

혼자
공부하는
파이썬

01-1 파이썬을 하기 전에 읽는 아주 간단한 설명

1. 009쪽 hint 참조
2. ①-a, ②-c, ③-f, ④-d, ⑤-e, ⑥-b
3. ③

01-2 파이썬을 배우기 위해 준비해야 할 것들

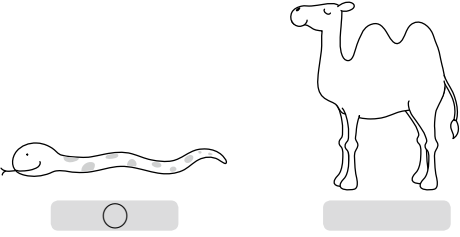
1.

```
>>> print("Hello Python")  
Hello Python
```
2. 소스 코드 01_2_2.py

```
Hello! Hello! Hello!  
혼자 공부하다 모르면 동영상 강의를 참고하세요!
```

01-3 이 책에서 자주 나오는 파이썬 용어들

1.

```
>>> print("Hello Python")  
Hello Python
```
2. ①-○, ②-○, ③-×, ④-×, ⑤-○
3. ②
4. 

5.	예시	스네이크 케이스	캐멀 케이스
	hello coding	hello_coding	HelloCoding
	hello python	hello_python	HelloPython
	we are the world	we_are_the_world	WeAreTheWorld
	create output	create_output	CreateOutput
	create request	create_request	CreateRequest
	init server	init_server	InitServer
	init matrix	init_matrix	InitMatrix

02-1 자료형과 문자열

1.	구문	의미
	"글자"	큰따옴표로 문자열 만들기
	'글자'	작은따옴표로 문자열 만들기
	"""문자열 문자열 문자열"""	여러 문자열 만들기

2.	이스케이프 문자	의미
	\	큰따옴표를 의미합니다.
	'	작은따옴표를 의미합니다.
	\n	줄바꿈을 의미합니다.
	\t	탭을 의미합니다.
	\\	\을 의미합니다.

3. 소스 코드 02_1_3.py

```
# 연습 문제
\\ \\
-----
```

4. 소스 코드 02_1_4.py

```
녕
하
세
요
Traceback (most recent call last):
  File "02_1_4.py", line 5, in <module>
    print("안녕하세요"[5])
IndexError: string index out of range
```

5. 소스 코드 02_1_5.py

```
녕하
하세
녕하세요
안녕하
```

02-2 숫자

1.

단어	예시
int	273, 52, 0, 1234, -25
float	0.0, 1.234, 2.73e2, -25.0

2.

연산자	의미
+	덧셈 연산자
-	뺄셈 연산자
*	곱셈 연산자
/	나눗셈 연산자
//	정수 나누기 연산자
%	나머지 연산자
**	제곱 연산자

3. 소스 코드 02_2_3.py

```
# 기본적인 연산
15 + 4 = 19
15 - 4 = 11
15 * 4 = 60
15 / 4 = 3.75
```

4. 소스 코드 02_2_4.py

```
print("3462를 17로 나누었을 때의")
print("- 몫:", 3462 // 17)
print("- 나머지:", 3462 % 16)
```

5.

```
>>> print(2 + 2 - 2 * 2 / 2 * 2)
0.0
>>> print(2 - 2 + 2 / 2 * 2 + 2)
4.0
```

02-3 변수와 입력

1. =

2.

연산자	내용
+=	숫자 덧셈 후 대입
-=	숫자 뺄셈 후 대입
*=	숫자 곱셈 후 대입
/=	숫자 나눗셈 후 대입
%=	숫자 나머지 구한 후 대입
**=	숫자 제곱 후 대입

3.

함수	내용
int	문자열을 int 자료형으로 변환
float	문자열을 float 자료형으로 변환
str	숫자를 문자열로 변환

4. 소스 코드 02_3_4.py

```
str_input = input("숫자 입력> ")
num_input = float(str_input)

print()
print(num_input, "inch")
print((num_input * 2.54), "cm")
```

5. 소스 코드 02_3_5.py

```
str_input = input("원의 반지름 입력> ")
num_input = float(str_input)
print()
print("반지름: ", num_input)
print("둘레: ", 2 * 3.14 * num_input)
print("넓이: ", 3.14 * num_input ** 2)
```

6. 소스 코드 02_3_6.py

```
a = input("문자열 입력> ")
b = input("문자열 입력> ")

print(a, b)
c = a
a = b
b = c
print(a, b)
```

02-4 숫자와 문자열의 다양한 기능

1. ③

2. ①-d, ②-b, ③-a, ④-c

3. 소스 코드 02_4_3.py

```
a = input("> 1번째 숫자: ")
b = input("> 2번째 숫자: ")
print()
print("{} + {} = {}".format(a, b, int(a) + int(b)))
```

4. 소스 코드 02_4_4.py

```
A 지점: hello
B 지점: HELLO
```

03-1 불 자료형과 if 조건문

1.

