파이썬 - HW3

임베디드스쿨1기

Lv1과정

2020. 08. 10

박하늘

1. Review

1) tuple(), set(), tuple()을 이용해 서로 변이 가능하다.
2) tuple은 const처럼 수정이 불가능 하다.
3) d.items()에서 d에 있는 요소들을 key, map으로 받아온다.

```
d = {"Apple":100, "Orange":200, "Banana":300}
for k, v in d.items():
    print(k,v)
```

```
Apple 100
Orange 200
Banana 300
```

4) dict() 는 key, value로 구성된다. 데이터들을 혼합하여 사용해도 된다.

```
test1 = {'age':40.5,'job':[1,2,3],'name':{'kim':2, 'cho':1}}
print(test1)
print(type(test1))
```

```
{'age': 40.5, 'job': [1, 2, 3], 'name': {'kim': 2, 'cho': 1}}
<class 'dict'>
```

5) copy.deepcopy()는 heap안에 있는 값 자체를 복사해놓음. 얕은 복사 방지

```
import copy
e = [1,2,3]
f = copy.deepcopy(e)
e[0] = 373

print(e)
print(f) #e를 deepcopy해 heap영역에 값 저장해놓음. 따라서 바뀌지 않음
```

```
[373, 2, 3]
[1, 2, 3]
```

1. Review

6) filter()

6-1. 문법: filter(함수, iteration 가능한 자료)

```
numList = [10, 20, 30]
iterList = filter(None, numList)#리스트에 필터를 건다. 필터가 없음
for i in iterList:
    print("item : {0}".format(i))
```

```
item : 10
item : 20
item : 30
```

```
numList = [10, 20, 30]
newList = list(filter(GetBiggerThan20, numList))
print(newList)
print(numList)
```

```
[30]
[10, 20, 30]
```

```
def GetBiggerThan20(i):
    return i > 20

iterList = filter(GetBiggerThan20, numList)#리스트에 필터를 건다. 20을 초과하는 값만 추출
for i in iterList:
    print("item : {0}".format(i))
```

```
item : 30
```

```
iterList = filter(lambda i : i > 20, numList)
for i in iterList:
    print("item: {0}".format(i))
```

```
item: 30
```

7) map()

7-1. map은 주로 데이터 계산할 때, filter는 주로 데이터 걸러낼 때 사용된다.

```
numList = [1,2,3]

retLIst = list(map((lambda i: i+10),numList))
print(retList)
```

```
[(10, 'A'), (20, 'B'), (30, 'C')]
```

1. Review

8) zip()

7-1. 여러개의 sequence형이나 iterator 객체를 묶음
7-2. *는 가변인자.

```
x = [10, 20, 30]
y = "abc"
z = (1.5, 2.5, 3.5)
retList = list(zip(x, y, z))
print(retList)
```

```
[(10, 'a', 1.5), (20, 'b', 2.5), (30, 'c', 3.5)]
```

```
x = [10, 20, 30]
y = ['A', 'B', 'C']
retList = list(zip(x,y))
print(retList)

x2, y2 = zip(*retList)
print(x2)
print(y2)
```

```
[(10, 'A'), (20, 'B'), (30, 'C')]
(10, 20, 30)
('A', 'B', 'C')
```

9) print()보다 join()을 사용하는 경우 속도가 더 빠름.

9-1. memory hierarchy에 의해 print는 for문을 돌면서 계속 print를 찍어냄.
print는 I/O 영역에 저장됨. (I/O 영역:속도느림, 용량큼)
join는 값을 한번에 모아 저장함 (메모리 영역: I/O보다 속도 개선)

```
import time
l = range(100000)

t = time.mktime(time.localtime())

for i in l:
    print(i,)

t1 = time.mktime(time.localtime()) - t

t = time.mktime(time.localtime())
print(", ".join(str(i) for i in l))
t2 = time.mktime(time.localtime()) - t

print("Using for statement timing")
print("take {0} seconds".format(t1))
print("join() timing")
print("take {0} seconds".format(t2))
```

```
Using for statement timing
take 13.0 seconds
join() timing
take 0.0 seconds
```

2. Preview

1)

```
def times(a = 10, b = 20):
    return a * b

print(times())
print(times(5))
```

```
200
240
```

Q1. times(5)의 뜻은 무엇인가? 함수에는 인자가 두개..

