파이썬 기초

python programming

파이썬은 scope 개념이 거의 없어요. 왜냐면 { } 대괄호가 없기 때문이에요
scope이 생길때는 함수, class 만들었을때만 새로 생겨요

파이썬 특징

인터프리터의 스크립트 언어이다.
{} 블록이 아닌 들여쓰기 블록 사용
다른 언어와의 연동성이 뛰어나 생산성 향상

파이썬 모토 "Life is too short, You need Python."

파이썬 레퍼런스

https://docs.python.org/ko/3/reference/index.html

표준입출력

출력: print(1+2)
입력: var = input("문자를 입력해주세요 : ")

주석

한줄주석 : #주석문장
여러줄주석 :
"""
주석문
"""
또는
'''
주석문
'''

변수

타입을 지정하지 않아도 되지만 메모리에 할당후 자동으로 타입이 지정 및 변경
korea = "대한민국"
print(korea)
type(korea) :: <class 'str'>

산술연산자

3//2 == 1

2**2 == 4

+ - * /(실수결과) //(정수결과) % **(지수)

불리언

```
print(bool(1))          # True
print(bool(0))          # False

print(bool(None))       # False
print(bool([]))         # False
print(bool(()))         # False
print(bool({}))         # False
print(bool([1,2,3]))    # True

print(bool(""))         # False
print(bool("python"))   # True
```

비교연산

< <= > >= == !=

논리연산

and or not

문자열 표현

"python" 'python'

비트연산

& | ^ ~ << >>

```
a=[]

if a:
 print("true")
else :
 print("false")
```

여러줄 문자표현

""" 여러줄 문자표현 가능 """

''' 여러줄 문자표현 가능 '''

' 여러줄 ₩n 문자표현 ₩n 가능 '

문자열 연산

"파이썬"+"문자열" :: 파이썬문자열
"파이썬"*2 :: 파이썬파이썬

문자열 선택

py="파이썬코딩"
py[0] :: 파
py[-1] :: 딩
len(py) :: 5

문자열 슬라이싱 선택

py="우리집 강아지는 멍멍이"
py[4:7] :: 강아지
py[:3] ::우리집
py[-3:] :: 멍멍이

+= , *=, /=, %=, **=, //=

특정문자의 개수 및 위치

py = “python programming“
py.count(‘p’) :: 2
py.find(‘o’) :: 4
py.index(‘o’) :: 4
찾는 문자가 없을때
find 는 -1리턴
index는 오류

대소문자 변환

py.upper() 전부다 대문자로 변
py.lower() 전부다 소문자로 변

문자변경

py.replace(대상문자,변경문자)

문자분할

py.split(delimiter) :: 분할된 리스트 :: 기본값 whitespace

문자 분활은 분할 시킬 문자는 문자열에서 사라지고, 문자열이 쪼개져서 배열로

조건문

```
if 조건식 :
  print("참")
elif 조건식:
  print("거짓참")
else:
  print("거짓")
```

조건문에 따른 행동을 반드시 해야합니다.
 아무일도 하지않을때는 pass 키워드 사용
파이썬은 switch 구문이 없는 대시
if elif 또는 dictionary 를 이용할 수 있음

반복문 while

```
while 조건식:
  조건식이 참 동안 반복
```

```
i = 1
while i < 11: # 조건식
    print("파이썬 " + str(i))
    i = i + 1 # 탈출 조건
```

반복문 for

```
for 변수 in 반복가능객체:
  반복가능객체 끝까지 실행
```

```
str = "파이썬 프로그래밍"

for ch in str:
    print(ch)
```

범위지정함수 range()

range([시작정수],마지막정수)
range(5) :: 0,1,2,3,4
range(2,5) :: 2,3,4

for 문과 range 사용

```
for col in range(2,10):
  print(col)
```

:: 2,3,4,5,6,7,8,9

반복문 제어

break; 가장 가까운 루프 1회 탈출
continue; 현재요소 중지후 다음 요소 실행

자료타입 - 리스트

리스트명 = [ele1, ele2, ele3,...]
요소 선택 – 인덱싱 :: 정수타입리턴
요소슬라이싱 - :: 리스트타입리턴

```
listTest = [3,5,7,9,10];
print(listTest[3]);
print(listTest[3:4]);

:: 9
:: [9]
```

