첫째마당. 파이썬 기본 문법 익히기

chapter1. 파이썬이란 무엇인가

1.1 파이썬이란?

귀도 반 로섬이 개발한 인터프리터 언어`

공동 작업, 유지보수가 매우 쉽고 편함

1.2 파이썬의 특징

파이썬은 인간다운 언어이다 파이썬은 사람이 생각하는 방식을 그대로 표현할 수 있음

파이썬은 문법이 쉬어 빠르게 배울 수 있다

파이썬은 무료이지만 강력하다 파이썬 = 오픈소스

파이썬은 간결하다 가장 좋은 방법을 사용

파이썬은 프로그래밍을 즐기게 해준다

파이썬은 개발 속도가 빠르다

1.3 파이썬으로 무엇을 할 수 있을까?

파이썬으로 할 수 있는 일 시스템 유틸리티 제작

GUI 프로그래밍 tkInter

C/C++와의 결합 파이썬 = 접착언어

다른 언어와 잘 어울려 결합이 잘됨

웹 프로그래밍

수치 연산 프로그래밍 넘파이 모듈 이용

데이터베이스 프로그래밍 피클 모듈 이용

데이터 분석∙사물인터넷 데이터분석 – 판다스 모듈

라즈베리파이 제어에 이용

파이썬으로 할 수 없는 일 시스템과 밀접한 프로그래밍 영역 대단히 빠른 속도를 요구하는 프로그램∙하드웨어를 직접 건드려야 하는 프로그램

모바일 프로그래밍

1.4 파이썬 설치하기

윈도우에서 파이썬 설치하기

1.5 파이썬 둘러보기

파이썬 기초 실습 준비하기 대화형 인터프리터 인터프리터란 사용자가 입력한 소스코드를 실행하는 환경

입력에 따른 결괏값이 바로 출력됨

파이썬 기초 문법 따라해보기 사칙연산



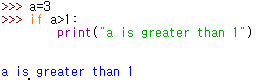
변수에 숫자 대입하고 계산하기



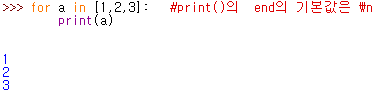
변수에 문자 대입하고 출력하기 print()대신 변수이름인 a만 입력해 출력 가능



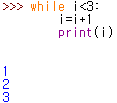
조건문if 들여쓰기 필수!



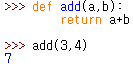
반복문 for



반복문 while



함수



1.6 파이썬과 에디터

에디터 소스코드를 편집할 수 있는 프로그래밍 툴

IDLE로 파이썬 프로그램 작성하기 IDLE 통합개발환경

주석문 작성 #1줄짜리 주성

“””여러줄 주석”””

명령프롬포트로 파이썬 프로그램 작성하기 cd 실행시키려는 py파일이 있는 디렉터리

python 파이썬파일

추천에디터 비주얼 스튜디오 코드

파이참

chapter2. 파이썬 프로그래밍의 기초

2.1 숫자형

숫자형태로 이루어진 자료형

정수 ex) 123

실수 ex) 12.34

8진수, 16진수

숫자형은 어떻게 만들고 사용할까? 정수형 정수

ex) a=123

실수형 실수

ex) a=1.2

a=4.24E10 #4.24\*1010

8진수 0o혹은 0O로 시작하는 숫자

ex) a = 0o177

16진수 0x로 시작하는 숫자

ex) a=0x8ff

숫자형을 활용하기 위한 연산자 사칙연산 ex) >>> a=3

>>> b=4

>>> a+b

7

ex) >>> a\*b

12

ex) >>> a/b

0.75

\*\*연산자 x\*\*y xy 연산

ex) >>> a=3

>>> b=4

>>> a\*\*b

81

%연산자 x%y x/y의 나머지 연산

ex) >>> 7%3

1

//연산자 x//y x/y의 몫 연산

ex) >>> 7//4

1

나 혼자 코딩



2.2 문자형

문자열 문자들의 집합

“”로 둘러 쌓이면 모두 문자열

문자열은 어떻게 만들고 사용할까? 문자열 만드는 방법 “””로 양쪽 둘러싸기

‘로 양쪽 둘러싸기

“””로 양쪽 둘러싸기

‘’’로 양쪽 둘러싸기

문자열 안에 ‘을 넣고 싶을 때 “로 양쪽을 둘러쌀 것

\을 앞에 삽입

ex) 'python\'favorite food is perl'

문자열 안에 “을 넣고 싶을 때 ‘로 양쪽을 둘러쌀 것

\을 앞에 삽입

ex) 'python"favorite food is perl'

여러줄인 문자열을 변수에 대입하고 싶을 때 \n 사용

''', """ 로 문자열 생성

문자열 연산하기 문자열 연결 +연산자 사용 ex) >>> head ="Python"

>>> tail =" is fun!"

>>> head + tail

'Python is fun!'

문자열 곱하기 \*연산자 사용 ex) >>> a="python"

>>> a\*2

'pythonpython'

문자열길이 구하기 len함수 사용 ex) >>> a="Life is too short"

>>> len(a)

17

문자열 인덱싱과 슬라이싱 문자열 인덱싱 0부터 시작

뒤에서부터는 -1부터 시작

문자열 슬라이싱 문자열변수[시작번호:끝번호]

시작번호부터 끝번호-1까지 추출

시작번호 생략 처음부터 추출

끝번호 생략 끝까지 추출

문자열 수정 문자열은 immutable한 문자열 = 수정불가능

슬라이싱 이용

ex) >>> a= a[:1]+'y'+a[2:]

>>> a

'Python'

문자열 포맷팅 문자열 안에 어떤 값을 삽입하는 것

하는 법 1. 숫자 대입 ex) >>> "I eat %d apples" %3

'I eat 3 apples'

2. 문자열 대입 ex) >>> "I eat %s apples" %"five"

'I eat five apples'

3. 변수 대입 ex) >>> number = 3

>>> "I eat %d apples" %number

'I eat 3 apples'

4. 2개 이상의 값 ex) >>> day ="three"

>>> "I ate %d apples. so I was sick for %s days." %(number, day)

'I ate 10 apples. so I was sick for three days.'

문자열 포맷 코드 %s 문자열 어떤 값이던 문자열로 변환함

%c 문자1개

%d 정수

%f 부동소수

%o 8진수

%x 16진수

%% %문자 자체

포맷코드와 숫자 함께 사용 정렬과 공백 ex) >>> "%10s"%"hi"

' hi'

#전체 길이가 10개인 문자열 공간에서 대입되는 값을 오른쪽 정렬

ex) >>> "%-10sjane"%"hi"

'hi jane'

#전체길이가 10개인 문자열 공간에서 대입되는 값을 왼쪽 정렬(-)

소수점 표현 ex) >>> "%0.4f" % 3.42134234

'3.4213'

#.은 소수점포인트, .뒤의 숫자는 소수점 뒤에 나올 숫자의 개수

ex) >>> "%10.4f" % 3.42134234

' 3.4213'

#소수점4번째 자리까지 표시, 길이가 10인 문자열공간에서 오른쪽정렬

format함수를 사용한 포매팅 숫자대입 ex) >>> "i eat {0} apples".format(3)

'i eat 3 apples'

#{0} 🡪 3

문자열 대입 ex) >>> "i eat {0} apples".format("five")

'i eat five apples'

#{0} 🡪 "five"

변수대입하기 ex) >>> num = 5

>>> "i eat {0} apples and {1} mango".format("five", num)

'i eat five apples and 5 mango'

#{0} 🡪 "five" {1} 🡪 num(=5)

이름으로 넣기 ex) >> "i eat {num} apples. so i was sick for {day} days".format(num=10, day=3)

'i eat 10 apples. so i was sick for 3 days'

변수 대입, 인덱스, 이름 혼용가능 ex) >>> num = 5

>>> "i eat {0} apples. so i was {day} for {1} days".format(10, num,day=3)

'i eat 10 apples. so i was 3 for 5 days'

왼쪽 정렬 ex) >>> "{0:<10}".format("hi")

'hi '

#<왼쪽정렬 10총자릿수 10

오른쪽 정렬 ex) >>> "{0:>10}".format("hi")

' hi'

#>오른쪽정렬, 10총자릿수 10

가운데 정렬 ex) >>> "{0:^10}".format("hi")

' hi '

#^가운데정렬 10총자릿수 10

공백채우기 정렬할 때 공백문자 대신 다른 문자로 정렬하기

ex) >>> "{0:=^10}".format("hi")

'====hi===='

#=채워넣을 문자(<>^바로 앞에 위치)

소수점 표현 ex) >>> y=3.42134234

>>> "{0:0.4f}".format(y)

'3.4213'

#소수점 4자리까지 표현

{, } 표현 ex) >>> "{{ and }}".format()

'{ and }'

#연속해서 사용

f 문자열 포맷팅 문자열 앞에 f 접두사를 붙이면 사용가능

ex) >>> name='lee'

>>> age = 23

>>> f'my name is {name}, i am {age}'

'my name is lee, i am 23'

{ } 안에서 연산도 가능

ex) >>> f'my name is {name}, i am {age+1}'

'my name is lee, i am 24'

정렬 ex) >>> f"{'hi':<10}"

'hi '

ex) >>> y=3.42134234

>>> f"{y:<0.4f}"

'3.4213'

#문자열, 숫자가 포맷코드 대신 들어가는 것 빼고 포맷코드를 이용한 정렬과 동일

나 혼자 코딩 답 >>> "{0:!^12}".format("python")

'!!!python!!!'

문자열 관련 함수 문자열 내장 함수

문자열객체 뒤에 .를 붙인 다음 함수 사용

count() 리턴 찾을려는 문자의 개수

문자열.count(‘찾을려는 문자’)

ex) >>> a='hobby'

>>> a.count('b')

2

find() 리턴 찾는 문자의 index

찾는 문자가 없다면 -1

문자열.find(‘찾는 문자’)

ex) >>> a="Python is the best choice"

>>> a.find('b')

14

>>> a.find('k')

-1

index() 리턴 찾는 문자의 index

찾는 문자가 없다면 오류

사용법은 find()와 동일

join() 리턴 문자열변수가 매개변수에 끼워넣어진 문자열

문자열. join(‘abcd’)

ex) >>> ','.join('abcd')

'a,b,c,d'

upper() 리턴 대문자로 바뀐 문자열

문자열.upper()

ex) >>> a='hi'

>>> a.upper()

'HI'

lower() 리턴 소문자로 바뀐 문자열

문자열.lower()

ex) >>> a='HI'

>>> a.upper()

'hi'

lstrip() 리턴 가장 왼쪽에 있는 연속된 공백이 모두 지워진 문자열

문자열.lstrip()

ex) >>> a=" hi "

>>> a.lstrip()

'hi '

rstrip() 리턴 가장 오른쪽에 있는 연속된 공백이 모두 지워진 문자열

문자열.lstrip()

ex) >>> a=" hi "

>>> a.rstrip()

' hi'

strip() 리턴 문자열 양쪽에 있는 연속된 공백이 모두 지워진 문자열

문자열.lstrip()

ex) >>> a=" hi "

>>> a.strip()