2) 사용자 URI연결하는 세가지 방법

```
#URI연결하는 세가지 방법
#server path 와 Port를 지정해준다.
def connectURI(server, port):
    str = "http://" + server + ":" + port
    return str

print(connectURI("문서/HW/Python/ES1기_박하늘_Python_3회차_HW/preview.ipynb", "8888"))
print(connectURI(port = "8888", server = "문서/HW/Python/ES1기_박하늘_Python_3회차_HW/preview.ipynb"))
print(connectURI("문서/HW/Python/ES1기_박하늘_Python_3회차_HW/preview.ipynb", port = "8888"))

name = "문서/HW/Python/ES1기_박하늘_Python_3회차_HW/preview.ipynb"
service = "8888"

print(connectURI(name, service))
```

```
http://문서/HW/Python/ES1기_박하늘_Python_3회차_HW/preview.ipynb:8888
http://문서/HW/Python/ES1기_박하늘_Python_3회차_HW/preview.ipynb:8888
http://문서/HW/Python/ES1기_박하늘_Python_3회차_HW/preview.ipynb:8888
http://문서/HW/Python/ES1기_박하늘_Python_3회차_HW/preview.ipynb:8888
```

2. Preview

3) * 가변 인자에 사용함. (언 패킹(*) 연산자에 의해 수동으로 언 패킹을 지시한 것)

```
print([1, 2, 3])
print(*[1, 2, 3])
```

```
[1, 2, 3]
1 2 3
```

```
def union2(*ar): #가변 인자를 받음
    res = []
    for item in ar:
        for x in item:
            if not x in res:
                res.append(x)
    return res

print(union2("HAM","EGG","POTATO"))
print(union2("test","tdd","essential"))
```

```
['H', 'A', 'M', 'E', 'G', 'P', 'O', 'T']
['t', 'e', 's', 'd', 'n', 'i', 'a', 'l']
```

4) 가변인자를 받아 key와 value로 나눔. 가변인자가 인식되면 먼저 실행 후, 나머지 실행

```
def userURIBuilder(server, port, **user):
    str = "http://"+server +":"+port +"/?"
    for key in user.keys():#가변인자 id='userid',passwd = '1234!'에서 key는 id와 passwd가 됨
        str += key + "=" + user[key] + "&"
    return str
print(userURIBuilder(name, service, id='userid',passwd = '1234!'))
print(userURIBuilder(name, service, id='userid',passwd = '1234!',email='phn159@gmail.com',age = '20'))
```

```
http://문서/HW/Python/ES1기_박하늘_Python_3회차_HW/preview.ipynb:8888/?id=userid&passwd=1234!&
http://문서/HW/Python/ES1기_박하늘_Python_3회차_HW/preview.ipynb:8888/?id=userid&passwd=1234!&email=phn159@gmail.com&age=20&
```

2. Preview

5) Lambda

5-1. 문법 = 인자 : <함수>
5-2. 여러가지 인자를 사용할 수 있음.
5-3. return 구문 없이 콜론 이후의 연산 수행 후 리턴
5-4. 메모리 할당 없이 일회성 (사용 후 해제)

```
g = lambda x, y : x * y
print(g(2,3))
print((lambda x: x*x)(3))
#print(globals()) #현재까지 선언된 변수 보여줌
```

```
6
9
```

6) 함수 속 함수 사용

```
def factorial(x):
    if x == 1:
        return 1
    return x * factorial(x-1)
print(factorial(5))
print(factorial(6))
```

```
120
720
```

```
def hanoi(ndisks, startPeg = 1, endPeg = 3):
    if ndisks:
        hanoi(ndisks - 1, startPeg, 6 - startPeg - endPeg)
        print(startPeg, "PIllar\'s'",ndisks, "disk to", endPeg, "Pillar")
        hanoi(ndisks - 1, 6 - startPeg - endPeg, endPeg)

hanoi(ndisks=3)
```

```
1 PIllar's' 1 disk to 3 Pillar
1 PIllar's' 2 disk to 2 Pillar
3 PIllar's' 1 disk to 2 Pillar
1 PIllar's' 3 disk to 3 Pillar
2 PIllar's' 1 disk to 1 Pillar
2 PIllar's' 2 disk to 3 Pillar
1 PIllar's' 1 disk to 3 Pillar
```

Q2. 동작 의문
결과가 매칭이 안됨..

