려운 버그를 만들어 내기도 합니다.
- 그러므로 여러 명이 함께 작업을 할 때는 들여쓰기를 어떻게 할지 명확히 정하는 것이 필수입니다.
- 다음과 같은 코드 블록이 있다고 합시다.

```
코드 블록1:  
    코드 블록2  
코드 블록3:  
    코드 블록 4
```

- 코드블록 2는 코드블록1의 하위 레벨이며, 파이썬에서는 하위 레벨의 코드블록이 나오려면 바로 전에 콜론을 사용합니다.
- 동일한 코드블록 내에서는 정확히 같은 스페이스, 또는 탭을 이용한 들여쓰기를 사용해야 합니다.

## 2. 파이썬 기본

### ■ 들여 쓰기

그러나 다음 예제처럼 동일한 코드블록에서 다른 들여쓰기를 사용하면 에러가 발생합니다.

```
if a = 1:
    print(1)  ← 1 tab
print(0)     ← 1 space
```

SyntaxError: unindent does not match any outer indentation level (<pyshell#11>, line 3)

즉, 이 예제의 경우 같은 레벨의 코드블록이나 상위레벨의 코드블록의 들여쓰기가 같지 않아서 나는 에러입니다.

```
if a = 1:
    print(1)  ← 1 tab
    print(0) ← 1 tab + 1 space
```

SyntaxError: unexpected indent (<pyshell#15>, line 3)

## 2. 파이썬 기본

---

### 들여쓰기

- 다음 예제는 하위레벨의 코드블록이 나올것을 예상하지 못해서 에러가 발생 하위 레벨의 코드블록이 나오기 전에는 반드시 콜론(:)을 사용하는 구문이 나와야 합니다. 예외적으로 1줄짜리 간단한 하위 레벨은 다음과 같이 사용할 수 있습니다.
- 물론 여러 구문도 다음과 같이 한줄로 표현할 수 있습니다.

```
>>> if a ==1 : print(1)
```

```
>>> if a ==1 : print(1);print(0)
```

## 2. 파이썬 기본

---

### 소스코드 인코딩

- 파이썬 프로그램에서 주석은 #을 사용합니다. 즉 #이후로는 코드를 실행하지 않습니다. 이밖에도 #의 용도는 여러가지가 있는데, 아래와 같이 실행 파일과 소스코드 인코딩을 명시하는데 사용할 수 있습니다.

```
#!/usr/bin/python  
# coding:latin-1
```

- 아래는 리눅스 환경에서 실행 파일 경로와 인코딩을 명시한 예입니다.
- 소스 인코딩의 경우 위의 두번째 줄처럼 coding 지시자를 이용해 지정할 수 있고, 지정돼 있지 않은 경우 아스키가 기본 소스코드 인코딩으로 설정됩니다. 다음과 같이 소스코드 인코딩은 -\*-를 이용해서도 지정할 수 있습니다.

```
#-*-:utf-8 -*-
```

## 2. 파이썬 기본

---

### 세미콜론

- 파이썬에서는 다음과 같이 라인의 끝에 아무것도 붙이지 않아도 됩니다. C/C++를 사용하던 분들은 세미콜론(;)을 붙이는 습관이 있는데, 파이썬에서는 세미콜론을 붙이거나 붙이지 않아도 동일하게 동작합니다.

```
>>> a = 1  
>>> b = 2;
```

- 다만 세미콜론을 사용해야 하는 경우가 있는데 바로 한라인에 여러구분이 올때는 각 구문을 세미콜론으로 구분합니다.
- 파이썬에서는 들여쓰기가 중요하다고 강조했습니다. 다음과 같이 문장이 아직 끝나지 않은 경우에는 들여쓰기를 하지 않아도 문법 에러가 나지 않지만 가독성을 위해서라도 들여쓰기는 지키는 것이 좋습니다.

```
>>> a = 1; b=2
```

## 2. 파이썬 기본

### 들여쓰기 및 문장의 연결

- 파이썬에서는 들여쓰기가 중요하다고 강조했습니다. 다음과 같이 문장이 아직 끝나지 않은 경우에는 들여쓰기를 하지 않아도 문법 에러가 나지 않지만 가독성을 위해서라도 들여쓰기는 지키는 것이 좋습니다.

```
>>> a = (1 +  
        2 +  
        3 +  
        4)  
>>> a  
10
```