'hi'

replace() 리턴 ‘바뀌게 될 문자열’이 ‘바꿀문자열’로 바뀐 문자열

문자열.replace(바뀌게될문자열, 바꿀문자열)

ex) >>> a='life is short'

>>> a.replace('life', 'your leg')

'your leg is short'

split() 리턴 구분자로 잘려서 나온 문자열들의 리스트

문자열.split(‘구분자’, 분할횟수) 구분자 생략가능 생략시, 공백문자

분할횟수 생략가능 생략시, -1 문자열 전체를 자름’

지정시에는 구분자를 생략하지 않거나, maxsplit=분할횟수로 매개변수를 지정해줘야 함

ex) >>> a='life is short'

>>> a.split()

['life', 'is', 'short']

2.3 리스트 자료형

리스트는 어떻게 만들고 사용할까? 요소는 []로 감싸주고, 각 요소는 ,로 구분

리스트명 = [요소1, 요소2, 요소3, …] ex) >>> odd=[1,3,5,7,9]

리스트의 요소는 어떤 자료형이든 가능함

리스트의 인덱싱과 슬라이싱 문자열과 동일

중첩리스트에서의 인덱싱 ex) >>> a= [1,2,3,['a','b','c']]

>>> a[-1][0]

중첩리스트에서의 슬라이싱 ex) >>> a=[1,2,3,['a','b','c'],4,5]

>>> a[3][:2]

['a', 'b']

나 혼자 코딩 답 >>> A=[1,2,3,4,5]

>>> A[1:3]

[2, 3]

리스트 연산하기 리스트 더하기 + ex) >>> a=[1,2,3]

>>> b=[4,5,6]

>>> a+b

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

리스트 반복하기 \* ex) >>> a=[1,2,3]

>>> a\*3

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

리스트 길이 구하기 len() ex) >>> a=[1,2,3]

>>> len(a)

3

리스트 수정∙삭제 리스트에서 값 수정 ex) >>> a=[1,2,3]

>>> a[2]= 4

>>> a

[1, 2, 4]

del 함수로 리스트 요소 삭제 ex) >>> a=[1,2,3]

>>> del a[1]

>>> a

[1, 3]

del a[x] a의 x번째 요소 값 삭제

del a[x:]와 같이 슬라이싱 기법 사용 가능 ex) del a[1:]

리스트 관련 함수 리스트 내장 함수

리스트객체 뒤에 .을 붙이고 함수 사용

append() 리스트의 맨 마지막에 요소를 추가

리스트객체.append(추가되는객체)

ex) >>> a=[1,2,3]

>>> a.append(4)

>>> a

[1, 2, 3, 4]

ex) >>> a=[1,2,3]

>>> a.append([4,5])

>>> a

[1, 2, 3, [4, 5]]

sort() 리스트의 요소를 순서대로 정렬

리스트객체.sort()

ex) >>> a=[1,4,3,2]

>>> a.sort()

>>> a

[1, 2, 3, 4]

ex) >>> a=['a','c','b']

>>> a.sort()

>>> a

['a', 'b', 'c']

reverse() 현재 상태의 리스트를 그대로 역순으로 뒤집음

리스트객체.reverse()

ex) >>> a=['a','c','b']

>>> a.reverse()

>>> a

['b', 'c', 'a']

index() 리스트의 찾는 값이 있으면 찾는 값의 위치 값을 돌려줌

리턴 찾는 값의 인덱스

리스트객체.index(찾는값)

ex) >>> a=[1,2,3]

>>> a.index(3)

2

존재하지 않는 값을 찾으면 오류발생

insert() 리스트에 요소를 삽입

리스트객체.insert(인덱스, 삽입되는객체)

ex) >>> a=[1,2,3]

>>> a.insert(0,4)

>>> a

[4, 1, 2, 3]

remove() 리스트에서 첫번째로 나오는 삭제하려는 요소를 삭제

리스트객체.remove(삭제하려는요소)

ex) >>> a=[1,2,3,1,2,3]

>>> a.remove(3)

>>> a

[1, 2, 1, 2, 3]

pop() 리스트객체의 인덱스번째 요소를 반환하고 삭제

리스트객체.pop(인덱스) 인덱스 생략시, 리스트 객체의 맨 마지막 인덱스

리턴 리스트객체의 지정한 요소

ex) >>> a=[1,2,3]

>>> a.pop()

3

>>> a

[1, 2]

ex) >>> a=[1,2,3]

>>> a.pop(1)

2

count() 리스트 객체에서의 요소의 개수 세기

리스트객체.count(세려는요소)

ex) >>> a=[1,2,3,1]

>>> a.count(1)

2

extend() 리스트객체에 리스트객체를 붙여 확장시킴

리스트객체.extend(붙여지는리스트객체)

ex) >>> a=[1,2,3]

>>> a.extend([4,5])

>>> a

[1, 2, 3, 4, 5]

# +=와 동일

2.4 튜플 자료형

리스트와 거의 동일

리스트와 다른 점 초기화 시. ( )로 둘러쌈

튜플은 수정이 불가능함

ex) >>> t1=()

>>> t2=(1,) #1개의 요소만 가질때에는 요소 뒤에 ,를 꼭 붙여줘야 함 (,를 붙이지 않으면 int로 인식)

>>> t3=(1,2,3)

>>> t4=1,2,3

>>> t5=('a','b',('ab','cd'))

튜플의 요소값 삭제, 변경 시 오류 발생

튜플 다루기 인덱싱하기 리스트, 문자열처럼 인덱싱 가능

ex) >>> t1=(1,2,'a','b')

>>> t1[0]

1

슬라이싱하기 리스트, 문자열처럼 슬라이싱 가능

ex) >>> t1=(1,2,'a','b')

>>> t1[1:]

(2, 'a', 'b')

튜플 더하기 +이용

ex) >>> t1=(1,2,'a','b')

>>> t2=(3,4)

>>> t1+t2

(1, 2, 'a', 'b', 3, 4)

튜플 곱하기 \*이용

ex) >>> t2=(3,4)

>>> t2\*3

(3, 4, 3, 4, 3, 4)

튜플 길이 구하기 len()이용

ex) >>> t1=(1,2,'a','b')

>>> len(t1)

4

나 혼자 코딩 답 >>> (1,2,3)+ (4,)

(1, 2, 3, 4)

2.5 딕셔너리 자료형

대응관계를 나타내는 자료형

다른 언어의 연관배열, 해시

key와 value를 한 쌍으로 가지는 자료형

인덱스로 값을 찾지 않고, key로 값을 찾음

딕셔너리는 어떻게 만들까? { Key1:Value1, Key2:Value2, Key3:Value3 }

ex) >>> dic= {'name':'pey', 'phone':119993323, 'birth':'1118'}

Key: Value 형태 Key 변하지 않는 값 사용

Value 변하는 값, 변하지 않는 값 모두 사용

,로 구분

딕셔너리 쌍 추가, 삭제 딕셔너리 쌍 추가 딕셔너리[key] = value

ex) >>> a={1:'a'}

>>> a[2]='b'

>>> a

{1: 'a', 2: 'b'}

딕셔너리 쌍 삭제 del 딕셔너리[key]

ex) >>> dic= {'name':'pey', 'phone':119993323, 'birth':'1118'}

>>> del dic['name']

>>> dic

{'phone': 119993323, 'birth': '1118'}

딕셔너리 사용 방법 key를 사용해 value 얻기 딕셔너리는 인덱스가 없으므로, 인덱스로 value 얻기 불가

딕셔너리[key]

ex) >>> grade = {'pey':10, 'julliet':99}

>>> grade['pey']

10

딕셔너리를 만들 때 주의사항 중복되는 key값을 설정하면 하나(맨 마지막값)를 제외한 나머지 값들 모두 무시

ex) >>> a={1:'a', 1:'b'}

>>> a

{1: 'b'}

딕셔너리 관련 함수 keys() 리턴 딕셔너리의 key값만을 모은 dict\_keys객체

list객체 아님 리스트메서드 사용을 위해 리스트로 변환 필요

딕셔너리객체.keys()

ex) >>> dic= {'name':'pey', 'phone':119993323, 'birth':'1118'}

>>> dic.keys()

dict\_keys(['name', 'phone', 'birth'])

values() 리턴 딕셔너리의 value값만을 모은 dict\_values객체

list객체 아님 리스트메서드 사용을 위해 리스트로 변환 필요

딕셔너리객체.values()

ex) >>> dic= {'name':'pey', 'phone':119993323, 'birth':'1118'}

>>> dic.values()

dict\_values(['pey', 119993323, '1118'])

items() 리턴 key와 value의 쌍을 튜플로 묶은 값을 요소로 가지는 dict\_items객체

딕셔너리객체.items()

ex) >>> dic= {'name':'pey', 'phone':119993323, 'birth':'1118'}

>>> dic.items()

dict\_items([('name', 'pey'), ('phone', 119993323), ('birth', '1118')])

clear() 딕셔너리 내의 모든 요소 삭제

딕셔너리객체.clear()

ex) >>> dic= {'name':'pey', 'phone':119993323, 'birth':'1118'}

>>> dic.clear()

>>> dic

{}

get() 리턴 딕셔너리 내의 Key값에 대응하는 Value

key값에 대응하는 value가 없다면, None 발생 – 기본값 매개변수가 있다면 기본값 반환

딕셔너리객체.get(Key, 기본값)

딕셔너리객체[Key]와의 차이 key값이 없다면 오류발생

ex) >>> dic= {'name':'pey', 'phone':119993323, 'birth':'1118'}

>>> dic.get('name')

'pey'

in 해당 Key가 딕셔너리 안에 있는지 조사

리턴 true Key가 딕셔너리가 있음

false Key가 딕셔너리가 없음

ex) >>> dic= {'name':'pey', 'phone':119993323, 'birth':'1118'}

>>> 'name' in dic

True

2.6 집합 자료형

set() 키워드로 선언

괄호 안에 리스트∙문자열 입력 ex) >>> s1=set([1,2,3])

>>> s1

{1, 2, 3}

ex) >>> s2=set("Hello")

>>> s2

{'o', 'e', 'l', 'H'}

집합자료형의 특징 중복을 허용하지 않음

순서가 없음 🡪 인덱싱으로 값을 얻을 수 없음

인덱싱으로 값을 얻으려면 리스트, 튜플로 변환해야함

교집합, 합집합, 차집합 구하기 교집합 & 연산자 사용 ex) >>> s1=set([1,2,3,4,5,6])

>>> s2=set([4,5,6,7,8,9])

>>> s1&s2

{4, 5, 6}

intersection 함수 ex) >>> s1=set([1,2,3,4,5,6])

>>> s2=set([4,5,6,7,8,9])

>>> s1.intersection(s2)

{4, 5, 6}

합집합 | 연산자 사용 ex) >>> s1=set([1,2,3,4,5,6])

>>> s2=set([4,5,6,7,8,9])

>>> s1|s2

{1, 2, 3, 4, 5, 6, 7, 8, 9}

union 함수 ex) >>> s1=set([1,2,3,4,5,6])

>>> s2=set([4,5,6,7,8,9])

>>> s1.union(s2)

{1, 2, 3, 4, 5, 6, 7, 8, 9}

차집합 - 연산자 사용 ex) >>> s1=set([1,2,3,4,5,6])