2. Preview

7) pass = continue

```
for i in range(1,5):
    if(i == 3):
        pass #continue와 같은의미
    else:
        print(i)
```

```
1
2
4
```

```
for i in range(1,5):
    if(i == 3):
        continue
    else:
        print(i)
```

```
1
2
4
```

8) help()

8-1. 함수내 doc를 볼 수 있음
8-2. 함수명.__doc__ 사용자가 추가한 내용 볼 수 있음
8-3. 매개변수, 리턴값 파악할 수 있음

```
help(print)
```

```
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

```
plus.__doc__ = "return the sum of parameter a, b"

help(plus)

def factorial(n):
    """Return the factorial of n, an examct integer >= 0
    ex) factorial(5)
    """
    if x == 1:
        return 1
    return x * factorial(x - 1)

help(factorial)
```

```
FILE
    /home/haneulpark/anaconda3/lib/python3.7/lib-dynload/math.cpython-37m-x86_64-linux-gnu.so


Help on function plus in module __main__:

plus(a, b)

Help on function plus in module __main__:

plus(a, b)
    return the sum of parameter a, b

Help on function factorial in module __main__:

factorial(n)
    Return the factorial of n, an examct integer >= 0
    ex) factorial(5)
```

2. Preview

9) list, tuple, dictionary

```
for element in [1,2,3]: #리스트를 하나씩 불러옴
    print(element)
print("----------------------")
for element in (1,2,3): #튜플을 하나씩 불러옴
    print(element)
print("----------------------")
for key in {'one':1, 'two':2, 'three':3}:#dictionary에서 key만 불러옴
    print(key)
print("----------------------")
for char in "123":
    print(char)
print("----------------------")
```

10) iter()

10-1. 순회 가능한 객체에서 인터레이터 객체를 가져옴

10

() 로 인자를 하나씩 불러옴

```
s = "abc"
it = iter(s) #iter: interator로서 반복기를 말함.
print(it)

print("----------------------")
print(next(it))
print("----------------------")
print(next(it))
print("----------------------")
print(it.__next__())
print("----------------------")
```

```
<str_iterator object at 0x7f2f1c39b250>
----------------------
a
----------------------
b
----------------------
c
----------------------
```

2. Preview

11) yield

11-1. 스택 메모리에 그대로 보관하고 결과값을 호출한 곳에 돌려줌.

11-2. yield는 값을 바로 반환하는 것이 아니고, next()를 통해 iteration이 소진될 때까지 순차적으로 진행되며 값을 반환하고, Stop Iteration 을 만나면 함수가 종료된다.

11-2-1. iterator객체란?

내부에 데이터가 있을때 그 값들을 빠르고 안전하게 나열하는 기능

for in 루프에 리스트나 튜플 실행하고, 문자열 타입도 각각 순차적으로 접근해서 리턴 가능

12) range(stop), range(start,stop), range(start, stop, step)

```
def reverse(data):
    for idx in range(len(data) -1, -1, -1):
        yield data[idx]

for char in reverse('golf'):
    print(char)
```

```
f
l
o
g
```

```
for jlist in [1,2,3]:
    print(jlist)
for jtuple in (4,5,6):
    print(jtuple)
for txt in "PHN":
    print(txt)
```

```
1
2
3
4
5
6
P
H
N
```

2. Preview

13) enumerate

13-1. iterator(리스트, 튜플, 문자열)을 입력받아 인덱스와 함께 나열한다.

13-2. tuple unpacking → a = b, b = a+b

```
def fib():
    a, b = 1, 1
    while 1:
        yield a
        a, b = b, a + b

for i, ret in enumerate(fib()):
    if i < 20: print(i, ret)
    else:
        break
```

```
0 1
1 1
2 2
3 3
4 5
5 8
6 13
7 21
8 34
9 55
10 89
11 144
12 233
13 377
14 610
15 987
16 1597
17 2584
18 4181
19 6765
```

```
for i, season in enumerate(['Spring','Summer','Fall','Winter']):
    print(i, season)
```

```
0 Spring
1 Summer
2 Fall
3 Winter
```

2. Preview

14) 함수 속 함수 사용

```
def fib2(n):
    if n < 2: return n
    else: return fib2(n-1) + fib2(n-2)
print(fib2(20)) # 함수에 함수를 타고 들어가서 계산. 피보나치 수열
```

```
6765
```

3. Homework

3) 문제은행 1-1

```
'''
문제1.
1~10까지 숫자중 짝수를 산출하는 함수를 작성하여 기능을 확인하시오.

