|  |  |  |
| --- | --- | --- |
| **2월 11일 2주차 1번 과제** | **제출일 : 2024.2.18** | **이 선 용** |

Input으로 받은 txt 파일의 내용을 base64로 인코딩하기.

* 사용자가 문자열을 입력해 txt파일을 만들어서 해당 txt파일에 있는 문자열을 읽어서 인코딩, 디코딩을 수행할 수 있게 프로그램을 만들었음.
* Python으로 동일한 기능을 수행하는 프로그램을 만들어라.
* 해당 프로그램에 대한 학습 노트를 500자 이상 작성하라.

!!주의사항!!

* Python을 사용해서 구현해야 함.
* 해당 프로그램을 구현할 때 모듈 사용 가능

import re

import os

import base64

word = {'000000': 'A', '000001': 'B', '000010': 'C', '000011': 'D',

'000100': 'E', '000101': 'F', '000110': 'G', '000111': 'H',

'001000': 'I', '001001': 'J', '001010': 'K', '001011': 'L',

'001100': 'M', '001101': 'N', '001110': 'O', '001111': 'P',

'010000': 'Q', '010001': 'R', '010010': 'S', '010011': 'T',

'010100': 'U', '010101': 'V', '010110': 'W', '010111': 'X',

'011000': 'Y', '011001': 'Z', '011010': 'a', '011011': 'b',

'011100': 'c', '011101': 'd', '011110': 'e', '011111': 'f',

'100000': 'g', '100001': 'h', '100010': 'I', '100011': 'j',

'100100': 'k', '100101': 'l', '100110': 'm', '100111': 'n',

'101000': 'o', '101001': 'p', '101010': 'q', '101011': 'r',

'101100': 's', '101101': 't', '101110': 'u', '101111': 'v',

'110000': 'w', '110001': 'x', '110010': 'y', '110011': 'z',

'110100': '0', '110101': '1', '110110': '2', '110111': '3',

'111000': '4', '111001': '5', '111010': '6', '111011': '7',

'111100': '8', '111101': '9', '111110': '+', '111111': '/'}

def base64\_enc(msg):

string = '' # 인코딩 할 문자열을 저장할 빈 문자열을 생성

zz=[]

result\_enc =''

for i in range(len(msg)):

a = bin(msg[i])

if(len(a)!=9):

a = '0'+a

string += a

else:

string+=a

string = re.sub("b","",string)

length=6

for i in range(0, len(string),length):

zz.append(string[i:i+length])

for i in range(len(zz)):

z = len(zz[i])

if(z == 4):

zd = zz[i];zd +='00';zz[i] =zd;zz.append("=")

if(z == 2):

zd = zz[i];zd +='0000';zz[i] =zd;zz.append("==")

for i in zz:

for key, value in word.items():

if i == str(key):

result\_enc += word[key]

if i == "=":

result\_enc += i

elif i == "==":

result\_enc += i

return result\_enc

def main():

choice = input("1: encoding 2: decoding\n")

filename = input("")

if not os.path.exists(filename):

print("파일이 존재하지 않습니다.")

return

with open(filename, 'rb') as f:

data = f.read()

if choice == '1':

result = base64\_enc(data)

with open(filename, 'w') as f:

f.write(result)

print("done")

elif choice == '2':

try:

result = base64.b64decode(data)

with open(filename, 'wb') as f:

f.write(result)

print("done")

except base64.binascii.Error as e:

print("디코딩할 수 없습니다. 유효한 base65 인코딩된 파일인지 확인하세요.")

else:

print("잘못된 선택입니다.")

main()

인코딩까지는 혼자 구글링하면서 작성하다가 ‘문득 온갖 라이브러리가 다 있는 파이썬에 혹시 base64 관련 라이브러리는 없을까?’ 싶어서 검색한 결과 아니나 다를까 역시나 있었다.

# 정규식(Regular expression) : re

\* 정규 표현식이란 파이썬 내부의 re모듈을 통해 사용할 수 있는, 전문화된 프로그래밍 언어다. 정규 표현식은 문자열 집합에 대한 규칙을 지정하고, 문자열들이 내가 지정한

규칙과 일치하는지 또는, 문자열이 내 패턴과 일치하는 부분이 있는지 확인할 수 있게 해 준다. 추가로, 이 점들을 활용하여 문자열을 수정하거나 내가 원하는 기준으로 분할

까지 할 수 있게 해 준다.

