함수

무명함수 lambda

* lambda함수는 여러 개의 매개변수를 가질 수 있음.

Ex)

1.lambda함수 사용

mul = lambda arg1, arg2: arg1\*arg2

print("곱셈 값 : ",mul(10,20))

* 값은 200이 나온다.

2.함수 사용

def nestfunc(n):

return lambda x:x\*n

dbl = nestfunc(2)

tri = nestfunc(3)

print(dbl(25))

print(tri(10))

-> 값은 50,30이 나온다. n에 2와 3이 들어가고 x값에 25와 10이 들어가니깐.

3.함수 사용

def cArea(radius):

result = 3.14 \* radius \*\* 2

return result

r = float(input("원의 반지름 : "))

area = cArea(r)

print(area)

-> 반지름에 값을 입력하면 제곱해서 값이 나온다.

4.전역 변수 global

def cArea(radius):

global area

area = 3.14 \* radius \*\* 2

return

area = 0

r = float(input("원의 반지름 : "))

cArea(r)

print(area)

-> 전역변수 global은 선언된 문장 밖에서도 사용 가능하게 만들어 줌.

5.지역변수와 전역변수

def func():

s = "banana"

print(s)

s = "apple"

func()

print(s)

-> banana, apple이 출력된다. 함수 내에서는 지역 변수, 호출 후에는 전역 변수를 참조한다.

6.전역 변수를 함수 안에서 사용

def func():

global s

print(s)

s = 'banana'

print(s)

s='apple'

func()

print(s)

-> apple,banana,banana 출력. 첫 print문 시행 전에 지역 변수를 참조하지 않았으나, s를 global로 선언함으로써 전역 변수 사용

7.예제 – 전역변수와 지역변수 활용

def sub(x,y):

global a

a=7

x,y=y,x

b=3

print(a,b,x,y)

a,b,x,y = 1,2,3,4

sub(x,y)

print(a,b,x,y)

-> 7 3 4 3 / 7 2 3 4 출력.

모듈

모듈 사용하는 법 : 사용할 모듈명.py 로 저장해두고 import 사용할 모듈명 해서 사용.

일반적 방법

import random

사용시 생략할 수 있는 방법

from random import \*

약어 사용 – random을 r로 줄여 부르겠다.

import random as r

순서형 자료구조

리스트 : 여러 개의 데이터가 저장되어 있는 장소 []

튜플 : 리스트와 유사한 순서형 구조 ()

집합 : 수학의 집합 개념과 동일 {}

사전 : 키와 값의 쌍으로 저장. {}

튜플 생성

콤마로 구분된 값을 나열

나열값을 () 사이에 배치

() 생략 가능

Ex1

tup1 = ("eng" , "math")

tup2 = (1, 4, 6)

tup3 = 'a','b

-> tup3을 실행시키면 ()이 생략되었지만 (‘a’,’b’)이렇게 나옴.

Ex2

tup5 = (33)

tup6 = (33,)

tup51 = 'hello'

tup61= 'hello',

print(tup5, type(tup5))

print(tup6, type(tup6))

print(tup51, type(tup51))

print(tup61, type(tup61))

-> tup5는 인트타입, tup6은 튜플타입이다. 하나의 원소만 가지는 튜플을 표시할때는 , 붙여 주기

-> tup51은 스트링타입, 61은 튜플타입이다. 문자열도 마찬가지.

튜플 예제 프로그램

#한 변의 길이가 length인 정사각형의 넓이와 둘레

#정사각형의 넓이와 둘레를 튜플로 반환

def square(length):

area = length \* length

circum = 4 \* length

return (area,circum)

radius = int(input("한 변의 길이 : "))

tupVal = square(radius)

print("넓이 :",tupVal[0], "둘레 :",tupVal[1])

print(type(tupVal))

->print() 안에 ,이게 있어서 원소가 2개가 됨. 그래서 튜플의 형태로 반환되는 것.

튜플 연산

Len((1,2,3)) = 3

3 in (1,2,3) = true

(1,2,3) + (10,20) = (1,2,3,10,20)

(‘ha!’,)\*3 = (‘ha!,’ha!’,’ha!’) / 여기서 콤마가 없으면 그냥 문자열로 ha!ha!ha!가 나옴.

튜플 내장함수

cmp(t1,t2) : 두 튜플의 원소를 비교

len(t) : 튜플 t의 길이(원소의 개수)

min(t) : 튜플 t의 가장 작은 원소

max(t) : 튜플 t의 가장 큰 원소

tuple(seq) : 리스트나 문자열 등을 튜플로 변환

튜플의 갱신과 삭제 -> 변경할 수 없는 순서형 자료구조이고, 원소의 값을 바꾸거나 추가, 삭제 불가능. 하지만 기존의 튜플에 새로운 튜플 할당은 가능하다.

Ex) tup1 = (1,4,5) tup1 = (2,3,4) print(tup) 🡺 (2,3,4) 가장 최근에 할당한 것이 나옴.

튜플 연습문제

var1 = (1,4,9,16,25)

var2 = 'x','y','z'

var1 = var1+var2

print(va1,var2)

->(1, 4, 9, 16, 25, 'x', 'y', 'z') ('x', 'y', 'z')

var1 = (1,4,9,16,25)

var2 = 'x','y','z'

var1 = var1,var2

print(var1,var2)

->((1, 4, 9, 16, 25), ('x', 'y', 'z')) ('x', 'y', 'z')

var1 = (1,4,9,16,25)

var2 = (4,16,1,9)

print(var1==var2)

->False

var1 = (1,4,9,16,25)

var2 = (4,16,1,9)

print(var1<var2)

->True : 첫번째 원소를 비교해서 참 거짓.