리스트,튜플,딕셔너리 참조형

```
* 레퍼런스참조는 주소만 복제됨
list1 = [1, 2, 3, 4, 5]
copy = list1
copy.append(6)
print(copy) :: [1,2,3,4,5,6]
print(list1) :: [1,2,3,4,5,6]

* 슬라이싱으로 새로운 복제 리스트 리턴
list1 = [1, 2, 3, 4, 5]
copy = list1[:]
copy.append(6)
print(copy)
print(list1)
```

리스트 연산

list1=[1,2,3]
list2 = list((4,5,6))
list1+list2 :: [1,2,3,4,5,6]
list1*2 :: [1,2,3,1,2,3]
list1.extend(list2) :: [1,2,3,4,5,6]

list(반복가능객체) :: list 변경

추가삭제

- list.append(2) :: 가장뒷쪽에 추가
- list.remove(3) :: 해당 요소 1개 지움

인덱스 번호로 지우란 소리가 아님. 진짜 원3을 하나 찾아서 지우란 뜻

삽입, 팝

list.insert(index,value) :: index 에 value 삽입 len 1커짐
list.pop([index]) :: index 번째 값 또는 맨뒤값 인출(배열에서는 삭제됨)

리버스, 소트

list.reverse() :: 원본 리버싱
list.sort() :: 원본 소팅
sorted(list) : 복사본 소팅 리턴 - 내장함수편

2차원이상

```
matrix = [[1, 2, 3], ["하나", "둘", "셋"]]
print(matrix[0])   :: [1,2,3]
print(matrix[0][0])   :: 1
print(matrix[1][2])   :: 셋
```

자료타입 - 튜플

리스트와 동일
차이점 : 값 변경불가
상수형으로 속도향상
추가,삭제,추출,정렬,역순 불가- sorted 는 복제본 리턴으로 가능함

튜플 선언

```
tuple1 = (1, 2, 3)
tupleTest = tuple(["원", "투", "쓰리"])
tuple2 = 1, 2, 3
tuple3 = (1,)
tuple4 = (1)
tuple5 = ()
```

원본유지후 사본 리턴하는 인덱싱선택, 슬라이싱, 튜플연산 가능

패킹언패킹

언패킹 받는곳이 적거나 더 많으면 오류발생

```
packingTest = ([1, 2, 3],[4, 5, 6],[7, 8, 9]);#패킹
a, b, c = packingTest;#언패킹
print(a);print(b);print(c)
```

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

```
packingTest1 = [[1, 2, 3],[4, 5, 6],[7, 8, 9]];#패킹
a1, b1, c1 = packingTest;#언패킹
print(a1);print(b1);print(c1)
```

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

```
tuple1 = 10, "열", True
# 언패킹(unpacking)
a, b, c = tuple1
# a, b, c, d = tuple1
# a, b = tuple1
print(a)
print(b)
print(c)
```

실행 결과

```
10
열
True
```

in 연산자

or 조건으로 각 요소 검사

```
tuple1 = 10, "열", True

print(10 in tuple1)
print("아홉" in tuple1)
print("아홉" not in tuple1)
```

실행 결과

True
False
True

```
listTest = [3, 5, 7, 9, 10];
print(10 in listTest) #:: True
```

자료타입 set

셋이름 = {요소1,요소2,....}

```
set1 = {1, 2, 3}
set2 = set("Python")
set3 = set("Hello")
print(set1)
print(set2)
print(set3)
```

실행 결과

{1, 2, 3}
{'P', 'y', 'h', 'n', 'o', 't'}
{'l', 'e', 'o', 'H'}

특징 :