### 정규식은 어떻게 사용할까?

#### 정규식 컴파일 하기

정규식은 패턴 객체로 컴파일 되는데, 패턴 일치를 검색하거나 문자열 취향을 수행하는 등의 다양한 메서드를 가지고 있다.

#### 메타문자

\* $ ( ) \* + . ? [ ] \ ^ { } |

#### 일치 수행하기

\* 정규식을 컴파일한 객체가 있다면, 다양한 메서드를 활용하여 우리가 원하는 작업을 할 수 있다.

\* 정규 표현식에 a는 문자 a와 매칠되지만 (와 (는 매칭되지 않는다. 메타문자인 소괄호를 매칭하고자 하면 백슬래쉬를 앞에 붙여 \(라고 작성해야 문자 (와 매칭이 가능하다.

### re 모듈의 함수

1. match(패턴, 문자열, 플래그)

match()는 문자열의 처음부터 시작해서 작성한 패턴이 일치하는지 확인함.

2. search(패턴, 문자열, 플래그)

search()는 match()와 유사하나, 패턴이 문자열의 처음부터 일치하지 않아도 됨. match()는 항상 시작점이 0이지만, search()는 문자열 전체를 훑기 때문에 시작점이 0이 아닐 수도 있다는 차이점이 있다.

3. findall(패턴, 문자열, 플래그)

findall()은 문자열 안에 패턴에 맞는 케이스를 전부 찾아서 리스트로 반환함.

4. finditer(패턴, 문자열, 플래그)

findall()과 유사하지만 패턴에 맞는 문자열의 리스트가 아닌 iterator 형식으로 반환함.

5. fullmatch(패턴, 문자열, 플래그)

fullmatch()는 문자열에 시작과 끝이 정확하게 패턴과 일치할 때 반환함. match()는 처음부터 패턴에 맞으면 반환을 하지만 해당 함수는 시작과 끝이 정확히 일치 해야한다.

6. split(패턴, 문자열, 최대 split 수, 플래그)

split은 문자열에서 패턴이 맞으면 이를 기점으로 리스트로 쪼개는 함수임. 만약 3번째 인자를 지정하면 문자열을 지정 수 만큼 쪼개고 그 수가 도달하면 쪼개지 않는다.

7. sub(패턴, 교체할 문자열, 문자열, 최대 교체 수, 플래그)

sub는 문자열에 맞는 패턴을 2번째 인자로 교체함. split의 최대 split 수와 동일하게 최대 교체 수를 지정하면 문자열에 맞는 패턴을 교체할 문자열로 교체하고 그 수가 도달하면 더 이상 교체하지 않는다.

8. subn(패턴, 교체할 문자열, 문자열, 최대 교체수, 플래그)

sub()과 동작은 동일하나 반환 결과가 결과(문자열, 매칭횟수) 형태로 반환됨.

9. compile(패턴, 플래그)

만약 패턴과 플래그가 동일한 정규식을 여러번 사용하려면 compile()을 사용하여 지정한 다음, 위의 함수들을 사용할 수 있음.

10. purge()

위 compile()로 만들어 놓은 객체는 캐시에 보통 100개까지 저장된다고 알려져 있으며 그 수를 넘어가면 초기화됨. purge()를 호출하면 100개가 넘어가지 않아도 캐시를 초기화 하는 함수임.

11. escape(패턴)

escape()는 패턴을 입력받으면 특수문자들에 이스케이프 처리를 한 다음 반환함.

12. match object method()

findall()을 제외하고 모든 함수들의 반환은 match object로 반환됨. match object에서는 group(), start(), end() 등과 같이 찾은 패턴이 문자열의 위치나 매칭 문자열을 반환하는 함수를 제공함.

13. 패턴

I, IGNORECASE : 대소문자 구분X

L, LOCATE : \w, \W, \b, \B를 현재의 로케일에 영향을 받음

M, MULTILINE : 여러 줄의 문자열에 대해 패턴을 탐색할 수 있게 함

S, DOTALL : .을 줄바꾸기 문자에도 포함하여 매치하게 함

U, UNICODE : \w, \W, \b, \B가 유니코드 문자 특성에 의존함

X, VERBOSE : 정규식 안의 공백 무시

# base64 Library

\*\*출처: Python documentation\*\*

1) base64.b64encode(s, altchars=None)

Base64를 사용하여 바이트열류 객체 s를 인코딩하고 인코딩된 bytes를 반환합니다.