>>> s2=set([4,5,6,7,8,9])

>>> s1-s2

{1, 2, 3}

difference 함수 ex) >>> s1=set([1,2,3,4,5,6])

>>> s2=set([4,5,6,7,8,9])

>>> s2.difference(s1)

{8, 9, 7}

집합자료형 관련 함수 add() 값 1개 추가

집합객체.add(추가할값)

ex) >>> s1=set([1,2,3])

>>> s1.add(4)

>>> s1

{1, 2, 3, 4}

update() 값 여러 개 추가

집합객체.update(추가할값들)

ex) >>> s1=set([1,2,3])

>>> s1.update([4,5,6])

>>> s1

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

remove() 특정 값 제거

집합객체.remove(제거하려는값)

제거하려는 값이 없다면 오류 발생

ex) >>> s1.remove(2)

>>> s1

{1, 3}

2.7 불 자료형

참 – True

거짓 – False

조건문의 반환값

자료형의 참과 거짓 False 빈문자열, 빈리스트, 빈튜플, 빈딕셔너리, 0, None

True False가 아닌 모든 것

불 연산 bool() 리턴 값이 True라면 True 리턴

값이 False라면 False 리턴

bool(값)

ex) >>> bool("python")

True

ex) >>> bool([])

False

2.8 자료형의 값을 저장하는 공간, 변수

변수 만들기 =연산자 변수이름 = 변수에 저장할 값

변수란? 변수는 객체를 가리키는 것

자료형(객체)이 자동으로 메모리에 생성됨 🡪 변수는 메모리를 가리키게 됨

id() 메모리주소 확인 함수 리턴 객체의 메모리상 주소

id(변수)

리스트를 복사할 때 얕은 복사 >>> a=[1,2,3]

>>> b=a

a, b는 완전히 동일 >>> a is b

True

a수정은 곧 b수정 >>> a[1]=4

>>> b

[1, 4, 3]

a, b는 모두 동일한 리스트 객체를 가리키기 때문

깊은 복사 [:] 사용 ex) >>> a=[1,2,3]

>>> b=a[:]

>>> a[1] = 4

>>> b

[1, 2, 3]

copy모듈 사용 ex) >>> from copy import copy

>>> a=[1,2,3]

>>> b=copy(a)

>>> a is b

False

변수를 만드는 여러가지 방법 튜플 ex) >>> a,b = ('python','life')

리스트 ex) >>> [a,b] = ['python','life']

여러 개 변수에 같은 값 대입 ex) a = b = 'python' #이 경우, a와 b는 얕은 복사

치환 ex) >>> a=3

>>> b=5

>>> a,b = b,a

>>> a

5

>>> b

3

나 혼자 코딩 답 False

첫째마당 연습문제

[C:\Users\Lee\OneDrive\공부\전공독서\프로그래밍 언어\점프 투 파이썬(이지스퍼블리싱, 박응용)\첫째마당\연습문제](file:///C:\Users\Lee\AppData\Roaming\Microsoft\Word\첫째마당\연습문제)

chapter3. 프로그램의 구조를 쌓는 제어문

3.1 if문

조건을 판단하여 해당 조건에 맞는 상황을 수행

ex) >>> money=True

>>> if money:

print("택시를 타고가라")

else:

print("걸어가라")

택시를 타고가라

if문의 기본 구조 참 if 블록 수행

거짓 else블록 수행

if 조건문:

수행문1

수행문2

else:

수행문a

수행문b

들여쓰기 if문에 속하는 모든 문장에 들여쓰기 필수

들여쓰기는 언제나 같은 너비로 해야됨

(tab)1번보단 (spacebar) 4번 권장

조건문이란 무엇인가 조건문 참과 거짓을 판단하는 문장

비교연산자 x<y x가 y보다 작다

x>y x가 y보다 크다

x==y x와 y가 같다

x!=y x가 y보다 다르다

x>=y x가 y보다 크거나 같다

x<=y x가 y보다 작거나 같다

논리연산자 x or y x와 y 둘 중 하나가 참이다

x and y x와 y 모두 참이다

not x x가 거짓이면 참이다, 참이면 거짓이다

in연산자 멤버연산자(멤버접근연산자 헷갈리지 말 것)

x in 리스트, 튜플, 문자열 x가 리스트, 튜플, 문자열에 있으면 True 리턴, 없으면 False 리턴

pass 아무 일도 하지 않도록 설정하고 싶을 때, pass 사용

pass가 수행되면 아무 것도 하지 않음

나 혼자 코딩 >>> card\_in\_pocket= True

>>> if card\_in\_pocket:

print("bus")

else:

print("walk")

다양한 조건을 판단하는 elif 다중 조건 판단 가능

이전 조건문이 거짓일 때 수행

개수에 제한없이 사용가능

형식 if 조건문1:

수행문1

수행문2

elif 조건문2:

수행문1

수행문2

elif 조건문3:

수행문1

수행문2

else:

수행문a

수행문b

ex) >>> pocket=['paper', 'cellphone']

>>> card = True

>>> if 'money' in pocket:

print("택시를 타고가라")

elif card:

print("택시를 타고가라")

else:

print("걸어가라")

택시를 타고가라

조건부 표현식 형식 변수 = “조건식이 참이면 대입되는 값” if 조건식 else “조건식이 거짓이면 대입되는 값”

ex) >>> score= 70

>>> message = "success" if score >= 60 else "failure"

>>> message

'success'

3.2 while문

while문의 기본 구조 반복해서 문장을 수행해야 하는 경우, while문 수행

형식 while 조건문:

수행할 문장1

수행할 문장2

ex) >>> treehit = 0

>>> while treehit < 10:

treehit = treehit+1

print("나무를 %d번 찍었습니다." %treehit)

if treehit ==10:

print("나무 넘어갑니다")

나무를 1번 찍었습니다.

나무를 2번 찍었습니다.

나무를 3번 찍었습니다.

나무를 4번 찍었습니다.

나무를 5번 찍었습니다.

나무를 6번 찍었습니다.

나무를 7번 찍었습니다.

나무를 8번 찍었습니다.

나무를 9번 찍었습니다.

나무를 10번 찍었습니다.

나무 넘어갑니다

while문 만들기 >>> prompt = """

1. Add

2. Del

3. List

4. Quit

Enter number: """

>>> number=0

>>> while number != 4:

print(prompt)

number = int(input())

1. Add

2. Del

3. List

4. Quit

Enter number:

3

1. Add

2. Del

3. List

4. Quit

Enter number:

2

1. Add

2. Del

3. List

4. Quit

Enter number:

1

1. Add

2. Del

3. List

4. Quit

Enter number:

4

while문 강제로 빠져나가기 break 강제로 while문 탈출

ex)

coffee=10

while True:

    money = int(input("돈을 넣어 주세요: "))

    if money == 300:

        print("커피를 줍니다")

        coffee = coffee-1

    elif money > 300:

        print("거스름돈 %d를 주고 커피를 줍니다" %(money-300))

        coffee=coffee-1

    else:

        print("돈을 다시 돌려주고 커피를 주지 않습니다")

        print("남은 커피의 양은 %d개입니다." %coffee)

    if coffee ==0:

        print("커피가 다 떨어졌습니다. 판매를 중지합니다.")

        break

while문의 맨 처음으로 돌아가기 while문을 빠져나가지 않고 while문의 맨 처음(조건문)으로 돌아가게 만들고 싶은 경우

continue 문

ex) >>> a=0

>>> while a<10:

a=a+1

if a%2==0: continue

print(a)

1

3

5

7

9

나 혼자 코딩 >>> a=1

>>> while a<=10:

if a%3!=0: print(a)

a+=1

1

2

4

5

7

8

10

무한루프 무한히 반복

형식 while True:

수행문1

수행문2

…

ex) >>> while True:

print("Ctrl+C를 눌러야 while문을 빠져나갈 수 있습니다.")

Ctrl+C를 눌러야 while문을 빠져나갈 수 있습니다.

Ctrl+C를 눌러야 while문을 빠져나갈 수 있습니다.

Ctrl+C를 눌러야 while문을 빠져나갈 수 있습니다.

Traceback (most recent call last):

File "<pyshell#54>", line 2, in <module>

print("Ctrl+C를 눌러야 while문을 빠져나갈 수 있습니다.")

KeyboardInterrupt

3.3 for문

반복문

for문의 기본 구조 첫번째요소부터 마지막 요소까지 차례로 변수에 대입됨 ex)리스트, 튜플, 문자열

형식 for 변수 in 리스트(혹은 튜플, 문자열):

수행문1

수행문2

ex) >>> test\_list = ['one', 'two', 'three']

>>> for i in test\_list:

print(i)

one

two

three

ex) >>> a=[(1,2),(3,4),(5,6)]

>>> for (first, last) in a:

print(first+last)

3

7

11

continue문 continue문을 만나면 for문의 처음으로 돌아가게 됨

ex)

marks=[90,25,67,45,80]

number = 0

for mark in marks:

    number = number +1

    if mark < 60: continue

    print("%d번 학생 축하합니다. 합격입니다." %number)

range함수 숫자 range객체를 자동으로 만들어주는 함수

리턴 range 객체

사용 range(시작숫자, 끝숫자, 간격) 끝 숫자는 포함안됨

간격은 생략가능, 생략시 1

ex)

marks = [90,25,67,45,80]

for number in range(len(marks)):

    if marks[number] < 60: continue

    print("%d번 학생 축하합니다. 합격입니다." %(number+1))

나 혼자 코딩 >>> result=0

>>> for i in range(1,101):

result+=i

>>> result

5050

리스트 내포 사용하기 리스트 안에 for문을 포함

형식 [표현식 for 항목 in 반복가능객체 if 조건]

if조건은 생략 가능

ex) >>> a = [1,2,3,4]

>>> res = [num\*3 for num in a]

>>> res

[3, 6, 9, 12]

ex)

연습문제

[C:\Users\Lee\OneDrive\공부\전공독서\프로그래밍 언어\점프 투 파이썬(이지스퍼블리싱, 박응용)\첫째마당\chapter1\_3장연습문제](file:///C:\Users\Lee\OneDrive\공부\전공독서\프로그래밍%20언어\점프%20투%20파이썬(이지스퍼블리싱,%20박응용)\첫째마당\chapter1_3장연습문제)

chapter4. 프로그램의 입력과 출력은 어떻게 해야할까?

