MSBA7001 Exercises I

Module 1, 2024-25
HKU Business School

Contents

Regular Expressions

Exercise – revision number

Extract all the revision numbers from "mbox_short.txt". For example:

New Revision: 39756

The revision number in this line is 39756. Store your result in a list called revs. Print out its length and the first 5 values.

```
print(len(revs))
26
```

```
print(revs[:5])
['39771', '39770', '39769', '39766', '39765']
```

Exercise – IP address

Extract all the senders' IP addresses from "mbox_short.txt". For example:

Received: from murder (mail.umich.edu [141.211.14.90])

The IP address in this line is 141.211.14.90. Store your result in a list called ips. Print out its length and the first 5 values.

```
print(len(ips))
189
```

```
print(ips[:5])
['141.211.14.90', '141.211.14.79', '127.0.0.1', '194.35.219.182',
'134.68.220.122']
```

Exercise – email address

Similar to the previous exercise, extract all email address from "mbox_short.txt". Store your result in a list called emails, and print out its length, the first 5 values, and the last 5 values.

Exercise – url

Similar to the previous exercise, extract all urls from "mbox_short.txt". A valid url starts with "http://" or "https://". See two examples below:

- https://collab.sakaiproject.org/portal
- http://source.sakaiproject.org/viewsvn/?view=rev&rev=39766

Note there should be no space in a valid url. Store your result in a list called urls, and print out its length, the first 5 values, and the last 5 values.

Exercise – username

Build a function called name_check(name) that verifies the validity of a username. The rules for a valid username are as follow:

- Must be 6 to 18 letter long

- Consists of any lowercase letter (a-z), number (0-9), an underscore, or a hyphen
- Cannot begin with number, underscore or hyphen

The function should return the following results:

- If the argument is not string, show a warning message. Otherwise, check its validity.
- If the argument is a valid username, return True, otherwise False.

```
name_check(131)
'WARNING: username must be a string'
```

```
name_check('python')
True
```

```
name_check('msba')
False
```

```
name_check('_username')
False
```

```
name_check('666fighting')
False
```

```
name_check('we-are-champion!')
False
```

```
name_check('we-are-champion')
True
```

Exercise – variable name

Define a function called VerifyName(name) that verifies whether a given name is a valid Python variable name and prints out a message. A valid variable name must

- start with a letter or an underscore _ (see examples 1 & 2)
- consist of only letters, numbers, or underscores (see examples 3, 4, & 5)
- not be a reserved keyword (see example 6)

In addition, the name must be a string (see example 7).

```
VerifyName('99MyStr')          # example 1
The name is invalid.
```

```
VerifyName('_')                # example 2
The name is valid.
```

```
VerifyName('My  Str')          # example 3
The name is invalid.
```

```
VerifyName('MyStr!!!')         # example 4
```

```
The name is invalid.
```

```
VerifyName('My_Str5')         # example 5
The name is valid.
```

```
VerifyName('None')            # example 6
The name is a reserved keyword. It cannot be a variable name.
```

```
VerifyName(-999)              # example 7
The argument must be a string.
```

```
VerifyName(1, 2, 3])          # example 8
The argument must be a string.
```

```
VerifyName(None)              # example 9
The argument must be a string.
```

Hint: You may use the keyword module. See the following demonstration:

```
>>> import keyword
>>> print(keyword.kwlist)
['False', 'None', 'True', '__peg_parser__', 'and', 'as', 'assert',
'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif',
'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import',
'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise',
'return', 'try', 'while', 'with', 'yield']
```

```
print(type(keyword.kwlist))
<class 'list'>
```

```
print(keyword.iskeyword('None'))
True
```

Exercise – password generator

Define a function called password_gen(num) that takes an integer number num and produces a valid password of length num. A valid password must

- Be created at random
- Have a length between 8 and 16 (both inclusive)
- Consist of only the following elements:
  1. at least one lowercase letter
  2. at least one uppercase letter
  3. at least one number between 0 and 9 (both inclusive)
  4. at least one special character in _$!? (they are underscore, dollar sign, exclamation mark, and question mark)

Instructions

- There is no additional requirement on the validity of the password.

- If the argument value is not an integer number between 8 and 16, the function should produce a warning message (see examples 1, 3, 6).
- You may define and call other functions as part of password_gen(num).
- *Hint*: The string module may come in handy. Import it and explore its methods.

```
password_gen('8')       # example 1
'WARNING: please enter an integer number between 8 and 16.'
```

```
password_gen(8)         # example 2
'3?zvEEb5'
```

```
password_gen(9.6)       # example 3
'WARNING: please enter an integer number between 8 and 16.'
```

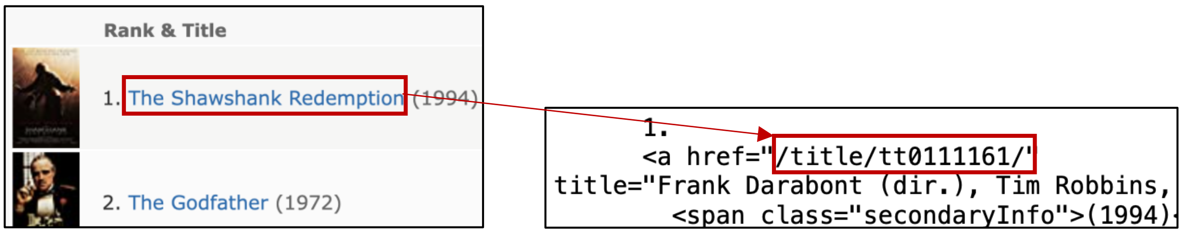
```
password_gen(10)        # example 4
'5uuCRg!U_v'
```

```
password_gen(13)        # example 5
'HEd$YTDpKs4Us'
```

```
password_gen(20)        # example 6
'WARNING: please enter an integer number between 8 and 16.'
```

Exercise – movies links

https://www.imdb.com/chart/top lists the top 250 most highly rated movies. Each movie has a relative path that can be extracted in the source code, illustrated below:



```
        1.
        <a href="/title/tt0111161/"
title="Frank Darabont (dir.), Tim Robbins,
            <span class="secondaryInfo">(1994)
```

The relative path "/title/tt0111161/" can be used to complete the link to the movie's webpage: https://www.imdb.com/title/tt0111161/

The source code of the top chart webpage is stored in the file "imdb.txt". Your job is to extract all 250 unique relative paths from the file and complete the links.

Store your result in a list called links. Print out its length and the first five values.

```
print(len(links))
250
```

```
print(links[:5])
['https://www.imdb.com/title/tt0111161/', 'https://www.imdb.com/titl
e/tt0068646/', 'https://www.imdb.com/title/tt0468569/', 'https://ww
w.imdb.com/title/tt0071562/', 'https://www.imdb.com/title/tt0050083/
']
```

Finally, write your result to a text file named “movie_links.txt”.