선택적 altchars는 +와 / 문자의 대체 알파벳을 지정하는 최소 길이 2(추가 문자는 무시됩니다)의 바이트열류 객체여야 합니다. 이를 통해 응용 프로그램은 URL이나 파일 시스템에서 안전한 Base64 문자열을 생성할 수 있습니다. 기본값은 None이며, 표준 Base64 알파벳이 사용됩니다.

2) base64.b64decode(s, altchars=None, validate=False)

Base64로 인코딩된 바이트열류 객체나 ASCII 문자열 s를 디코딩하고 디코딩된 bytes를 반환합니다.

선택적 altchars는 +와 / 문자 대신에 사용되는 대체 알파벳을 지정하는 최소 길이 2(추가 문자는 무시됩니다)의 바이트열류 객체나 ASCII 문자열이어야 합니다.

s가 잘못 채워지면(padded) binascii.Error 예외가 발생합니다.

validate가 False(기본값)면, 일반 base-64 알파벳도 대체 알파벳도 아닌 문자는 채우기(padding) 검사 전에 버려집니다. validate가 True면, 입력에 이 알파벳이 아닌 문자가 있으면 binascii.Error가 발생합니다.

|  |  |  |
| --- | --- | --- |
| **2월 11일 2주차 2번 과제** | **제출일 : 2024.2.18** | **이 선 용** |

로또 추첨 프로그램 만들기

* 사용자가 로또 번호를 생성하는 프로그램을 작성하라.
* 프로그램은 무작위로 1 ~ 45까지의 서로 다른 6개의 숫자를 선택하여 로또 번호를 생성한다.
* 프로그램은 사용자가 원하는 만큼 여러 세트의 로또 번호를 생성할 수 있다.
* 해당 프로그램에 대한 학습 노트를 500자 이상 작성하라.

!!주의사항!!

* 프로그램은 1~45 사이의 서로 다른 6개의 숫자를 무작위로 선택한다.
* 서로 다른 6개의 숫자는 오름차순으로 나열 되어야 한다.
* 마지막 보너스 번호는 앞에 출력될 6개의 숫자와 다른 숫자여야 한다.
* 프로그램은 사용자가 원하는 횟수만큼 로또 번호를 생성할 수 있어야한다.
* 해당 프로그램을 구현할 때 모듈을 사용해도 된다.

import random

# 랜덤번호 생성 함수

def random\_num():

numbers = sorted(random.sample(range(1,46),6))

while True:

bonus = random.randint(1,45)

if bonus not in numbers:

break

numbers.append(bonus)

return numbers

def lotto(): #로또 함수

print("몇개의 로또를 구매 하시겠습니까?")

howmany = int(input()) #몇개의 로또를 구매할지 입력

print(f"{howmany}개의 로또를 구매하셨습니다.")

for \_ in range(howmany):

auto\_answers = random\_num()

print(\*auto\_answers[:6], '+', auto\_answers[6])

lotto()

# Lotto

## unpacking(\* 연산자)

\* Asterisk를 붙여서 배열을 출력할 경우 반복문을 사용하지 않고도 괄호 없이 배열 요소들을 출력할 수 있다.

## random library

random 라이브러리는 난수(랜덤 숫자)를 생성하는 함수들을 제공하는 라이브러리입니다. 다양한 난수 생성 함수와 시퀀스 조작 함수를 포함하고 있습니다.

### 난수 생성 함수:

1. random.random(): 0 이상 1 미만의 실수 난수를 반환합니다.

2. random.uniform(a, b): a 이상 b 이하의 실수 난수를 반환합니다.

3. random.randint(a, b): a 이상 b 이하의 정수 난수를 반환합니다.

4. random.randrange(start, stop, step): start 이상 stop 미만의 정수 난수를 step 간격으로 반환합니다.

5. random.choice(sequence): 시퀀스(리스트, 문자열 등)에서 임의의 요소를 반환합니다.

1. sorted()

\* 수를 순서대로 나열해 놓는다.

2. random.sample(a, b)

\* 랜덤으로 생성된 a를 b의 수 만큼 나열한다.

3. range(a, b)

\* a부터 b-1 범위 만큼

sorted(random.sample(range(1,46),6))

-> 1부터 45까지의 범위 내에서 랜덤으로 6개의 숫자를 뽑아 순서대로 나열한다.

무작위 숫자를 활용하는 로또 프로그램 특성상 python의 random 라이브러리가 있다는 걸 생각해낸 후 random 라이브러리 및 로또 프로그램 작성 시

필요한 함수들이 무엇인지 생각해봤다.