4.1 함수

함수를 사용하는 이유 똑같은 내용을 반복하기 용이

프로그램흐름을 알기 쉬움

파이썬 함수의 구조 def 키워드 사용

return 함수의 결과값을 돌려주는 키워드

형식 def 함수이름(매개변수):

수행문1

수행문2

return문

ex) >>> def add(a,b):

return a+b

매개변수와 인수 매개변수 함수에 입력으로 전달된 값을 받는 변수 ex) def add(a,b)

인수 함수에 호출할 때 전달하는 입력값 ex) add(1,3)

입력값과 결괏값에 따른 함수의 형태 매개변수O 리턴값O 형식 def 함수이름(매개변수):

수행문1

return문

사용법 리턴값을 받을 변수 = 함수이름(인수)

매개변수X 리턴값O 형식 def 함수이름():

수행문1

return문

사용법 리턴값을 받을 변수 = 함수이름()

매개변수O 리턴값X 형식 def 함수이름(매개변수):

수행문1

사용법 함수이름(인수)

None값 반환

매개변수X 리턴값X 형식 def 함수이름():

수행문1

사용법 함수이름()

매개변수 지정하여 호출하기 함수 호출 시에 매개변수 지정

순서에 상관없이 지정

형식 함수이름(매개변수=인수, )

ex) >>> def div(a,b):

return a/b

>>> div(a=10,b=2)

5.0

>>> div(b=10,a=2)

0.2

입력값이 몇 개나 될지 모를 때 가변매개변수

매개변수 이름 앞에 \*를 붙임 🡪 인수들을 튜플로 만듬

형식 def 함수이름(\*매개변수):

수행문1

ex) >>> def add\_many(\*args):

result = 0

for i in args:

result+=i

return result

>>> add\_many(1,2,3)

6

ex) >>> def add\_mul(choice, \*args):

if choice == "add":

result = 0

for i in args:

result+=i

elif choice == "mul":

result = 1

for i in args:

result\*=i

return result

>>> add\_mul("add", 1,2,3,4)

10

>>> add\_mul("mul",1,2,3,4)

24

키워드 피라미터 매개변수 앞에 \*\*을 붙임

형식 def 함수이름(\*\*매개변수):

수행문1

매개변수가 딕셔너리가 됨

ex) >>> def print\_kwargs(\*\*kwargs):

print(kwargs)

>>> print\_kwargs(a=123)

{'a': 123}

함수의 결괏값은 언제나 하나 리턴문으로 반환할 수 있는 결괏값은 언제나 하나

ex) >>> def add\_and\_mul(a,b):

return a+b, a\*b

>>> res = add\_and\_mul(10,2)

>>> res

(12, 20) #res는 튜플

return문 단독으로 사용해 아무 값도 리턴하지 않고 함수를 끝낼 수 있음

매개변수에 초깃값 설정하기 매개변수에 미리 값을 넣어두기

단, 초깃값이 설정된 매개변수는 맨 마지막에 둬야함

ex) >>> def say\_myself(myname, old, man=True):

...

ex) >>> def say\_myself(myname, old=0, man): #syntaxError

...

함수 안의 선언한 변수의 범위 함수안에 선언한 변수는 함수 안에서만 사용 가능

global 키워드 함수 안에서 함수 밖에 선언된 변수를 사용하기 위한 키워드

형식 global 변수

ex) a=1

def vartest():

    global a

    a+=1

vartest()

print(a)

lamda 함수를 생성할 때 사용하는 키워드

간단한 함수를 만들 때, def를 사용할 수 없을 때 사용

형식 lambda 매개변수 : 매개변수를 사용한 표현식

ex) >>> add = lambda a, b : a+b

>>> add(2,3)

5

4.2 사용자 입력과 출력

사용자 입력 사용자가 입력한 값을 어떤 변수에 대입 input(“질문내용”) 입력값을 문자열로 취급

ex) >>> a=input()

Life is too short, you need python

>>> a

'Life is too short, you need python'

ex) >>> number = input("숫자를 입력하세요: ")

숫자를 입력하세요: 3

>>> number

'3'

print 자세히 알기 “”로 둘러싸인 문자열은 +연산과 동일 ex) >>> print("life" "is" "too short")

lifeistoo short

문자열 띄어쓰기는 콤마로 ex) >>> print("Life", "is", "too short")

Life is too short

한 줄에 결괏값 출력 결괏값을 계속 이어서 출력하려면 매개변수 end 사용

ex) >>> for i in range(10):

print(i, end=' ')

0 1 2 3 4 5 6 7 8 9

4.2 파일 읽고 쓰기

파일을 통한 입출력 방법

파일생성하기 open() 형식 파일객체 = open(파일이름, 파일열기모드)

파일열기 모드 r 읽기모드

w 쓰기모드

원래 있던 파일 삭제, 재생성

a 추가모드

ex) >>> f= open("C:/Users/Lee/OneDrive/공부/전공독서/프로그래밍 언어/점프 투 파이썬(이지스퍼블리싱, 박응용)/첫째마당/새파일.txt ", 'w')

>>> f.close()

나 혼자 코딩 답 f= open("C:/doit/복습.txt", 'w')

f.close()

파일을 쓰기모드로 열어 출력값 적기

ex) f = open("C:/Users/Lee/OneDrive/공부/전공독서/프로그래밍 언어/점프 투 파이썬(이지스퍼블리싱, 박응용)/첫째마당/새파일.txt" , 'w')

for i in range(1,10):

    data = "%d번째 줄입니다.\n" %i

    f.write(data)

f.close()

프로그램의 외부에 저장된 파일을 읽는 여러가지 방법 외부 파일을 읽어들여 프로그램에서 사용

readline함수 리턴 파일에서 읽은 한줄 = \n까지의 문자열

더 이상 읽을 문자열이 없을 때 None리턴

형식 파일객체.readline()

ex) f = open("C:/Users/Lee/OneDrive/공부/전공독서/프로그래밍 언어/점프 투 파이썬(이지스퍼블리싱, 박응용)/첫째마당/새파일.txt" , 'r')

while True:

    line = f.readline()

    if not line: break

    print(line)

f.close()

readlines함수 리턴 파일의 모든 줄을 요소로 갖는 리스트

형식 파일객체.readlines()

ex) f = open("C:/Users/Lee/OneDrive/공부/전공독서/프로그래밍 언어/점프 투 파이썬(이지스퍼블리싱, 박응용)/첫째마당/새파일.txt" , 'r')

lines = f.readlines()

for line in lines:

    print(line)

f.close()

read함수 리턴 파일의 전체 내용의 문자열

형식 파일객체.read()

ex) f = open("C:/Users/Lee/OneDrive/공부/전공독서/프로그래밍 언어/점프 투 파이썬(이지스퍼블리싱, 박응용)/첫째마당/새파일.txt" , 'r')

data = f.read()

print(data)

f.close()

파일에 새로운 내용 추가하기 파일을 추가모드로 열기

원래 가지고 있던 내용 바로 다음부터 결괏값을 적기 시작

ex) f = open("C:/Users/Lee/OneDrive/공부/전공독서/프로그래밍 언어/점프 투 파이썬(이지스퍼블리싱, 박응용)/첫째마당/새파일.txt" , 'a')

for i in range(11,20):

    data = "%d번째 줄입니다.\n" %i

    f.write(data)

f.close()

with문 함께 사용하기 파일을 열고 닫는 것을 자동으로 처리

with블록을 벗어나는 순간 열린 파일 객체가 자동으로 close됨

형식 with open(“파일”, “모드”) as 파일객체:

수행문

ex) >>> with open("foo.txt", "w") as f:

f.write("Life is too short, you need python")

sys모듈 sys모듈을 이용해 매개변수를 직접 줌

sys.argv 명령창에 입력한 인수를 모은 리스트

cmd에서 실행했다면 argv[0] = 파일 이름

argv[1]부터 인수들이 옴

5.1 클래스

클래스는 왜 필요한가 클래스 독립적인 값을 유지

클래스와 객체 클래스 똑같은 무엇인가를 계속해서 만들어 낼 수 있는 설계도면

객체 클래스로 만든 피조물

객체마다 고유한 성격 = 서로 영향을 주지 않음

메서드 클래스안에 구현된 함수

self 매서드의 0번째 매개변수

self에는 메서드를 호출한 객체가 자동으로 들어감

C++의 \*this처럼 사용

객체변수(멤버변수)에 접근 시 이용

클래스로 만든 객체의 객체변수는 다른 객체의 객체변수와 상관없이 독립적인 값을 유지

ex) class FourCal:

    def setdata(self, first, second):

        self.first = first

        self.second = second

    def add(self):

        result = self.first + self.second

        return result

    def mul(self):

        result = self.first\*self.second

        return result

    def sub(self):

        result = self.first\*self.second

        return result

    def div(self):

        result = self.first/self.second

        return result

>>> a.add()

30

>>> a.mul()

200

생성자 생성자 구현 객체에 초깃값을 설정해야 할 필요가 있을 때

객체 생성될 때 자동으로 호출되는 메서드

\_\_init\_\_ 파이썬 메서드 이름으로 \_\_init\_\_

객체가 생성되는 시점에서 자동으로 호출

ex) class FourCal:

    def \_\_init\_\_(self, first, second):

        self.first = first

        self.second = second

    def setdata(self, first, second):

        self.first = first

        self.second = second

    def add(self):

        result = self.first + self.second

        return result

    def mul(self):

        result = self.first\*self.second

        return result

    def sub(self):

        result = self.first\*self.second

        return result

>>> a=FourCal(4,3)

>>> a.add()

7

클래스의 상속 어떤 클래스를 만들 때 다른 클래스의 기능을 물려받을 수 있게 만드는 것

형식 class 클래스이름(상속할 클래스이름)

ex) >>> class MoreFourCal(FourCal):

pass

상속을 해야하는 이유 기존 클래스가 라이브러리 형태로 제공되거나 수정이 허용되지 않는 상황에 상속 사용

ex) >>> class MoreFourCal(FourCal):

def pow(self):

result = self.first\*\*self.second

return result

메서드 오버라이딩 부모클래스의 메서드를 동일한 이름으로 다시 작성

부모클래스의 메서드 대신 파생클래스의 메서드(오버라이딩된 메서드) 호출

ex) >>> class SafeFourCal(FourCal):

def div(self):

if self.second == 0:

return 0

else:

return self.first/self.second

클래스 변수 객체변수와 다름

self.이 붙지 않음

클래스 안에 함수를 선언하는 것과 동일

클래스 안에 변수 선언

사용 클래스이름.클래스변수

클래스변수 값은 객체끼리 공유

ex) >>> class Family:

lastname = "김"

>>> a=Family()

>>> b=Family()

>>> print(a.lastname)

김

>>> print(b.lastname)

김

>>> Family.lastname = "박"

>>> print(a.lastname)

박

>>> print(b.lastname)

박

5.2 모듈

모듈 함수, 변수, 클래스를 모아 놓은 파일

모듈만들기 확장자가 .py인 파일 = 모듈

모듈불러오기1 cmd에서 모듈이 저장된 디렉터리로 이동 후, python실행

import 명령어 이미 만들어 놓은 파이썬 모듈을 사용할 수 있게 해주는 명령어

형식 모듈전체를 import하는 경우 import 모듈이름

모듈의 함수만을 import하는 경우 import 모듈이름.함수이름 함수사용시, 모듈이름.함수이름()

from 모듈이름 import 모듈함수 함수사용시, 함수이름()

ex) #mod1.py

def add(a,b):

    return a+b

def sub(a,b):

    return a-b

C:\Users\Lee>cd C:\Users\Lee\OneDrive\공부\전공독서\프로그래밍 언어\점프 투 파이썬(이지스퍼블리싱, 박응용)\첫째마당

C:\Users\Lee\OneDrive\공부\전공독서\프로그래밍 언어\점프 투 파이썬(이지스퍼블리싱, 박응용)\첫째마당>python

Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>> import mod1

>>> print(mod1.add(1,3))