1. 저장 순서 무작위
2. 중복데이터허용안함

```
ltest = [1,3,5,7,5]
set4 = set(ltest)
print(set4)
```

{1, 3, 5, 7}

빈 셋

set();

셋의 요소 추가삭제제거

set.add(추가데이터)
set.update(반복객체)
set.remove(삭제데이터)

셋의 집합연산

```
set1 = {1, 2, 3, 4, 5}
set2 = set((1, 3, 5, 7, 9))
print(set1)
print(set2)
print(set1 | set2) # 합집합
print(set1 & set2) # 교집합
print(set1 - set2) # 차집합
print(set1 ^ set2) # 여집합 = 합집합 - 교집합
```

실행 결과

{1, 2, 3, 4, 5}
{1, 3, 5, 7, 9}
{1, 2, 3, 4, 5, 7, 9}
{1, 3, 5}
{2, 4}
{2, 4, 7, 9}

값 접근은 리스팅 변환 또는 루프실행
인덱스 접근 안됨

자료타입 - dictionary

키: set 타입-중복안됨, tuple-변경안됨
값: list 타입, 변경가능

```
dict1 = {'하나': 1, '둘': 'two', '파이': 3.14} #키= 문자열 또는 숫자
dict2 = dict({'하나': 1, '둘': 'two', '파이': 3.14})
dict3 = dict([('하나', 1), ('둘', 'two'), ('파이', 3.14)])
dict4 = dict(하나=1, 둘='two', 파이=3.14) #키= 변수형태, 숫자불가
dict5 = dict(([["일",1],["이",2],["삼",3]));

print(dict1)
print(dict2)
print(dict3)
print(dict4)
print(dict5)
```

출력
```
{'하나': 1, '둘': 'two', '파이': 3.14}
{'하나': 1, '둘': 'two', '파이': 3.14}
{'하나': 1, '둘': 'two', '파이': 3.14}
{'하나': 1, '둘': 'two', '파이': 3.14}
{ ' 일 ' : 1, '이': 2, '삼': 3}
```

접근-key 로 value 접근

```
dict1 = dict({'자바': 80, 'PHP': 90, 'HTML': 70})
print(dict1['자바'])
print(dict1.get('자바'))
# print(dict1['파이썬']) dict1.자바 - 오류(닷 접근안됨)
print(dict1.get('파이썬'))
```

실행 결과
```
80
80
None
```

키로 사용되는 값은 immutable 값이어야 한다.
- tuple 가능, 문자리터럴, 숫자리터럴 , 상수

[key] 접근은 없는 요소 접근시 오류발생시키나
.get(key) 는 None 를 리턴

빈 딕셔너리
dict1 = {}

값 추가
dict[key] = value;

딕셔너리 모든요소 삭제

dict.clear() :: {}

딕셔너리 정보추출

```
dict1 = dict({'자바': 80, 'PHP': 90, 'HTML': 70})
print(dict1.keys())
print(dict1.values())
print(dict1.items())
print('HTML' in dict1) # 키 검사
print('파이썬' in dict1)
```

dict_keys(['자바', 'PHP', 'HTML'
dict_values([80, 90, 70]
dict_items([('자바', 80), ('PHP', 90), ('HTML', 70

출력 :: py 3.0 부터는 객체형태리턴(리스트등 타입변환 접근필요 또는 루프접근)
dict.keys() :: dict_keys(['자바', 'PHP', 'HTML'])
dict.values() :: dict_values([80, 90, 70])
dict.items() :: dict_items([('자바', 80), ('PHP', 90), ('HTML', 70)])
True
False

튜플,리스트 중복요소와 딕셔너리 키 및 값 접근

```
dict1 = dict(["일",1],["이",2],["삼",3]));
testTuple = ([1, 2, 3], [4, 5, 6]);
print(testTuple[0][0]);
  # 1

testTuple[0][0]=5;
print(testTuple[0][0]);
  # 5  :: 리스트에 접근했으므로 값 변경 가능

#testTuple[0]=[5, 6, 7];#오류 출력중지, 튜플단 변경 불가

key = dict1.keys();
print(key)
  #dict_keys(['일', '이', '삼'])

value = dict1.values();
print(list(key)[0])
 # 일

print(tuple(value)[0]);
 # 1

for k in key :
    print(k)
 # 일
 # 이
 # 삼
```