조건식	결과
10 == 100	False
10 != 100	True
10 > 100	False
10 < 100	True
10 <= 100	True
10 >= 100	False

2. ③

3. ① OR ② AND ③ OR

4. 소스 파일: 03_1_4.py

```

a = float(input("> 1번째 숫자: "))
b = float(input("> 2번째 숫자: "))
print()

if a > b:
    print("처음 입력했던 {}가 {}보다 더 큼니다".format(a, b))
if a < b:
    print("두 번째로 입력했던 {}가 {}보다 더 큼니다".format(b, a))

```

03-2 if~else와 elif 구문

1. ① 12, ② 5, ③ 출력 없음

2.

```

if x > 10 and x < 20
    print("조건에 맞습니다.")

```

3. 소스 코드 03_2_3.py

```

str_input = input("태어난 해를 입력해 주세요> ")
birth_year = int(str_input) % 12

if birth_year == 0:
    print("원숭이 띠입니다.")
elif birth_year == 1:
    print("닭 띠입니다.")

```

```

elif birth_year == 2:
    print("개 띠입니다.")
elif birth_year == 3:
    print("돼지 띠입니다.")
elif birth_year == 4:
    print("쥐 띠입니다.")
elif birth_year == 5:
    print("소 띠입니다.")
elif birth_year == 6:
    print("범 띠입니다.")
elif birth_year == 7:
    print("토끼 띠입니다.")
elif birth_year == 9:
    print("뱀 띠입니다.")
elif birth_year == 10:
    print("말 띠입니다.")
elif birth_year == 11:
    print("양 띠입니다.")

```

04-1 리스트와 반복문

함수	list_a의 값
list_a.extend(list_a)	[0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4, 5, 6, 7]
list_a.append(10)	[0, 1, 2, 3, 4, 5, 6, 7, 10]
list_a.insert(3, 0)	[0, 1, 2, 0, 3, 4, 5, 6, 7]
list_a.remove(3)	[0, 1, 2, 4, 5, 6, 7]
list_a.pop(3)	[0, 1, 2, 4, 5, 6, 7]
list_a.clear()	[]

2. 소스 코드 04_1_2.py

```

numbers = [273, 103, 5, 32, 65, 9, 72, 800, 99]

for number in numbers:
    if number > 100:
        print("- 100 이상의 수:", number)

```

3. 왼쪽 실행결과 `소스 코드 04_1_3_1.py`

```
numbers = [273, 103, 5, 32, 65, 9, 72, 800, 99]
```

```
for number in numbers:
    if number % 2 == 1:
        print(number, "는 홀수입니다.")
    else:
        print(number, "는 짝수입니다.")
```

오른쪽 실행결과 `소스 코드 04_1_3_2.py`

```
numbers = [273, 103, 5, 32, 65, 9, 72, 800, 99]
```

```
for number in numbers:
    print(number, "는", len(str(number)), "자리수입니다.")
```

4. `소스 코드 04_1_4.py`

```
list_of_list = [
    [1, 2, 3],
    [4, 5, 6, 7],
    [8, 9],
]
```

```
for line in list_of_list:
    for item in line:
        print(item)
```

5. `소스 코드 04_1_5.py`

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
output = [[], [], []]
```

```
for number in numbers:
    output[(number + 2) % 3].append(number)

print(output)
```

04-2 딕셔너리와 반복문

1.	dict_a의 값	dict_a에 적용할 코드	dict_a의 결과
	{}	dict_a["name"] = "구름"	{"name": "구름"}
	{"name": "구름"}	del dict_a["name"]	{}

2. 소스 코드 04_2_2.py

```
# 딕셔너리를 선언합니다.
pets = [
    {"name": "구름", "age": 5},
    {"name": "초코", "age": 3},
    {"name": "아지", "age": 1},
    {"name": "호랑이", "age": 1}
]

print("# 우리 동네 애완 동물들")
for pet in pets:
    print(pet["name"], str(pet["age"]) + "살")
```

3. 소스 코드 04_2_3.py