4

if \_\_name\_\_ == “\_\_main\_\_”의 의미 \_\_name\_\_ 모듈의 이름이 저장되는 변수

ex) import된 mod1 🡪 \_\_name\_\_은 mod1

“\_\_main\_\_” 직접 파일이 실행할 경우, \_\_name\_\_에 저장되는 값

ex) #mod1.py

def add(a,b):

    return a+b

def sub(a,b):

    return a-b

#mod1.py를 직접 실행할때는 print()문이 실행됨

#mod1.py를 import할 때는 실행되지 않음

if \_\_name\_\_ =="\_\_main\_\_":

    print(add(1,2))

    print(sub(4,2))

클래스나 변수등을 포함한 모듈 모듈에 함수, 클래스, 변수 포함가능

모듈안에 있는 클래스 사용 형식 인스턴스 = 모듈이름.클래스()

ex) #mod2.py

PI = 3.141592

class Math:

    def solve(self, r):

        return PI\*(r\*\*2)

def add(a,b):

    return a+b

>>> mod2.PI

나 혼자 코딩 1. a=mod2.Math()

print(a.solve(5))

다른 파일에서 모듈 불러오기 파이썬 파일에서 import 명령어 사용 가능

ex) #modtest.py

import mod2

result = mod2.add(3,4)

print(result)

모듈 불러오기2 모듈이 저장된 디렉터리로 이동하지 않고 모듈을 불러와서 사용

1. sys.path.append(모듈이 저장된 디렉터리) 사용 sys모듈 파이썬 설치 시, 함께 설치됨

파이썬 모듈이 위 디렉터리에 들어있다면, 모듈이 저장된 디렉터리로 이동할 필요없이 바로 사용 가능

sys.path의 리턴값 파이썬 라이브러리가 설치된 디렉터리가 요소인 리스트 🡪 append() 메서드 사용 가능

2. PYTHONPATH 환경 변수 사용 PYTHONPATH 환경변수 환경을 설정하는 변수

set PYTHONPATH = 모듈이 저장된 디렉터리

5.3 패키지

.을 이용해 파이썬 모듈을 계층적으로 관리

디렉터리, 파이썬 모듈로 관리 루트디렉터리 – 서브디렉터리

공동작업, 유지보수 유리

패키지 만들기 1. 루트디렉터리 생성

2. 루트디렉터리의 하위 디렉터리인 서브디렉터리 생성

3. 디렉터리마다 \_\_init\_\_.py 파일 생성

4. set 명령어로 PYTHONPATH환경변수에 디렉터리 추가

패키지 안의 함수 실행하기 형식 import 루트디렉터리.서브디렉터리.모듈 사용 루트디렉터리.서브디렉터리.모듈명.함수()

from 루트디렉터리.서브디렉터리 import 모듈 사용 모듈명.함수()

from 루트디렉터리.서브디렉터리.모듈명 import 함수 사용 함수()

\_\_init\_\_.py의 용도 해당 디렉터리가 패키지의 일부임을 알려줌

특정 디렉터리의 모듈을 \*을 사용해 import할 때 형식 from 루트디렉터리.서브디렉터리 import \*

\_\_init\_\_.py에 \_\_all\_\_변수 설정 🡪 import 가능 모듈 정의 \*을 통해 import할 경우, \_\_all\_\_에 정의된 모듈만 import됨

ex) \_\_all\_\_ = [ ‘acho’ ]

나 혼자 코딩 from game.graphic import \*

relative 패키지 패키지 안의 모듈이 패키지 안의 다른 모듈을 쓰고 싶을 때

전체 경로 사용 import ex) game.sound.echo import

.. 부모디렉터리 ex) ..sound.echo import echo\_test

. 현재디렉터리

5.4 예외처리

try, except 사용

오류는 어떨 때 발생하는가? FileNotFoundError, ZeroDivisionError, IndexError …

오류 예외 처리 기법 try, except문 try블록 수행 중 오류가 발생하면 except 블록 수행

형식 try:

…

except [발생오류[as 오류 메시지 변수]]: #[]기호 – 괄호 안의 내용을 생략할 수 있다는 관례 표현기법

…

try, except만 사용 오류 종류에 상관없이 오류가 발생하면 except블록 수행

형식 try:

…

except:

…

발생오류만 포함한 try,except문 오류가 발생했을 때, except문에 지정된 오류 이름이 일치하면 except 블록 수행

형식 try:

…

except:

…

발생오류와 오류메시지변수까지 포함한 except문 오류 발생 시, except문에 지정된 오류 이름이 일치하면 except 블록 수행, 오류메시지의 내용을 변수에 저장

형식 try:

…

except 발생오류 as 오류메시지변수:

…

ex) try:

    4/0

except ZeroDivisionError as e:

    print(e)

try…finally 구문 finally절 try문 수행 중 예외발생여부에 상관없이 항상 finally 구문 수행

파일객체.close() 할 때 자주 사용

여러 개의 오류 처리 try문 안에서 여러 개의 오류 처리

형식 try:

… #발생오류2

… #발생오류1

except 발생오류1:

…

except 발생오류2:

….

발생오류2가 먼저 발생한다면 발생오류1은 발생하지 않음

형식 try:

…

except (발생오류1, 발생오류2) as 오류발생변수:

…

2개 이상의 오류를 동시 처리 할 때, 괄호를 사용해 함께 묶어 처리

나 혼자 코딩 답 a=[1,2,3]

try:

    a[4]

except IndexError as e:

    print(e)

오류 회피하기 특정오류가 발생할 시, 통과시킬 것

형식 try:

…

except 파일오류:

pass

파일오류오류 발생 시, pass 사용해 회피

오류 일부러 발생시키기 raise 명령어 형식 raise 발생오류

예외 만들기 파이썬의 내장 클래스인 Exception 클래스 상속

ex) class 클래스이름(Exception):

…

오류메시지가 보이게 하려면 오류클래스에 \_\_str\_\_메서드 구현 오류메시지를 print()문으로 출력할 때, 호출

5.5 내장함수

파이썬 내장함수 import 필요 X

아무런 설정없이 바로 사용 가능

abs() 어떤 숫자를 입력받았을 때, 그 숫자의 절댓값을 돌려주는 함수

형식 abs(숫자)

리턴 숫자의 절댓값

ex) >>> abs(3)

3

all() 반복가능한 자료형을 입력인수로 받아, 인수의 요소가 모두 참이면 True, 인수의 요소가 하나라도 거짓이면 False를 돌려줌

형식 all(반복가능한자료형)

리턴 불린

ex) >>> all([1,2,3])

True

>>> all([1,2,3,0])

False

any() 반복가능한 자료형을 입력인수로 받아, 인수의 요소가 하나라도 참이라면 True, 인수의 요소가 모두 거짓이면 False를 돌려줌

형식 any(반복가능한자료형)

리턴 불린

ex) >>> any([1,2,3,0])

True

>>> any([0,""])

False

chr() 아스키코드 값을 입력받아, 코드에 해당하는 문자 출력

형식 chr(아스키코드)

리턴 아스키코드에 해당하는 문자

ex) >>> chr(97)

'a'

>>> chr(48)

'0'

dir() 객체가 자체적으로 가지고 있는 변수, 함수

형식 dir(객체)

리턴 객체가 가진 변수, 함수의 이름을 요소로 가진 리스트

ex) >>> dir([1,2,3])

['\_\_add\_\_', '\_\_class\_\_', '\_\_class\_getitem\_\_', '\_\_contains\_\_', '\_\_delattr\_\_', '\_\_delitem\_\_', '\_\_dir\_\_', '\_\_doc\_\_', '\_\_eq\_\_', '\_\_format\_\_', '\_\_ge\_\_', '\_\_getattribute\_\_', '\_\_getitem\_\_', '\_\_gt\_\_', '\_\_hash\_\_', '\_\_iadd\_\_', '\_\_imul\_\_', '\_\_init\_\_', '\_\_init\_subclass\_\_', '\_\_iter\_\_', '\_\_le\_\_', '\_\_len\_\_', '\_\_lt\_\_', '\_\_mul\_\_', '\_\_ne\_\_', '\_\_new\_\_', '\_\_reduce\_\_', '\_\_reduce\_ex\_\_', '\_\_repr\_\_', '\_\_reversed\_\_', '\_\_rmul\_\_', '\_\_setattr\_\_', '\_\_setitem\_\_', '\_\_sizeof\_\_', '\_\_str\_\_', '\_\_subclasshook\_\_', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']

divmod() 2개의 숫자를 입력으로 받아, 몫과 나머지를 튜플 형태로 리턴

형식 divmod(a,b)

리턴 a를 b로 나눈 몫과 나머지가 요소인 튜플

ex) >>> divmod(7,3)

(2, 1)

enumerate() 순서있는 자료형을 입력으로 받아 인덱스 값을 포함하는 enumerate객체를 돌려줌

형식 enumerate(순서있는자료형)

리턴 enumerate객체 – 인덱스값과 매칭된 인수의 요소가 요소인 반복가능한 객체

ex) >>> for i, name in enumerate(['body', 'foo', 'bar']):

print(i,name)

0 body

1 foo

2 bar

eval() 실행가능한 문자열을 입력으로 받아 문자열을 실행한 결괏값을 리턴

형식 eval(실행가능한 문자열)

리턴 실행가능한 문자열을 실행한 결괏값

ex) >>> eval('1+3')

4

입력받은 문자열로 파이썬 함수나 클래스를 동적으로 실행하고 싶을 때 사용

filter() 함수이름, 반복가능한 자료형을 인수로 받아 반복가능한 자료형의 요소를 함수에 입력해 반환값이 참인 것만 묶어 리턴

형식 filter(함수이름, 반복가능한자료형)

리턴 함수의 반환값이 참인 것만 모은 filter객체

ex) >>> def positive(x):

return x>0

>>> print(list(filter(positive, [1,-3,2,0,-5,6])))

[1, 2, 6]

lamda를 사용하면 간편한 코드 작성 가능

hex() 정수 값을 입력받아 16진수로 변환하여 돌려주는 함수

형식 hex(정수)

리턴 입력된 정수가 16진수로 바뀐 문자열

ex) >>> hex(234)

'0xea'

id() 객체를 입력받아 객체의 고유주소값을 돌려주는 함수

형식 id(객체)

리턴 객체의 고유주소 값을 나타내는 int 객체

ex) >>> a=3

>>> id(3)

1949630032240

>>> id(a)

1949630032240

input() 사용자의 입력을 받는 함수

형식 입력이저장되는변수 = input([안내문자열])

리턴 사용자가 입력한 문자열

ex) >>> a=input()

안녕하세요

>>> a

'안녕하세요'

int() 문자열 형태의 숫자, 소수점이 있는 숫자를 정수 형태로 돌려주는 함수

숫자형문자열 or 소수의 진수를 지정해 10진수로 변환가능

형식 int(숫자형문자열or소수, [숫자형문자열or소수의 진수 = 10])

리턴 숫자형문자열or소수를 정수로 바꾼 것

ex) >>> int('3')

3

>>> int('11',2)

3

isinstance() 첫번째 인수로 인스턴스, 두번째 인수 클래스이름을 받아 인스턴스가 클래스의 인스턴스인지 판별 🡪 참, 거짓 반환

형식 isinstance(인스턴스, 클래스)