함수

```
def functionName(param1, ...):
  run code1 ...
  run code2 ...
  return result
```

```
def sum(a, b):
    print("- 함수 시작 -")
    # return a + b
    print("- 함수 끝 -")
    return a + b


c = sum(1, 2)
print(c)
print(sum(3, 4))
```

실행 결과

```
- 함수 시작 -
- 함수 끝 -
3
- 함수 시작 -
- 함수 끝 -
7
```

파라미터 전달 방법

```
def sub(a, b):
    print(a - b)
sub(1, 2)
sub(a=1, b=2)
sub(b=1, a=2)
```

실행 결과

```
-1
-1
1
```

파라미터 기본값 설정

```
def total(a, b=5, c=10):
    print(a + b + c)
total(1)# 파라미터 순서대로 삽입
total(1, 2)
total(1, 2, 3)
# total(b=2, c=3) 오류
# total(1, 2, 3, 4) 오류
```

실행 결과

```
16
13
6
```

튜플형 가변파라미터

```
def add(*paras):  # * 모두 라는 표현식
    print(paras)
    total = 0
    for para in paras:
        total += para
    return total


print(add(10))
print(add(10, 100))
print(add(10, 100, 1000))
```

실행 결과

```
(10,)
10
(10, 100)
110
(10, 100, 1000)
1110
```

딕셔너리형 가변파라미터

```
def print_map(**dicts):
    for item in dicts.items():
        print(item)


print_map(하나=1)
print_map(one=1, two=2)
print_map(하나=1, 둘=2, 셋=3)
```

실행 결과

```
('하나', 1)
('one', 1)
('two', 2)
('하나', 1)
('둘', 2)
('셋', 3)
```

여러 개의 결괏값 반환하기

두 개 이상의 결괏값을 반환하고 싶다면 셋/튜플/리스트/딕셔너리을 사용해야 합니다.

```
def arith(a, b):
    add = a + b
    sub = a - b
    return add, sub

i, j = arith(10, 1)
print(i) #11
print(j) # 9
```

람다함수식

함수를 더럽게 어렵게 만든거

lambda param1,param2, ... : 파라미터 이용한 표현식

a,b 가 매개변수

함수 본체. auto return 기능 있음

```
lambFun = lambda a, b:a+b;
print(lambFun(3,5));
 # 8
print((lambda a, b:a-b)(1,2))
 # -1


# 오류발생
varfun = def (a,b):
            return a+b;
print(varfun(3,5))

""" 람다는 변수에 담을 수 있으며
익명처리가 가능하다."""
```

삼항연산

true 면 왼쪽, false 면 오른쪽

```
print("True는 참" if False else "True는 거짓")
```

```
year = 2000;
isLeapYear = year%400==0 or year%4==0 and year%100
print('{0}년은 "{1}"입니다.'.format(year,'윤년' if isLeapYear else '평년'))
```

출력

2000년은 "윤년"입니다.

변수범위

지역변수 : 함수 내부 사용

전역변수 : 함수 외부 사용(함수내부에서는 global 변수명으로 접근)

```
def func():
    global global_var
    local_var = "지역 변수"
    print(local_var)
    print(global_var)

global_var = "전역 변수"
func()
print(global_var)
```

실행 결과

지역 변수

전역 변수

전역 변수

클래스 선언

```
class Dog:              # 클래스 선언
    name = "삼식이" # 속성 선언
    age = 3
    breed = "골든 리트리버"

    def bark(self): # 메소드 선언
        print(self.name + "가 멍멍하고 짖는다.")
```

객체생성 사용

```
my_dog = Dog()          # 인스턴스 생성

print(my_dog.breed) # 인스턴스의 속성 접근
my_dog.bark()           # 인스턴스의 메소드 호출
```

실행 결과
골든 리트리버
삼식이가 멍멍하고 짖는다.

```
class Dog:
    def setInfo(self, name):
        self.name = name
    def bark(self):
        print(self.name + "가 멍멍하고 짖는다.")


my_dog = Dog()
my_dog.setInfo("쥐쥐");
my_dog.bark()
```

출력
쥐쥐가 멍멍하고 짖는다.