함수를 활용.

1. 랜덤 숫자를 뽑아내고 순서대로 정렬하는 함수 -> random\_num()

2. 그 함수를 사용하여 출력하는 본문 함수 -> lotto()

|  |  |  |
| --- | --- | --- |
| **2월 11일 2주차 3번 과제** | **제출일 : 2024.2.18** | **이 선 용** |

CMD 커맨드 tree 구현

* 사용자가 지정한 시작 디렉토리부터 모든 하위 디렉토리와 파일들을 탐색하는 프로그램을 작성하라.
* 탐색한 디렉토리와 파일들을 트리 형태로 시각적으로 출력해야 한다.
* 각 라인에는 하나의 디렉토리 또는 파일이 표시되며, 들여쓰기나 특수문자를 사용하여 구조적인 관계를 나타낸다.
* 해당 프로그램에 대한 학습 노트를 500자 이상 작성하라.

!!주의사항!!

* 현재 위치해 있는 디렉토리를 기준으로 하위 디렉토리 및 파일들을 계층적으로 구분가능해야 한다.
* 프로그램은 사용자로부터 시작 디렉토리를 입력받을 수 있어야 한다.
* 해당 프로그램을 구현할 때 모듈을 사용해도 된다.

import os

import os.path

import anytree.exporter

import anytree

# 노드 추가 함수

def add\_nodes(parent\_node, directory):

items = os.listdir(directory)

for i in items:

path = os.path.join(directory, i)

child\_node = anytree.Node(i, parent=parent\_node)

if os.path.isdir(path):

add\_nodes(child\_node, path)

def main():

# 사용자 디렉명 입력.

root\_dir = input("루트 디렉토리를 입력하세요: ")

if not os.path.isdir(root\_dir):

print("입력하신 디렉토리가 존재하지 않습니다.")

return

root\_node = anytree.Node(root\_dir)

add\_nodes(root\_node, root\_dir)

for pre, fill, node in anytree.RenderTree(root\_node):

print("%s%s" % (pre, node.name))

main()

add\_nodes()는 주어진 디렉토리의 모든 하위 디렉토리와 파일을 트리에 노드로 추가하는 함수

부모 노드와 디렉토리 경로를 인수로 받게 만들었습니다.

1. Node 클래스

anytree 라이브러리의 Node 클래스는 트리 구조를 만드는데 사용됨. Node 객체는 이름과 부모를 가지며, 부모 노드를 지정하면 자동으로 부모 노드의 자식 노드로 추가됨

ex) child\_node = Node(i, parent=parent\_node)에서 i는 현재 디렉토리나 파일의 이름이고, parent\_node는 상위 디렉을 나타내는 노드임. 이 코드는 i 이름을 가진 새로운 노드를 생성하고,

parent\_node의 자식 노드로 추가함.

2. os.path.join(dir, i)

dir과 i를 연결허여 파일이나 디렉토리의 전체 경로를 생성함.

ex) dir이 '/home/user'이고 i가 'Documents'라면, os.path.join(dir, i)는 '/home/user/Documents'를 반환함.

3. os.path.isdir(path)

path가 dir인지 확인함. 디렉토리라면 True를, 아니라면 False를 반환함.

ex) path = /home/user/documents라면, os.path.isdir(path)는 True를 반환함. 그러나 path = /home/user/file1.txt라면 False를 반환함.

4. RenderTree(root\_node)

루트 노드부터 시작하여 트리를 순회하며 각 노드를 출력함.

각 노드는 이름과 들여쓰기(노드의 깊이에 따라)를 가지며, 이를 통해 디렉토리의 계층 구조를 시각적으로 표현함.

ex) root\_node = /home/user 를 나타내는 노드라면, RenderTree(root\_node)는 /home/user 밑의 모든 디렉토리와 파일을 출력함.

루트 디렉토리는 들여쓰기 없이 출력되고, 하위 디렉토리와 파일은 깊이에 따른 들여쓰기와 함께 출력됨.

5. DotExporter는 트리를 Graphviz의 DOT 언어로 변환하는 클래스.

이 클래스를 이용하면 트리 구조를 DOT 언어로 변환하여 파일로 저장하거나, Graphviz 도구를 이용해 시각적으로 표현할 수 있음.

ex) DotExporter(root\_node).to\_picture("tree.png") 코드는 루트 노드가 root\_node인 트리를 DOT 언어로 변환하고 이를 tree.png 파일에 저장하여 트리를 시각적으로 표현하는 코드임.