- 다음과 같이 \로 문장이 아직 끝나지 않았다는 사실을 명시 할 수도 있습니다.

```
>>> a = [ 1 +\  
        2 +\  
        4 +\  
        3]  
>>> a  
[10]  
>>> b =\  
        3.14
```

## 2. 파이썬 기본

---

### 진수 변환

```
sel = int(input("입력 진수 결정(16/10/8/2) :"))
num = input("값 입력:")
if sel == 16:
    num10 = int(num,16)
if sel == 10:
    num10 = int(num,10)
if sel == 8:
    num10 = int(num,8)
if sel == 2:
    num10 = int(num,2)
print(num10) #16
print(type(num))
print(type(num10))
num = num10
print(num)
print(type(num))
```

## 2. 파이썬 기본

---

10 진수를 진법의 값으로 출력

```
print("16진수=>",hex(num10))  
print("8진수=>",oct(num10))  
print("2진수=>",bin(num10))
```



## 2. 파이썬 기본

---

### print 함수

```
print(출력값1,출력값2,...)
print("형식지정문자를 가진 문자열",(출력값1,출력값2,...))
print(출력값,end="") : end =마지막 문자 지정. 기본값 \n
print(format 함수 지정하기)
```

```
a = int(input("값1 입력:"))
b = int(input("값2 입력:"))
result = a+b
print(a,"+",b,"=",result)
print("%d + %d = %d" % (a,b,result))
print("안"+"녕")           #문자열+문자열 가능
#print(a+" "+b+"="result)  #오류발생. 문자열+숫자형 연산안됨
print("안녕하세요",end=":")
print("홍길동 입니다.")
#format 함수 사용하기
print("{0:d} {1:5d} {2:05d}".format(100,200,300))
print("{2:d} {1:5d} {0:05d}".format(100,200,300))
```

## 2. 파이썬 기본

---

print 함수 연산자 사용하기

```
print("abc"+"abc"+"abc")  
print("abc","abc","abc")  
print("abc"*3)
```

print 함수 """ 문자열 연결하여 출력하기

```
print("안녕하세요 홍길동 입니다. 반갑습니다."  
      + "안녕하세요 홍길동 입니다. 반갑습니다.")  
print(""" 안녕하세요 홍길동 입니다. 반갑습니다.  
      안녕하세요 홍길동 입니다. 반갑습니다.""")  
print('안녕하세요 홍길동 입니다. 반갑습니다.'  
      + '안녕하세요 홍길동 입니다. 반갑습니다.')  
print(""" 안녕하세요 홍길동 입니다. 반갑습니다.  
      안녕하세요 홍길동 입니다. 반갑습니다.""")
```

파이썬에서 문자열 표시는 큰따옴표( " ), 작은 따옴표(') 모두 표시가 가능하다

### 3. 연산자

---

```
num1 = int(input("첫번째 정수 입력 : "))  
num2 = int(input("두번째 정수 입력 : "))  
print(num1,"+",num2,"=", (num1+num2)) #더하기  
print(num1,"-",num2,"=", (num1-num2)) #빼기  
print(num1,"*",num2,"=", (num1*num2)) #곱하기  
print(num1,"/",num2,"=", (num1/num2)) #나누기  
print(num1,"//",num2,"=", (num1//num2)) #정수 나누기  
print(num1,"%",num2,"=", (num1%num2)) #나머지  
print(num1,"**",num2,"=", (num1**num2)) #제곱
```

# 3. 연산자

---

## 비트 연산자

```
a = ord('A')                # 'A' 41 : 01000001 : 65
print(a)
mask = 0x0F                 # 16진수      mask 0F : 00001111 : 15

print("%X & %X = %X" % (a, mask, a & mask))      #&   : 00000001 : 1
print("%X | %X = %X" % (a, mask, a | mask))      #|   : 01001111 : 4F
print("%X ^ %X = %X" % (a, mask, a ^ mask))      #^   : 01001110 : 4E
```

## 4. 연산자 연습문제

---

- 1. 금액을 입력받아 동전(500,100,50,10,1)으로 바꿔 주는 프로그램 작성하기  
단 동전의 갯수는 최소개로 한다

결과

금액을 입력해주세요 : 3650  
500원 동전의 갯수: 7 개  
100원 동전의 갯수: 1 개  
50원 동전의 갯수: 1 개  
10원 동전의 갯수: 0 개  
1원 동전의 갯수: 0 개