초기화메소드

```
class Dog: # 클래스 선언
    def __init__(self, name): #초기화, 생성자 메소드
        self.name = name

    def bark(self):
        print(self.name + "가 멍멍하고 짖는다.")


my_dog = Dog("삼식이") # 인스턴스 생성
my_dog.bark() # 인스턴스의 메소드 호출
```

실행 결과
삼식이가 멍멍하고 짖는다.

클래스 변수와 객체변수

클래스변수는 static 메모리 영역으로 객체 공유변수이며
객체변수는 heap 영역으로 객체 특성 변수이다.
self 키워드는 객체 자신을 나타내는 키워드 이며
super 은 부모를 나타내는 키워드 이다.

```
class Dog: # 클래스 선언
    sound = "멍멍" # 클래스 변수 선언

    def __init__(self, name):
        self.name = name # 인스턴스 변수 선언

    def bark(self):
        print(self.name + "가 멍멍하고 짖는다.")


my_dog = Dog("삼식이") # 인스턴스 생성
your_dog = Dog("콩이") # 인스턴스 생성

print(my_dog.sound) # 클래스 변수에 접근
print(my_dog.name) # 인스턴스 변수에 접근

print(your_dog.sound) # 클래스 변수에 접근
print(your_dog.name) # 인스턴스 변수에 접근
```

출력
멍멍
삼식이
멍멍
콩이

상속

```
class Bird:
    def __init__(self):
        self.flying = True

    def birdsong(self):
        print("새소리")

class Sparrow(Bird):
    def birdsong(self): # 메소드 오버라이딩
        print("짹짹")

my_pet = Sparrow()
print(my_pet.flying) # 자신에 없으면 부모를 찾아감
my_pet.birdsong()
```

실행 결과
True
짹짹
tip : 메소드 외부에서 self.flying 선언 할 수없음,
메소드외부는 모두 클래스 영역으로 취급

변수나 메소드의 작명법에 의한 접근제어

C++ 접근 제어자	파이썬
public	멤버 이름에 어떠한 언더스코어(_)도 포함되지 않음. 예) name
private	멤버 이름 앞에 두 개의 언더스코어(__)가 접두사로 포함됨. 예) __name
protected	멤버 이름 앞에 한 개의 언더스코어(_)가 접두사로 포함됨. 예) _name

모듈

import 모듈명 [as custom_name] == from 모듈명 import *
from 모듈명 import 함수, 또는 클래스명, 또는 변수,

파일입출력

```
fp = open('test.txt', 'w')
# mode1 r(read), w(write), a(append)
# mode2 t(text),b(binary)
# mode3 x(exclusive),+(update)
# rtx, rb, …
...
fp.close()
```

x(exclusive) 열고자 하는 파일이 이미 존재하면 파일 개방에 실패함.

+(update) 파일을 읽을 수도 있고 쓸 수도 있도록 개방함.