리턴 인스턴스가 클래스의 인스턴스이면 True, 아니면 False

ex) >>> class Person: pass

>>> a=Person()

>>> isinstance(a, Person)

True

len() 입력값의 길이(요소의 전체 개수)를 돌려주는 함수

형식 len(객체)

리턴 요소의 전체 개수인 int객체

ex) >>> len("python")

6

list() 반복가능한 자료형을 입력받아 리스트로 만들어 돌려주는 함수

형식 list(반복가능한 자료형)

리턴 반복가능한 자료형을 리스트로 만듬

ex) >>> list("pyhton")

['p', 'y', 'h', 't', 'o', 'n']

map() 함수와 반복가능한 자료형을 입력받고, 각 요소를 인수로 받아 함수가 수행한 결과를 묶어 map 객체를 반환하는 함수

형식 map(함수, 반복가능한자료형)

리턴 반복가능한 자료형의 요소가 입력으로 들어간 함수의 출력값을 모은 map 객체

ex) >>> def two\_times(x): return x\*2

>>> list(map(two\_times, [1,2,3,4]))

[2, 4, 6, 8]

max() 인수로 반복가능한 자료형을 입력받아, 최댓값을 돌려주는 함수

형식 max(반복가능한 자료형)

리턴 반복가능한 자료형 중 최댓값인 요소

ex) >>> max("python")

'y'

min() 인수로 반복가능한 자료형을 입력받아, 최솟값을 돌려주는 함수

형식 min(반복가능한 자료형)

리턴 반복가능한 자료형 중 최솟값이 요소

ex) >>> min("python")

'h'

oct() 정수형태의 숫자를 8진수 문자열로 바꿈

형식 oct(정수형태숫자)

리턴 8진수로 바뀐 문자열

ex) >>> oct(34)

'0o42'

open() 파일이름, 열기모드를 입력받아 파일객체를 리턴

형식 파일객체변수 = open(파일이름, [읽기방법]) #읽기방법의 기본값 = r

리턴 파일객체

ex) f=open("binary\_file", "rb")

ord() 문자를 입력으로 받아 문자의 아스키코드값을 리턴

형식 ord(문자)

리턴 문자의 아스키코드 값

ex) >>> ord('a')

97

pow() 제곱한 결과값을 리턴

형식 pow(피제곱수, 지수)

리턴 피제곱수의 지수만큼 제곱한 결괏값

ex) >>> pow(2,4)

16

range() 입력받은 숫자에 해당하는 범위값을 반복가능한 range객체로 만들어 리턴

형식 range([시작숫자=0], 끝숫자, [단위=1])

리턴 시작숫자부터 (끝숫자-1)까지의 단위만큼 건너뛰는 숫자

ex) >>> list(range(1,10,2))

[1, 3, 5, 7, 9]

>>> list(range(1,10))

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

round() 숫자를 입력받아 반올림해주는 함수

형식 round(반올림되는 숫자, [반올림해 표시하고 싶은 소수점의 자릿수])

리턴 반올림된 숫자

ex) >>> round(5.678, 2)

5.68

sorted() 입력값을 정렬한 리스트 반환

리스트.sort 리스트 객체 정렬, 리턴값 X

형식 sorted(반복가능한 객체)

리턴 정렬된 리스트

ex) >>> sorted([3,1,2])

[1, 2, 3]

str() 문자열 형태로 객체 변환

형식 str(객체)

리턴 문자열

ex) >>> str(4)

'4'

sum() 반복가능한 객체를 입력으로 받아 모든 요소의 합 리턴

형식 sum(반복가능한 객체)

리턴 인수의 모든 요소의 합

ex) >>> sum([1,2,3])

6

tuple() 반복가능한 객체를 입력으로 받아 튜플형태로 리턴

형식 tuple(반복가능한 객체)

리턴 튜플로 바뀐 반복가능한 객체

ex) >>> tuple("abc")

('a', 'b', 'c')

type() 객체를 입력받아 자료형이 뭔지 알려주는 함수

형식 type(객체)

리턴 객체의 자료형

ex) >>> type([])

<class 'list'>

zip() 반복가능한 자료형 여러 개를 받아 동일한 개수로 이루어진 리스트가 모인 zip객체로 리턴

형식 zip(반복가능한 자료형, 반복가능한 자료형 …)

리턴 동일한 개수로 묶인 자료형 리턴

ex) >>> list(zip([1,2,3],[4,5,6],[7,8,9]))

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

>>> list(zip("abc","de"))

[('a', 'd'), ('b', 'e')]

5.6 외장함수

sys 파이썬 인터프리터가 제공하는 변수와 함수와 변수와 함수를 직접 제어할 수있게 해주는 모듈

sys.argv 명령 행에서 인수 전달하기

리스트 변수

sys.exit() 대화형 인터프리터를 종료하는 것과 같은 기능

sys.path() 자신이 만든 모듈 불러와 사용

파이썬 모듈이 저장되어있는 위치가 저장된 리스트 변수

ex) >>> import sys

>>> sys.path

['', 'C:\\Users\\Lee\\AppData\\Local\\Programs\\Python\\Python39\\Lib\\idlelib', 'C:\\Users\\Lee\\AppData\\Local\\Programs\\Python\\Python39\\python39.zip', 'C:\\Users\\Lee\\AppData\\Local\\Programs\\Python\\Python39\\DLLs', 'C:\\Users\\Lee\\AppData\\Local\\Programs\\Python\\Python39\\lib', 'C:\\Users\\Lee\\AppData\\Local\\Programs\\Python\\Python39', 'C:\\Users\\Lee\\AppData\\Local\\Programs\\Python\\Python39\\lib\\site-packages']

‘’는 현재 디렉터리

pickle 객체의 형태를 그대로 유지하면서 파일에 저장하고 불러올수 있게하는 모듈

pickle.dump() 파일에 객체를 저장

형식 pickle.dump(객체변수, 파일객체변수)

리턴 없음

ex) >>> import pickle

>>> f=open("text.txt", "wb")

>>> data = {1:'python', 2:'you need'}

>>> pickle.dump(data, f)

pickle.dump() 파일에 저장된 객체 반환

형식 pickle.load(파일객체변수)

리턴 파일에 저장된 객체

ex) >>> f=open("text.txt", 'rb')

>>> data = pickle.load(f)

>>> print(data)

{1: 'python', 2: 'you need'}

OS OS자원을 제어할 수 있게 해주는 모듈

os.environ 내 시스템의 환경변수 값을 보여줌

환경변수 정보가 담긴 딕셔너리 객체

ex) >>> import os

>>> os.environ

environ({'ALLUSERSPROFILE': 'C:\\ProgramData', 'APPDATA': 'C:\\Users\\Lee\\AppData\\Roaming', 'COMMONPROGRAMFILES': 'C:\\Program Files\\Common Files', 'COMMONPROGRAMFILES(X86)': 'C:\\Program Files (x86)\\Common Files', 'COMMONPROGRAMW6432': 'C:\\Program Files\\Common Files', 'COMPUTERNAME': 'DESKTOP-JRSL6O7', 'COMSPEC': 'C:\\Windows\\system32\\cmd.exe', 'DRIVERDATA': 'C:\\Windows\\System32\\Drivers\\DriverData', 'HOME': 'C:\\Users\\Lee', 'HOMEDRIVE': 'C:', 'HOMEPATH': '\\Users\\Lee', 'LOCALAPPDATA': 'C:\\Users\\Lee\\AppData\\Local', 'LOGONSERVER': '\\\\DESKTOP-JRSL6O7', 'NUMBER\_OF\_PROCESSORS': '12', 'ONEDRIVE': 'C:\\Users\\Lee\\OneDrive', 'ONEDRIVECONSUMER': 'C:\\Users\\Lee\\OneDrive', 'OS': 'Windows\_NT', 'PATH': 'C:\\Program Files\\Common Files\\Oracle\\Java\\javapath;C:\\Windows\\system32;C:\\Windows;C:\\Windows\\System32\\Wbem;C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\;C:\\Windows\\System32\\OpenSSH\\;C:\\Program Files (x86)\\NVIDIA Corporation\\PhysX\\Common;C:\\Program Files\\NVIDIA Corporation\\NVIDIA NvDLISR;C:\\Users\\Lee\\AppData\\Local\\Programs\\Python\\Python39\\Scripts\\;C:\\Users\\Lee\\AppData\\Local\\Programs\\Python\\Python39\\;C:\\Users\\Lee\\AppData\\Local\\Microsoft\\WindowsApps;C:\\Users\\Lee\\AppData\\Local\\Programs\\Microsoft VS Code\\bin;C:\\Program Files\\Bandizip\\', 'PATHEXT': '.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC', 'PROCESSOR\_ARCHITECTURE': 'AMD64', 'PROCESSOR\_IDENTIFIER': 'AMD64 Family 25 Model 33 Stepping 0, AuthenticAMD', 'PROCESSOR\_LEVEL': '25', 'PROCESSOR\_REVISION': '2100', 'PROGRAMDATA': 'C:\\ProgramData', 'PROGRAMFILES': 'C:\\Program Files', 'PROGRAMFILES(X86)': 'C:\\Program Files (x86)', 'PROGRAMW6432': 'C:\\Program Files', 'PSMODULEPATH': 'C:\\Program Files\\WindowsPowerShell\\Modules;C:\\Windows\\system32\\WindowsPowerShell\\v1.0\\Modules', 'PUBLIC': 'C:\\Users\\Public', 'SESSIONNAME': 'Console', 'SYSTEMDRIVE': 'C:', 'SYSTEMROOT': 'C:\\Windows', 'TEMP': 'C:\\Users\\Lee\\AppData\\Local\\Temp', 'TMP': 'C:\\Users\\Lee\\AppData\\Local\\Temp', 'USERDOMAIN': 'DESKTOP-JRSL6O7', 'USERDOMAIN\_ROAMINGPROFILE': 'DESKTOP-JRSL6O7', 'USERNAME': 'Lee', 'USERPROFILE': 'C:\\Users\\Lee', 'WINDIR': 'C:\\Windows'})

os.chdir() 딕셔너리 위치 변경

형식 os.chdir(‘바꾸고싶은 디렉터리’)

리턴 없음

ex) >>> os.chdir("c:/")

os.getcwd() 디렉터리 위치 리턴

형식 os.getcwd()

리턴 현재 디렉터리 문자열

ex) >>> os.getcwd()

'c:\\'

os.system() 시스템 자체의 프로그램이나 기타 명령어를 파이썬에서 호출

형식 os.system(‘명령어’)

리턴 없음

ex) >>> os.system("dir")

os.popen() 시스템 명령어를 실행한 결괏값을 읽기모드 형태의 파일객체로 리턴

형식 파일객체변수 = os.popen(“dir”)

ex) >>> f = os.popen("dir")

>>> print(f.read())

C 드라이브의 볼륨에는 이름이 없습니다.