## 4. 연산자 연습문제

---

2. 초를 입력받아 몇시간 몇분 몇초인지 출력하기

결과

초를 입력해주세요 : 3800  
1 시간 3 분 20 초

# 5. 조건문

---

if... else 구문

```
score = 50
if score >= 90:
    print("A학점")
else :
    if score >= 80 :
        print("B학점")
    else :
        if score >= 70 :
            print("C학점")
        else :
            if score >= 60 :
                print("D학점")
            else :
                print("F학점")
```

■ if... elif 구문

```
if score >= 90:
    print("A학점")
elif score >= 80:
    print("B학점")
elif score >= 70:
    print("C학점")
elif score >= 60:
    print("D학점")
else:
    print("F학점")
```

## 6. 반복문

---

for 구문

- 숫자를 입력 받아 입력 받은 수까지의 합을 구하기

```
num=int(input("숫자를 입력하세요 : "))
sum=0;
for i in range(1,num+1) :
    sum += i
print("1부터",num,"까지의 합:",sum)
```



## 6. 반복문

---

while 구문

```
num=1
while num <= 5:
    print(num,end="")
    num +=1
```

- for 구문으로 변경하기

```
for num in range(1,6) :
    print(num,end="")
```

## 6. 반복문

---

랜덤함수를 사용하여 숫자 맞추기

```
import random                #난수값 구하기 위한 모듈
rnum = random.randrange(1,100) #1부터 99까지의 임의의 수
i=1
while True :                 #무한반복
    a = int(input("숫자를 입력하세요 : "))
    if a > rnum :
        print("작은수 입니다")
    elif a < rnum :
        print("큰수 입니다.")
    else :
        print("정답입니다.")
        break                #반복문을 벗어나
    i = i+1
print("%d번만에 맞췄습니다" % i)
```

## 7. 문자열

---

<code>print("안녕하세요")</code>	<code>#문자열</code>
<code>print("안녕하세요"[0])</code>	<code>#0번 인덱스 문자</code>
<code>print("안녕하세요"[4])</code>	<code>#4번 인덱스 문자</code>
<code>print("안녕하세요"[-1])</code>	<code>#요. 뒤에서 첫번째 인덱스에 해당하는 문자</code>
<code>print("안녕하세요"[-2])</code>	<code>#세. 뒤에서 두번째 인덱스에 해당하는 문자</code>
<code>print("안녕하세요"[1:3])</code>	<code>#1번인덱스부터 2번인덱스까지의 부분문자열</code>
<code>print("안녕하세요"[:3])</code>	<code>#0번인덱스부터 2번인덱스까지의 부분문자열</code>
<code>print("안녕하세요"[3:])</code>	<code>#3번인덱스부터 끝까지 부분 문자열</code>
<code>print("안녕하세요"[:2])</code>	<code>#2칸씩 부분 문자</code>
<code>print("안녕하세요"[:-1])</code>	<code>#역순 문자</code>
<code>print("안녕하세요"[:-2])</code>	<code>#역순 2칸 건너 문자</code>
<code>print("'안녕하세요' 문자열의 길이 :",len("안녕하세요"))</code>	

## 7. 문자열

---

```
a="hello"                                # l문자의 갯수 출력하기
cnt=0
for i in range(0,len(a)) :
    if(a[i]=='l') :
        cnt += 1
print("l문자의 갯수:",cnt)

print("l문자의 갯수:",a.count('l'))
print("h문자의 갯수:",a.count('h'))
print("a문자의 갯수:",a.count('a'))

print("l문자의 인덱스값:",a.find('l'))
print("a문자의 인덱스값:",a.find('a'))    #-1 : 값없음

print("l문자의 인덱스값:",a.index('l'))
#print("a문자의 인덱스값:",a.index('a'))  # 오류 : 값없음
```

# 7. 문자열

---

문자열에서 문자의 종류관련 함수

```
instr="123"  
instr="Aa123"  
instr="  "  
if instr.isdigit() :  
    print(instr,"숫자입니다.")  
if instr.isalpha() :  
    print(instr,"문자입니다.")  
if instr.isalnum() :  
    print(instr,"문자+숫자입니다.")  
if instr.islower() :  
    print(instr,"소문자입니다.")  
if instr.isupper() :  
    print(instr,"대문자입니다.")  
if instr.isspace() :  
    print(instr,"공백입니다.")
```