1. fp.read() 함수 – 모두읽기, 텍스트반환
2. fp.readline() 함수 – EOF 시 None 리턴 , 한줄씩 읽기
3. fp.readlines() 함수 – 모두읽기, 리스트 반환
4. fp.write("₩n추가된 라인입니다.") – 내용추가

자동으로 파일닫기

```
with open('test.txt', 'r') as fp:
    file_data = fp.read()
    print(file_data)
```

예외처리

```
try:
    3 / 0
except IndexError as e:
    print("인덱스가 안맞아요!:",e)
except ZeroDivisionError:
    print("0으로 나누면 안돼요!")
[else]:
    print("예외가 발생하지 않았어요!")
[finally]:
    print("예외에 상관없이 언제나 실행되요!")

print("프로그램이 정상적으로 종료됩니다!")
```

강제예외 발생

```
raise NotImplementedError
```

print

print("i:%d,f:%f,s:%s"%(10,0.14,"hello"))

출력
i:10,f:0.140000,s:hello

```
print('i: %9d, f: %5.2f, s: %7s' % (10, 0.14, "hello"))
```

출력
i: 10, f: 0.14, s: hello

```
print('f: {1}, i: {0}, s: {2}'.format(10, 0.14, "hello"))
```

출력
f: 0.14, i: 10, s: hello

```
print('f: {ff}, i: {ii}, s: {ss}'.format(ii=10, ff=0.14, ss="hello"))
```

출력
f: 0.14, i: 10, s: hello

```
a = 'apple'
b = 'banana'
print('a is {0[a]}, b is {0[b]}'.format(locals()))
```
0 == locals 리턴 딕셔너리

```
print('a is {a}, b is {b}'.format(**locals()))
```
** 딕셔너리 모든 키로 접근

출력
a is apple, b is banana
a is apple, b is banana

```
print(locals())
```

{'__name__': '__main__', '__doc__': None, '__package__': None, '__loader__': <_frozen_importlib_external.SourceFileLoader object at 0x0000020371B56D00>, '__spec__': None, ...}

Python Built in Functions

```
x = abs(-7.25)
```
7.25

```
mylist = [True, True, True]; x = all(mylist)
```
True

```
mylist = [False, True, False]; x = any(mylist)
```
True

```
x = bin(36)
```
0b100100

```
x = bool(1)
```
True

```
x = chr(97)
```
a

```
class Person:
  name = "John"
  age = 36
  country = "Norway"
delattr(Person, 'age')
print(dir(Person))
```

```
x = dict(name = "John", age = 36, country = "Norway")
```
{'name': 'John', 'age': 36, 'country': 'Norway'}

```
x = divmod(5, 2)
```
(2, 1)

```
x = ('apple', 'banana', 'cherry')
y = enumerate(x)
print(list(y))
```
[(0, 'apple'), (1, 'banana'), (2, 'cherry')]

```
x = 'print(55)'
eval(x)
```

55

```
ages = [5, 12, 17, 18, 24, 32]
def myFunc(x):
  if x < 18:
    return False
  else:
    return True
adults = filter(myFunc, ages)
for x in adults:
  print(x)
```

18
24
32

```
 x = float(3)
 print(x)
```

3.0

```
 x = format(0.5, '%')
```

50.000000%

```
 x = format(255, 'x')
```

ff

```
class Person:
  name = "John"
  age = 36
  country = "Norway"
x = getattr(Person, 'age')
```

36

```
class Person:
  name = "John"
  age = 36
  country = "Norway"
x = hasattr(Person, 'age')
```

True

```
x = hex(255)
```

0xff

```
print('Enter your name:')
x = input()
print('Hello, ' + x)
```

```
  x = int(3.5)
```

3

```
  x = iter(["apple", "banana", "cherry"])
  print(next(x))
  print(next(x))
  print(next(x))
```

apple
banana
cherry

```
  mylist = ["apple", "banana", "cherry"]
  x = len(mylist)
```

3

```
  x = list(('apple', 'banana', 'cherry'))
```

['apple', banana', 'cherry']

```
def myfunc(n):
  return len(n)
x = map(myfunc, ('apple', 'banana', 'cherry'))
print(x)
print(list(x))
```

```
<map object at 0x056D44F0>
[5, 6, 6]
```

```
x = oct(12)
```

```
0o14
```

```
x = max(5, 10)
```

```
10
```

```
x = min(5, 10)
```

```
5
```

```
f = open("demofile.txt", "r")
print(f.read())
```

```
x = ord("h")
```

```
104
```

```
x = pow(4, 3)
```

```
64
```

```
x = range(6)
for n in x:
  print(n)
```

```
0
1
2
3
4
5
```

print(*object(s)*, sep=*separator*, end=*end*, file=*file*, flush=*flush*)