볼륨 일련 번호: E0B8-419E

c:\ 디렉터리

2021-08-09 오후 07:43 <DIR> AMD

2021-10-21 오후 10:26 <DIR> Download

2021-10-21 오후 10:27 <DIR> Nexon

2019-12-07 오후 06:14 <DIR> PerfLogs

…

2021-11-14 오후 11:56 <DIR> Windows

2021-08-09 오후 07:51 <DIR> 소프트웨어

0개 파일 0 바이트

11개 디렉터리 294,186,295,296 바이트 남음

shutil 파일을 복사해주는 파이썬 모듈

형식 shutil.copy(“원본파일이름”,”복사한파일이름”)

리턴 없음

ex) >>> import shutil

>>> shutil.copy("src.txt", "dst.txt")

glob 특정 디렉터리의 있는 모든 파일 이름 알기

glob.glob() 디렉터리 안에 있는 모든 파일의 이름들을 리스트로 리턴

메타문자를 써서 원하는 파일 이름만 읽는 것도 가능

형식 glob.glob(“경로”)

리턴 파일의 이름이 요소인 리스트

ex) >>> glob.glob("C:/Users/Lee/OneDrive/공부/전공독서/프로그래밍 언어/점프 투 파이썬(이지스퍼블리싱, 박응용)/첫째마당/mark\*")

['C:/Users/Lee/OneDrive/공부/전공독서/프로그래밍 언어/점프 투 파이썬(이지스퍼블리싱, 박응용)/첫째마당\\marks1.py', 'C:/Users/Lee/OneDrive/공부/전공독서/프로그래밍 언어/점프 투 파이썬(이지스퍼블리싱, 박응용)/첫째마당\\marks2.py', 'C:/Users/Lee/OneDrive/공부/전공독서/프로그래밍 언어/점프 투 파이썬(이지스퍼블리싱, 박응용)/첫째마당\\marks3.py']

tempfile 파일을 임시로 만들어서 사용

tempfile.mkstemp() 중복되지 않는 임시 파일의 이름은 만들어 튜플 형식으로 리턴

형식 변수 = tempfile.mkstemp()

리턴 중복되지 않는 임시파일의 이름이 저장된 튜플

ex) >>> f=tempfile.mkstemp()

>>> f

(4, 'C:\\Users\\Lee\\AppData\\Local\\Temp\\tmp1jsubq9k')

tempfile.TemporaryFile() 저장공간으로 사용할 파일객체(wb모드) 리턴

형식 파일객체변수 = temp.TemporaryFile()

리턴 wb모드로 열린 파일객체

ex) >>> f=tempfile.TemporaryFile()

>>> f.close()

time time.time() 1970년 1월 1일 0시 0분 0초를 기준으로 지난 초를 실수형태로 리턴

형식 time.time()

리턴 1970년 1월 1일 0시 0분 0초를 기준으로 지난 초

ex) >>> time.time()

1637724941.6913402

time.localtime() 실수값을 년, 월, 일, 시, 분, 초의 형태로 바꿈

형식 time.localtime(실수)

리턴 time.struct\_time 객체

ex) >>> time.localtime(time.time())

time.struct\_time(tm\_year=2021, tm\_mon=11, tm\_mday=24, tm\_hour=12, tm\_min=59, tm\_sec=39, tm\_wday=2, tm\_yday=328, tm\_isdst=0)

time.asctime() time.struct\_time객체를 받아 알아보기 쉬운 문자열로 바꿈

형식 time.asctime(time.struct\_time객체)

리턴 시간을 알아보기 쉽게 만든 문자열

ex) >>> time.asctime(time.localtime(time.time()))

'Wed Nov 24 13:01:45 2021'

time.ctime() = time.asctime(time.localtime(time.time()))

항상 현재 시간 리턴

형식 time.ctime()

리턴 시간을 알아보기 쉽게 만든 문자열

ex) >>> time.ctime()

'Wed Nov 24 13:03:18 2021'

time.strftime() 포맷코드를 이용해 시간을 정밀하게 표현

형식 time.strftime(‘출력할 형식 포맷코드’, time.struct\_time객체)

리턴 형식포맷코드에 따른 문자열

ex) >>> time.strftime("%x", time.localtime(time.time()))

'11/24/21'

time.sleep() 일정한 시간 간격을 두고 루프 시행

형식 time.sleep(실수)

리턴 없음

ex) import time

for i in range(10):

    print(i)

    time.sleep(1)

calendar calendar.calendar() 입력된 연도의 전체 달력 출력

형식 calendar.calendar(년도)

리턴 입력된 연도의 달력이 들어있는 문자열

ex) >>> calendar.calendar(2021)

' 2021\n\n January February March\nMo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su\n 1 2 3 1 2 3 4 5 6 7 1 2 3 4 5 6 7\n 4 5 6 7 8 9 10 8 9 10 11 12

calendar.pral() calendar.calendar(연도)보다 더 알아보기 쉽게 출력

형식 calendar.prcal(연도, 월) 월이 없을시 전체 달력 출력

리턴 없음

ex) >>> type(calendar.prcal(2021))

2021

January February March

Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su

1 2 3 1 2 3 4 5 6 7 1 2 3 4 5 6 7

4 5 6 7 8 9 10 8 9 10 11 12 13 14

…

calendar.weekday() 날짜에 해당하는 요일정보 리턴

월 – 0, 화 – 1, 수 – 2, … 일 – 6

형식 calendar.weekday(년도, 월, 일)

리턴 날짜에 해당하는 정수

ex) >>> calendar.weekday(2021,11,25)

3

calendar.monthrange() 연도의 달의 1일이 무슨 요일인지, 몇일까지 있는지 튜플형태로 반환

형식 calendar.monthrange(2021, 12)

리턴 (달의 1일의 요일, 달의 마지막 일) //튜플

ex) >>> calendar.monthrange(2021,12)

(2, 31)

random random.random() 0.0부터 1.0사이의 난수값 리턴

형식 random.random()

리턴 0.0과 1.0 사이의 난수값

ex) >>> random.random()

0.8593135429868145

random.randint() 입력한 숫자들 사이의 정수 난수값 리턴

형식 random.randint(정수1, 정수2)

리턴 정수1과 정수2 사이의 정수 난수값

ex) >>> random.randint(1,10)

3

random.choice() 입력으로 받은 리스트에서 무작위 요소 하나 리턴

형식 random.choice(리스트객체)

리턴 랜덤 요소 1개

ex) >>> random.choice([1,2,3])

3

random.shuffle() 리스트 요소를 섞음

형식 random.shuffle(리스트객체)

리턴 없음

ex) >>> data=[1,2,3,4,5]

>>> random.shuffle(data)

>>> data

[1, 4, 5, 2, 3]

webbrowser 자신의 시스템에서 사용하는 기본 웹 브라우저를 자동으로 실행하는 모듈

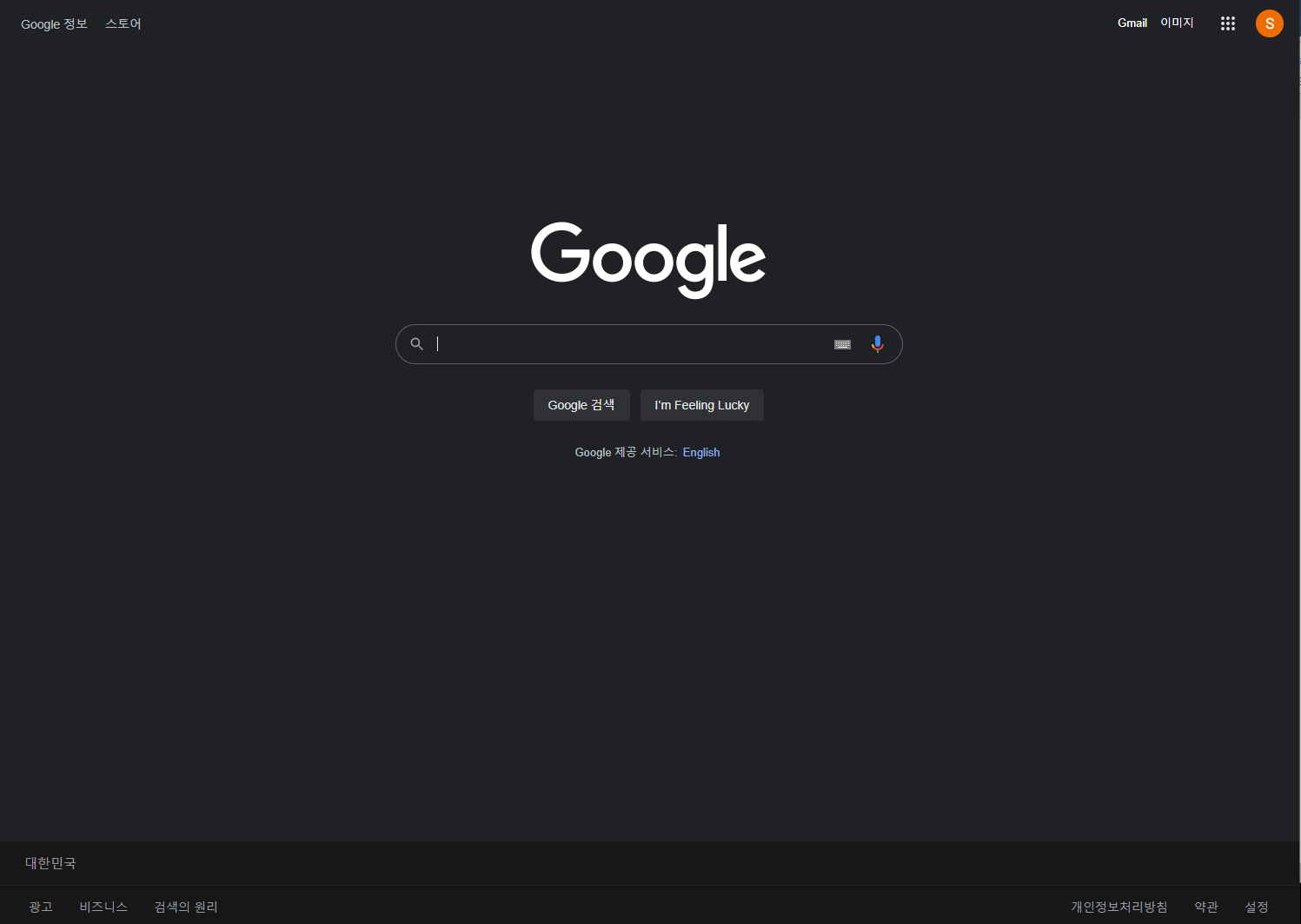
webbrowser.open() 입력주소로 이동

형식 webbrowser.open(“URL”)

리턴 열리면 True, 안 열리면 False

ex) >>> webbrowser.open("https://www.google.co.kr/")

True



webbrowser.open\_new() 새창을 열어 입력주소로 이동

형식 webbrowser.open(“URL”)

리턴 열리면 True, 안 열리면 False

ex) >>> webbrowser.open\_new("https://www.naver.com")

True

threading 스레드 한 프로세스 안에서 2가지 이상의 일을 동시 수행

import time

import threading

def long\_task():

    for i in range(5):

        time.sleep(1)

        print("working: %s\n" %i)

print("start")

threads =[]

for i in range(5):

    t = threading.Thread(target = long\_task)

    threads.append(t)

for t in threads:

    t.start()

for t in threads:

    t.join()

print("End")

둘째마당. 파이썬 기본 문법 익히기

6.1 내가 프로그램을 만들 수 있을까?