'''
##try1
# even_num = []
# for i in range(1,11):
#     if (i % 2 == 0):
#         even_num.append(i)

# print(even_num)

##try2
def even(num): #짝수일때 return함
    if (num % 2 == 0):
        return True

num = []
for i in range(1,11):
    num.append(i)

for i in filter(even, num): #filter(함수, 확인하고자하는 리스트)
    print(i)
```

```
[2, 4, 6, 8, 10]
```

3. Homework

3) 문제은행 1-2, 문제은행 1-3

```
'''
문제2.
#import random과 randint()를 통해 정수형 난수를 생성할 수 있다.
현재 내가 있는 좌표(x, y)를 랜덤으로 생성하도록 한다.
이 상태에서 7개의 랜덤한 좌표를 추가로 생성한다.
내가 잇는 좌표로부터 랜덤한 7개의 좌표가 얼마나 멀리 떨어져 있는지 계측한다.

문제3.
2번 문제에서 가장 가까운 거리에 있는 정보를 획득한다.

'''
```

```
curr_pos: (73, 19) / random_pos[1]: (83, 62) / diff_distance: 44.15
curr_pos: (73, 19) / random_pos[2]: (15, 64) / diff_distance: 73.41
curr_pos: (73, 19) / random_pos[3]: (52, 67) / diff_distance: 52.39
curr_pos: (73, 19) / random_pos[4]: (57, 67) / diff_distance: 50.6
curr_pos: (73, 19) / random_pos[5]: (8, 83) / diff_distance: 91.22
curr_pos: (73, 19) / random_pos[6]: (58, 45) / diff_distance: 30.02
curr_pos: (73, 19) / random_pos[7]: (31, 74) / diff_distance: 69.2
현재 좌표 (73, 19)와 가장 가까운 거리에 있는 좌표는 (58, 45)이고, 거리차이는 30.02 입니다. 6번째 랜덤함수 입니다.
```

3. Homework

```python
import random as rd
import math

MAX = 100
curr_pos = (rd.randint(0, MAX), rd.randint(0, MAX)) #현재 내가 있는 좌표를 튜플로 둠

x_pos = []
y_pos = []
def rand_value(start, end, cnt):#랜덤함수 생성 함수
    for i in range(0, cnt): #7개의 랜덤 함수를 생성.
        x_pos.append(rd.randint(start, end))
        y_pos.append(rd.randint(start, end))
    rand_value = list(zip(x_pos, y_pos)) #x,y를 zip으로 묶음
    return rand_value

dis = []
def distance(x, y):#거리 구하는 함수
    for i in range(len(x)):
        a = curr_pos[0] - x[i]
        b = curr_pos[1] - y[i]
        result = round(math.sqrt(pow(a,2)+pow(b,2)),2) #두점 사이의 거리 구하는 공식 적용
        dis.append(result)
        print("curr_pos: ({0}, {1}) / random_pos[{4}]: ({2}, {3}) / diff_distance: {5}".format(curr_pos[0], curr_pos[1],x[i],y[i],i+1,resul

    return dis

rand_pos = rand_value(0, MAX, 7) #7개의 랜덤 x,y값 생성
x, y = zip(*rand_pos) #가변인자인 이터레이터 객체를 x, y 변수에 저장
diff = distance(x, y) #거리 구하는 함수 적용 (math함수 사용하여 sqrt와 pow사용)

min_dis = min(diff) #계측된 거리중 가장 최소값 추출

for i, ret in enumerate(diff):
    if (ret == min_dis):
        print("현재 좌표 {0}와 가장 가까운 거리에 있는 좌표는 {1}이고, 거리차이는 {2} 입니다. {3}번째 랜덤함수 입니다.".format(curr_pos,rand_pos[
```