Parameter	Description
object(s)	Any object, and as many as you like. Will be converted to string before printed
sep='*separator*'	Optional. Specify how to separate the objects, if there is more than one. Default is ' '
end='*end*'	Optional. Specify what to print at the end. Default is '₩n' (line feed)
file	Optional. An object with a write method. Default is sys.stdout
flush	Optional. A Boolean, specifying if the output is flushed (True) or buffered (False). Default is False

```
print("Hello", "how are you?", sep="---")
```

```
Hello---how are you?
```

```
alph = ["a", "b", "c", "d"]
ralph = reversed(alph)
for x in ralph:
  print(x)
```

```
d
c
b
a
```

```
x = round(5.76543, 2)
```

5.77

```
x = set(('apple', 'banana', 'cherry'))
```

{'cherry', 'banana', 'apple'}

```
class Person:
  name = "John"
  age = 36
  country = "Norway"
setattr(Person, 'age', 40)

x = getattr(Person, 'age')
print(x)
```

40

```
 a = ("b", "g", "a", "d", "f", "c", "h", "e")
 x = sorted(a)
 print(x)
```

['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']

```
 x = str(3.5)
```

3.5

```
 a = (1, 2, 3, 4, 5)
 x = sum(a)
```

15

```
class Parent:
  def __init__(self, txt):
    self.message = txt
  def printmessage(self):
    print(self.message)

class Child(Parent):
  def __init__(self, txt):
    super().__init__(txt)

x = Child("Hello, and welcome!")
x.printmessage()
```

Hello, and welcome!

```
x = tuple(('apple', 'banana', 'cherry'))
```

('banana', 'cherry', 'apple')

```
a = ("John", "Charles", "Mike")
b = ("Jenny", "Christy", "Monica")

x = zip(a, b)
```

(('John', 'Jenny'), ('Charles', 'Christy'), ('Mike', 'Monica'))

2. 다음은 파이썬 코드이다. 출력 결과를 쓰시오.

```
a={'일본','중국','한국'}
a.add('베트남')
a.add('중국')
 a.remove('일본')
a.update(['홍콩','한국','태국'])
print(a)
```

닫기

{'중국','한국','베트남','홍콩','태국'}

9. 다음은 파이썬 소스 코드이다. 출력 결과를 쓰시오.

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

print(lol[0])
print(lol[2][1])
    for sub in lol:
        for item in sub:
            print(item, end = '')
    print()
```

닫기

```
[1,2,3]
7
123
45
6789
```

5. 다음은 파이썬 코드이다. 출력 결과를 쓰시오.

닫기
skiddp

```
class good :
    li = ["seoul", "kyeonggi","inchon","daejeon","daegu","pusan"]

g = good()
str01 = ''

for i in g.li:
        str01 = str01 + i[0]


print(str01)
```

7. 파이썬 비트 연산자 코드 결과

```
a = 100
i=0
result = 0

for i in range(1,3):

result = a >> i
result = result + 1

pirnt(result)
```


a=100
b1100100

14. 다음 파이썬 코드이다. 알맞는 출력값을 쓰시오.

닫기
False

```
a,b = 100, 200
print(a==b)
```

6. 다음은 파이썬 코드에서 출력되는 a와 b의 값을 작성하시오.

```
def exam(num1, num2=2):
    print('a=', num1, 'b=', num2)
exam(20)
```

닫기

a= 20 b= 2

13. 다음은 파이썬 코드이다. 알맞는 출력값을 작성하시오.

```
a = "REMEMBER NOVEMBER"
b = a[:3] + a[12:16];
c = "R AND %s" % "STR";
print(b+c);
```

닫기
REMEMBER AND STR