구구단 프로그램

def GuGu(n):

    result =[]

    i=1

    while i<10:

        result.append(n\*i)

        i=i+1

    return result

print(GuGu(2))

6.2 3과 5의 배수 합하기

result = 0

for n in range(1,1000):

    if n % 3 == 0 or n % 5 == 0:

        result += n

print(result)

6.3 게시판 만들기

def getTotalPage(m,n):

    if m%n ==0:

        return m//n

    else:

        return m//n+1

print(getTotalPage(5,10))

print(getTotalPage(15,10))

print(getTotalPage(25,10))

print(getTotalPage(30,10))

6.4 간단한 메모장 만들기

import sys

option = sys.argv[1]

if option == '-a':

    memo = sys.argv[2]

    f=open('memo.txt','a')

    f.write(memo)

    f.write('\n')

    f.close()

elif option == '-v':

    f = open('memo.txt')

    memo = f.read()

    f.close()

    print(memo)

6.5 탭을 4개의 공백으로 바꾸기

import os

def search(dirname):

    try:

        filenames = os.listdir(dirname)

        for filename in filenames:

            full\_filename = os.path.join(dirname, filename)

            if os.path.isdir(full\_filename):

                search(full\_filename)

            else:

                ext = os.path.splitext(full\_filename)[-1]

                if ext == '.py':

                    print(full\_filename)

    except PermissionError:

        pass

search("c:/")

7.1 정규표현식 살펴보기

정규표현식 복잡한 문자열을 처리할 때 사용하는 기법

정규표현식은 왜 필요한가? 정규식을 사용해 훨씬 간편하고, 직관적인 코드를 작성할 수 있음

data = """

park 800905-1049118

kim 700908-1069119

"""

result =[]

for line in data.split("\n"):

    word\_result = []

    for word in line.split(" "):

        if len(word) == 14 and word[:6].isdigit() and word[7:].isdigit():

            word = word[:6]+"-"+"\*\*\*\*\*\*\*"

        word\_result.append(word)

    result.append(" ".join(word\_result))

print("\n".join(result))

🡪

import re

data = """

park 800905-1049118

kim 700905-1059119

"""

pat = re.compile("(\d{6})[-]\d{7}")

print(pat.sub("\g<1>-\*\*\*\*\*\*\*", data))

7.2 정규표현식 시작하기

정규표현식의 기초, 메타문자 메타문자 그 문자가 가진 뜻이 아닌 특별한 용도로 사용하는 문자

문자클래스[] 문자클래스로 만들어진 정규식 []사이의 문자들과 매치

ex) [abc] #a,b,c 중 한 개의 문자와 매치

두 문자 사이의 - 문자 두 문자 사이의 범위

ex) [a-c] #a-c

[0-5] #012345

^문자 반대라는 의미

.문자 ‘\n’을 제외한 모든 문자와 매치

ex) [a.b] #”a+\n을 제외한 모든 문자+b”

\*문자 반복이라는 의미(0부터)

ex) [ca\*t] #caat, cat ct와 매칭

+문자 반복(1부터)

ex) [ca\*t] #caat, cat과 매칭, ct와 매칭 안됨

{}문자 반복횟수 고정

{m} m번 반복 고정

ex) [ca{2}t] #cat매칭 안됨, caat과 매칭

{m,n} m에서 n번 반복되면 매치

ex) [ca{2,5}t] # cat매칭 안됨, caat과 매칭, caaaaat과 매칭

?문자 문자의 존재 유무

= {0,1}

ex) [ab?c] #abc매치, ac매치, abbc매치안됨

^문자 문자의 처음

ex) ^python #문자열의 처음은 항상 python

$문자 문자의 끝

ex) python$ #문자열의 끝은 항상 python

파이썬에서 정규 표현식을 지원하는 re모듈 정규표현식을 지원하는 re모듈

import re

re.compile() 정규표현식을 컴파일

형식 re.compile(“정규표현식”)

리턴 패턴객체

정규식을 사용한 문자열 검색 컴파일된 패턴객체의 메서드 패턴객체.match() 문자열의 처음부터 정규식과 매치되는 지 조사

형식 패턴객체.match(“문자열”)

리턴 정규식에 부합하면 match객체 리턴, 부합하지 않으면 None

ex) >>> import re

>>> p=re.compile('[a-z]+')

>>> m=p.match('python')

>>> print(m)

<re.Match object; span=(0, 6), match='python'>

패턴객체.search() 문자열 전체를 검색해 정규식과 매칭되는 지 조사

형식 패턴객체.search(“문자열”)

리턴 정규식에 부합하면 match객체 리턴, 부합하지 않으면 None

ex) >>> import re

>>> p=re.compile('[a-z]+')

>>> m=p.search("3 python")

>>> print(m)

<re.Match object; span=(2, 8), match='python'>

패턴객체.findall() 문자열의 단어를 정규식과 매치해 리스트로 리턴

형식 패턴객체.findall(“문자열”)

리턴 정규식에 부합하는 단어만 모아 리스트로 돌려줌

ex) >>> result = p.findall("life is too short 123")

>>> print(result)

['life', 'is', 'too', 'short']

패턴객체.finditer() 반복가능한 match객체 모음

형식 패턴객체.finditer(“문자열”)

리턴 callable\_iterator 객체

ex) >>> result = p.finditer("life is too short 123")

>>> print(result)

<callable\_iterator object at 0x00000221F48AA970>

>>> for r in result: print(r)

<re.Match object; span=(0, 4), match='life'>

<re.Match object; span=(5, 7), match='is'>

<re.Match object; span=(8, 11), match='too'>

<re.Match object; span=(12, 17), match='short'>

match객체의 메서드 match객체.group() 매치된 문자열을 돌려준다

match객체.start() 매치된 문자열의 시작위치를 돌려준다

match객체.end() 매치된 문자열의 끝 위치를 돌려준다

match객체.span() (시작, 끝)에 해당하는 튜플을 돌려준다

ex) >>> import re

>>> p=re.compile('[a-z]+')

>>> m=p.match('python')

>>> m.group()

'python'

>>> m.start()

0

>>> m.end()

6

>>> m.span()

(0, 6)

컴파일 옵션 DOTALL 약어 S

.문자가 줄바꿈문자를 포함한 모든 문자와 매칭됨

ex) >>> p=re.compile("a.b", re.DOTALL)

>>> m=p.match('a\nb')

>>> print(m)

<re.Match object; span=(0, 3), match='a\nb'>

IGNORECASE 약어 I

대소문자 구별 X

MULTILINE 약어 M

여러 줄과 매치

^ 메타 문자를 각 라인의 처음으로 인식시키고 싶은 경우에 사용

ex) import re

p=re.compile("^python\s\w+", re.MULTILINE)

data = """python one

life is too short

python two

you need python

python three"""

print(p.findall(data))  #python one

VERBOSE 약어 X

verbose 모드를 사용 🡪 정규식을 보기편하기 함, 주석 사용가능

정규식을 주석 또는 줄 단위로 구분

whitespace는 컴파일할 때 제거됨

ex) p=re.compile(r"""

&[#]

(

    0[0-7]+

    | [0-9]+

    | x[0-9a-fA-F]+

)

;

""", re.VERBOSE)

백슬래시 문제 ex) \section 🡪 \s ection 🡪 [\t\n\r\f\v]ection

백슬래시 2개를 사용해 \표시 ex) [\\section](file:///\\section)

정규식 문자열 앞에 r문자를 삽입해 ex) r[\\section](file:///\\section) 🡪 [\\section](file:///\\section)

7.3 강력한 정규표현식의 세계로

메타문자 +, \*, [], {} 매치가 진행될 때, 현재 매치되고 있는 문자열의 위치가 변경됨

= 문자열이 소비됨

| or과 동일한 의미

A|B A 또는 B

ex) >>> p=re.compile('Crow|Servo')

>>> m=p.match('CrowHello')

>>> print(m)

<re.Match object; span=(0, 4), match='Crow'>

^ 문자열의 맨 처음과 일치함

re.MULTILINE을 사용할 경우, 여러줄의 문자열일 때 각 줄의 처음과 일치

ex) >>> print(re.search('^Life', 'My Life'))

None

$ 문자열의 끝과 매치함

ex) >>> print(re.search("short$", "Life is too short, you need python"))

None

\A 문자열의 처음과 매치됨

\A는 줄과 상관없이 전체 문자열의 처음하고만 매치됨

\Z 문자열의 끝과 매치됨

\Z는 전체 문자열의 처음하고만 매치됨

\b 단어구분자

단어 구분 = whitespace

raw string을 알려주는 기호 r을 꼭 붙여야함

ex) >>> p=re.compile(r"\bclass\b")

>>> print(p.search("disclassified"))

None

\B \b와 반대

whitespace로 구분된 단어가 아닌 경우에만 배치

raw string을 알려주는 기호 r을 꼭 붙여야함

ex) >>> p=re.compile(r"\Bclass\B")

>>> print(p.search("one subclass is"))

None

그루핑 문자열이 계속 반복되는 정규식

() 그룹을 만들어주는 메타문자

ex) (ABC)+ #ABCABCABC…

match객체의 group메서드 그루핑된 부분의 문자열을 뽑음

형식 매치객체.group(인덱스)

매치객체.group(0) #매치된 전체 문자열

매치객체.group(n) #n번째 그룹에 해당하는 문자열

리턴 문자열

ex) >>> p=re.compile(r"(\w+)\s(\d+[-]\d+[-]\d+)")

>>> m=p.search("park 101-1234-1234")

>>> m.group(0)

'park 101-1234-1234'

그루핑된 문자열 재참조하기 \n 정규식의 n번째 그룹을 가리킴

ex) >>> p=re.compile(r'(\b\w+)\s\1') #\1 = (\b\w+)

>>> p.search('Paris in the the spring').group()

'the the'

그루핑된 문자열에 이름 붙이기 정규식에 그룹을 만들 때 그룹이름을 지정

(?P<그룹이름>그룹식)

ex) >>> p=re.compile(r"(?P<name>\w+)\s((\d+)[-]\d+[-]\d+)")

>>> m=p.search("park 010-1234-1234")

>>> print(m.group("name"))

park

전방 탐색 긍정 ?=…

…에 해당되는 정규식과 매치되어야 조건 통과

부정 ?!...

…에 해당하는 정규식과 매치되지 않아야 조건 통과

긍정형 전방 탐색

부정형 전방 탐색

문자열 바꾸기 패턴객체의 메서드

패턴객체.sub() 정규식과 매치되는 부분을 다른 문자로 바꿀 수 있음

형식 패턴객체.sub(바꿀문자열, 대상문자열, 바꾸기횟수) #바꾸기횟수는 기본적으로 모두

리턴 바뀐 문자열

ex) >>> p=re.compile('(blue|white|red)')

>>> p.sub('color', 'blue socks and red shoes')

'color socks and color shoes'

ex) >>> p=re.compile('(blue|white|red)')

>>> p.sub('color', 'blue socks and red shoes', 1)

'color socks and red shoes'

sub메서드 참조구문 바꿀 문자열 부분에 \g<그룹이름>, g<참조번호> 사용

sub메서드 함수 사용

Greedy vs Non-Greedy ?문자 가능한한 가장 최소한 의 반복 설정

dsf

(“URL”)