1월 12일 실습_1

[실습 1]

1. 파일명 : funcLab11.py
2. 구현해야 하는 함수 사양
   함수명 : mydict
   매개변수 : **가변 키워드형**(키=값 형식으로 전달받을 수 있는 아규먼트 개수에 제한이 없다.)
   리턴값 : 1개
   기능 : 아규먼트는 키=값 형식으로 전달되며 몇 개가 전달되든 처리해야 한다.
   아규먼트가 한 개도 **전달되지 않으면 비어있는 딕셔너리를 리턴**한다.
   비어있는 딕셔너리를 생성한 다음 아규먼트로 전달된 키=값 쌍에서 **키 앞에는 my 를 붙여서** 사용한다.
   생성된 딕셔너리를 리턴한다.
3. 다양한 구성으로 키워드 아규먼트를 전달하면서 mydic() 함수를 호출하고 리턴 결과를 화면에 출력한다.

```
def mydict(**p):
    d = {}
    print(p)
    print(type(p))
    if p:
        for i in p:
            d['my' + i] = p[i]
    return d

print(mydict(a=1, b=2, c=3))
print(mydict())
print(mydict(교육='멀티캠퍼스',현재='파이썬'))


def mydic(**kwargs):
    mdic = {}
    for k, v in kwargs.items():
        mdic['my'+k] = v
    return mdic

print(mydic())
print(mydic(name='Lee', age=20))
print(mydic(apple=10, lemon=3, melon=1))
```

{'a': 1, 'b': 2, 'c': 3}
<class 'dict'>
{'mya': 1, 'myb': 2, 'myc': 3}

{}
<class 'dict'>
{}

{'교육': '멀티캠퍼스', '현재': '파이썬'}
<class 'dict'>
{'my교육': '멀티캠퍼스', 'my현재': '파이썬'}

{}
{'myname': 'Lee', 'myage': 20}
{'myapple': 10, 'mylemon': 3, 'mymelon': 1}

[실습 2]

1. 파일명 : funcLab12.py
2. 구현해야 하는 함수 사양
   함수명 : myprint
   매개변수 : **가변 아규먼트1개, 가변 키워드 아규먼트 1개**
   리턴값 : 없음
   기능 : 전달되는 아규먼트의 개수에는 제한이 없다.
   호출시 전달되는 아규먼트의 데이터 타입에도 제한이 없다.
   아규먼트가 전달되지 않으면 "Hello Python"을 출력한다.
   화면 출력은 print() 함수를 사용하며 **개행 처리는 기본이며 변경불가**로 정한다.
3. myprint(10, 20, 30, deco="@", sep="-") 호출시
   @10-20-30@ 출력
   myprint("python", "javascript", "R", deco="$") 호출시
   $python,javascript,R$ 출력
   myprint("가", "나", "다") 호출시
   가,나,다 출력
   myprint(100) 호출시
   100 출력
   myprint(True, 111, False, "abc", deco="&", sep="") 호출시
   &True111Falseabc& 출력
4. 위에 제시된 호출식들을 가지고 호출했을 때 제시된 결과가 출력되면 완성이다.

```
def myprint(*a, **args):
    '''deco = '**'
    s = ','
    if 'deco' in args.keys():
        deco = args['deco']
    if 'sep' in args.keys():
        s = args['sep']'''

    deco = args.get('deco', '**')
    s = args.get('sep', ',')

    result = ''
    if len(a) == 0:
        print("Hello Python")
    else:
        result += deco
        for i in range(len(a)):
            result += str(a[i])
            if i < (len(a)-1):
                result += s
        result += deco
        print(result)
```

```
print("[case1]")
myprint(10, 20, 30, deco="@", sep="-")
myprint("python", "javascript", "R", deco="$")
myprint("가", "나", "다")
myprint(100)
myprint(True, 111, False, "abc", deco="&", sep="")
myprint()

def myprint(*p, **q):
    sep, deco = ',', '**'
    if len(p) != 0:
        if 'sep' in q:
            sep = q['sep']
        if 'deco' in q:
            deco = q['deco']
        p = [str(i) for i in p]
        print(deco, sep.join(p), deco, sep='')
    else:
        print('Hello Python')
print("\n[case2]")
myprint(10, 20, 30, deco="@", sep="-")
myprint("python", "javascript", "R", deco="$")
myprint("가", "나", "다")
myprint(100)
myprint(True, 111, False, "abc", deco="&", sep="")
myprint()


def myprint(*args,**kwargs):
    if(len(args)==0):
        print("Hello Python!!!")
        return
    deco="**"
    sep=","

    if "deco" in kwargs:
        deco=kwargs["deco"]

    if "sep" in kwargs:
        sep=kwargs["sep"]

    print(deco,end='')
    print(*args,sep=sep,end='')
    print(deco)

print("\n[case3]")
myprint(10,20,30,deco="@",sep='-')
myprint("python","javascript","R",deco="$")
myprint("가","나","다")
myprint(100)
myprint(True,111,False,"abc",deco="&",sep='')
myprint()
```

[case1]
@10-20-30@
$python,javascript,R$
가,나,다

100
&True111Falseabc&
Hello Python

[case2]
@10-20-30@
$python,javascript,R$
가,나,다
100
&True111Falseabc&
Hello Python

[case3]
@10-20-30@
$python,javascript,R$
가,나,다
100
&True111Falseabc&
Hello Python!!!