```
# 숫자는 무작위로 입력해도 상관 없습니다.
numbers = [1,2,6,8,4,3,2,1,9,5,4,9,7,2,1,3,5,4,8,9,7,2,3]
counter = {}

for number in numbers:
    if number in counter:
        counter[number] = counter[number] + 1
    else:
        counter[number] = 1

# 최종 출력
print(counter)
```

4. 소스 코드 04_2_4.py

```
# 딕셔너리를 선언합니다.
character = {
    "name": "기사",
    "level": 12,
    "items": {
        "sword": "불꽃의 검",
        "armor": "풀플레이트"
    },
    "skill": ["베기", "세계 베기", "아주 세계 베기"]
}
```

```
# for 반복문을 사용합니다.
for key in character:
    if type(character[key]) is dict:
        for small_key in character[key]:
            print(small_key, ":", character[key][small_key])
    elif type(character[key]) is list:
        for item in character[key]:
            print(key, ":", item)
    else:
        print(key, ":", character[key])
```

04-3 반복문과 while 반복문

1.

코드	나타내는 값
range(5)	[0, 1, 2, 3, 4]
range(4, 6)	[4, 5]
range(7, 0, -1)	[7, 6, 5, 4, 3, 2, 1]
range(3, 8)	[3, 4, 5, 6, 7]
range(3, 9 + 1, 3)	[3, 6, 9]

2.

소스 코드 04_3_2.py

```
# 숫자는 무작위로 입력해도 상관없습니다.
key_list = ["name", "hp", "mp", "level"]
value_list = ["기사", 200, 30, 5]
character = {}

for i in range(0, len(key_list)):
    character[key_list[i]] = value_list[i]

# 최종 출력
print(character)
```

3.

소스 코드 04_3_3.py

```
limit = 10000
i = 1
# sum은 파이썬 내부에서 사용하는 식별자이므로 sum_value라는 변수 이름을 사용합니다.
sum_value = 0
while sum_value < limit:
    sum_value += i
    i += 1
print("{}를 더할 때 {}을 넘으며 그때의 값은 {}입니다.".format(i, limit, sum_value))
```

4. 소스 코드 04_3_4.py

```
max_value = 0
a = 0
b = 0

for i in range(1, 100 // 2 + 1):
    j = 100 - i

    # 최댓값 구하기
    current = i * j
    if max_value < current:
        a = i
        b = j
        max_value = current

print("최대가 되는 경우: {} * {} = {}".format(a, b, max_value))
```

04-4 문자열 리스트, 딕셔너리와 관련된 기본 함수

1. ①, ②

2. 소스 코드 04_4_2.py

```
# 리스트 내포를 사용해본 코드입니다.
output = [i for i in range(1, 100 + 1)
          if "{:b}".format(i).count("0") == 1]

for i in output:
    print("{} : {}".format(i, "{:b}".format(i)))
print("합계:", sum(output))
```

05-1 함수 만들기

1. ①

```
def f(x):
    return 2 * x + 1
print(f(10))
```

②

```
def f(x):
    return x ** 2 + 2 * x + 1
print(f(10))
```

2. 소스 코드 05_1_2.py

```
def mul(*values):
    output = 1
    for value in values:
        output *= value
    return output
```

```
# 함수를 호출합니다.
print(mul(5, 7, 9, 10))
```

3. ①

05-2 함수의 활용

1. 소스 코드 05_2_1.py

```
def flatten(data):
    output = []
    for item in data:
        if type(item) == list:
            output += flatten(item)
        else:
            output.append(item)
    return output
```

```
example = [[1, 2, 3], [4, [5, 6]], 7, [8, 9]]
print("원본:", example)
print("변환:", flatten(example))
```

2. 소스 코드 05_2_2.py

```
앉힐 수 있는 최소 사람 수 = 2
앉힐 수 있는 최대 사람 수 = 10
전체 사람의 수 = 100
memo = {}
```

```
def 문제(남은 사람 수, 앉힌 사람 수):
    key = str([남은 사람 수, 앉힌 사람 수])
    # 종료 조건
    if key in memo:
        return memo[key]
```

```

if 남은 사람 수 < 0:
    return 0 # 무효하니 0을 리턴
if 남은 사람 수 == 0:
    return 1 # 유효하니 수를 세면 되서 1을 리턴
# 재귀 처리
count = 0
for i in range(앉힌사람수, 앉힐수있는최대사람수 + 1):
    count += 문제(남은사람수 - i, i)
# 메모화 처리
memo[key] = count
# 종료
return count

```