집합 생성

콤마로 구분된 값을 {} 사이에 나열

st = {1,2,3,4,5}

print(st,type(st))

-> {1,2,3,4,5}와 집합 클래스임을 알 수 있다.

빈 집합 생성

set()함수 이용. 빈 튜플은 그냥 ().

Set()함수는 리스트나 튜플,문자열을 집합으로 변환시킨다.

st1 = set({1,2,3,4})

print(type(st1))

->집합 클래스임을 알 수 있음.

집합 연산 -> 집합에서는 순서가 의미 없음

len({1,1,2,3}) = 3

3 in {1,2,3} = true

Sum{1,2,10} = 13

{1,2,3} == {2,3,1} = true

집합의 비교 연산

s1 = {1,2,3,4,5,6}

s2 = {1,3,5}

print(s1>s2)

* true

이게 집합에서 비교 연산은, s1이 s2의 원소를 모두 포함하고 있으면 true이다.

집합의 합,교,차집합

sa = {1,3,5}

sb = {1,2,4}

print(sa&sb) -> 교집합

print(sa.union(sb)) -> 합집합

print(sa-sb) -> 차집합

집합의 삭제 – discard,remove

st = {'a','b','c'}

st.discard('a')

print(st)

st.remove('l')

결과

{'c', 'b'}

Traceback (most recent call last):

File "C:\Users\USER\Desktop\사내의 기본\2학년\파이썬\연습.py", line 4, in <module>

st.remove('l')

KeyError: 'l'

discard = 지우려는 원소 있으면 삭제, 없으면 아무일도 없음

remove = 지우려는 원소 있으면 삭제, 없으면 에러.

번외 – del st[0] = 인덱스를 사용하여 삭제 st.pop() = 마지막 원소 반환

집합 연습문제1 – 주어진 문장에서 단어가 몇번 사용되었는지.

stmt = '''sex appeal'''

print('문장:\n',stmt)

words = set()//words라는 빈 집합

wordlist = stmt.split()//스플릿 문자열이있으면 빈 공간을 중심으로 나눠서 리스트로 바꿔 줌.

for word in wordlist:

words.add(word.lower())//소문자로 바꿔서 더해주겠다는말

print('단어 수:',len(word))

print(words)

결과

문장:

sex appeal

단어 수: 6

{'appeal', 'sex'}

사전

키와 값 쌍으로 저장

키를 사용하여 값을 호출

키와 값을 :으로 연결하여 한 쌍으로 함.

여러 개의 키는 ,로 구분

전체는 {}로 감싸서 만듬.

사전 생성

grade = {'eng':87,'math':98}

empty={}

print(grade,type(grade))

print(empty)

결과

{'eng': 87, 'math': 98} <class 'dict'>

{}

함수 dict()를 이용하여 사전 생성.

Ex)

Phone = dict([“부산”,051],(“서울”,02)])

Em = dict(tom = 4112, jack = 3345)

결과

{'부산': 51, '서울': 2}

{'tom': 4112, 'jack': 3345}

* 0은 생략됨. 그리고 개떡같이 입력해도 찰떡같이 :으로 됨.

for문을 활용하여 임의의 키와 값을 나타내는 수식으로 사전 생성

ns = {x:x\*\*2 for x in (3,5,7)}

print(ns)

결과

{3: 9, 5: 25, 7: 49}

사전 검색

키에 대한 값 검색 : 사전이름[key값]

grade = {"eng":87,"math":98}

print(grade["math"])

결과

98

만약 키가 없다면, 오류 발생

사전 추가

grade = {"eng":87,"math":98}

grade[“sci”] = 95

print(grade)

결과

{'eng': 87, 'math': 98, 94: 'sci'} -> sci값이 추가 됨.

반대로, grade[95] = “sci” 이렇게 하면, 94: ‘sci’가 출력됨.

사전 갱신

grade = {"eng":87,"math":98, "sci":95}

grade["sci"] = 94

print(grade)

결과

{'eng': 87, 'math': 98, 'sci': 94} -> 갱신한 값이 출력

사전 삭제

grade = {"eng":87,"math":98, "sci":95}

del grade("math")

print(grade)

하면 math만 삭제

grade = {"eng":87,"math":98, "sci":95}

grade.clear()

print(grade)

clear()사용하면 모두 삭제.

사전 연산

len({"eng":87,"math":98, "sci":95}) = 2

"eng" in {"eng":87,"math":98, "sci":95} = true

{1:"o",2:"t"} == {2:"t",1:"o"} = true

사전 객체의 메소드

dct.clear() -> 사전 dct의 모든 항목을 제거

dct.copy() -> 복사본을 반환

dct.keys() -> 키들의 리스트를 반환

dct.values() -> 값들의 리스트를 반환

dct.items() -> (키,값) 쌍 튜플들의 리스트를 반환

dct.update(dct2) -> dct2의 항목들을 사전dct에 추가

dct.get(key) -> 키 key에 대한 값을 반환

dct.pop(key) -> 키 key에 대한 값을 반환하고 항목 제거

메소드 활용 예제1

grade = {'eng' : 87, 'math':98 }

print("key", grade.keys())

print("value : ",tuple(grade.values)())

결과

key dict\_keys(['eng', 'math'])

values: (87, 98)

메소드 활용 예제2

grade = {'eng' : 87, 'math':98 }

for k,v in grade.items():

print(k,v)

결과

eng 87

math 98

메소드 활용 예제3

grade = {'eng' : 87, 'math':98 }

grd2 = {"kor":87,"sci":95}

grade.update(grd2)

print(grade)

결과

{'eng': 87, 'math': 98, 'kor': 87, 'sci': 95}