```
print(문제(전체 사람의 수, 앉힐 수 있는 최소 사람수))
```

05-3 함수 고급

1. 소스 코드 05_3_1.py

```

numbers = [1, 2, 3, 4, 5, 6]

print(":".join(map(str, numbers)))

```

2. 소스 코드 05_3_2.py

```

numbers = list(range(1, 10 + 1))

print("# 홀수만 추출하기")
print(list(filter(lambda x: x % 2 == 1, numbers)))
print()

print("# 3 이상, 7 미만 추출하기")
print(list(filter(lambda x: 3 <= x < 7, numbers)))
print()

print("# 제곱해서 50 미만 추출하기")
print(list(filter(lambda x: x ** 2 < 50, numbers)))

```

06-1 구문 오류와 예외

1. 구문 오류: 프로그램이 실행되기도 전에 발생하는 오류. 해결하지 않으면 프로그램 자체가 실행되지 않음.

예외: 프로그램 실행 중에 발생하는 오류. 프로그램이 일단 실행되고 해당 지점에서 오류를 발생.

2. 소스 코드 06_1_2_1.py 06_1_2_2.py

```
numbers = [52, 273, 32, 103, 90, 10, 275]

print("# (1) 요소 내부에 있는 값 찾기")
print("- {}는 {} 위치에 있습니다.".format(52, numbers.index(52)))
print()

print("# (2) 요소 내부에 없는 값 찾기")
number = 10000
try: 또는 if number in numbers:
    print("- {}는 {} 위치에 있습니다.".format(52, numbers.index(52)))
except: 또는 else:
    print("- 리스트 내부에 없는 값입니다.")
print()

print("--- 정상적으로 종료되었습니다. ---")
```

3. ① 예외: ValueError, ② 예외: ValueError, ③ 구문 오류: SyntaxError,
④ 예외: IndexError

06-2 예외 고급

1. ② 2. 직접 정리해 보세요.

07-1 표준 모듈

1. ② 2. 직접 정리해 보세요.

3.

```
# 모듈을 읽어 들입니다.
import os

# 폴더를 읽어 들이는 함수
def read_folder(path):
    # 폴더의 요소 읽어 들이기
    output = os.listdir(path)
    # 폴더의 요소 구분하기
    for item in output:
        if os.path.isdir(item):
            # 폴더라면 계속 읽어 들이기
            read_folder(item)
        else:
            # 파일이라면 출력하기
            print("파일:", item)

# 현재 폴더의 파일/폴더를 출력합니다.
read_folder(".")
```

07-2 외부 모듈

1. primenumbers 모듈을 사용할 경우 다음과 같습니다. (외부 모듈의 경우 사용하기 전에 설치해 주어야 합니다. 333쪽 참고)

```
>>> import primenumbers
>>> primenumbers.all_PrimeNumbers_inRange(100, 1000)
[101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191,
193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283,
293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401,
409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509,
521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631,
641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751,
757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877,
881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997]
>>> len(primenumbers.all_PrimeNumbers_inRange(100, 1000))
143
```

2. 직접 정리해 보세요.

08-1 클래스의 기본

1. 직접 정리해 보세요. 2. 직접 정리해 보세요. 3. 직접 정리해 보세요.

08-2 클래스의 추가적인 구문

1. 소스 코드 08_2_1.py

```
# 클래스를 선언합니다.
class Student:
    def __init__(self, name, korean, math, english, science):
        self.name = name
        self.korean = korean
        self.math = math
        self.english = english
        self.science = science

    def get_sum(self):
        return self.korean + self.math + \
            self.english + self.science

    def get_average(self):
        return self.get_sum() / 4

    def __eq__(self, value):
        return self.get_average() == value
    def __ne__(self, value):
        return self.get_average() != value
    def __gt__(self, value):
        return self.get_average() > value
    def __ge__(self, value):
        return self.get_average() >= value
```



```
def __lt__(self, value):
    return self.get_average() < value

def __le__(self, value):
    return self.get_average() <= value
```

학생을 선언합니다.

```
test = Student("A", 90, 90, 90, 90)
```

출력합니다.

```
print("test == 90:", test == 90)
print("test != 90:", test != 90)
print("test > 90:", test > 90)
print("test >= 90:", test >= 90)
print("test < 90:", test < 90)
print("test <= 90:", test <= 90)
```