CSE 101: Introduction to Computational and Algorithmic Thinking

Stony Brook University

Lab Assignment #13

Spring 2018

Assignment Due: May 4, 2018 by 11:59 pm

Assignment Objectives

This lab will review the basics of regular expressions and string processing.

Getting Started

Visit Piazza and download the "bare bones" file `lab13.py` onto your computer. Open `lab13.py` in PyCharm and fill in the following information at the top:

1. your first and last name as they appear in Blackboard
2. your Net ID (e.g., jsmith)
3. your Stony Brook ID # (e.g., 111999999)
4. the course number (CSE 101)
5. the assignment name and number (Lab #13)

Submit your final `lab13.py` file to Blackboard by the due date and time. Late work will not be graded. Code that crashes and cannot be graded will earn no credit.

Preliminaries

For this lab you will be working with regular expressions in Python. Various functions for working with regular expressions are available in the `re` module. Fortunately, Python makes it pretty easy to see if a string matches a particular pattern.

At the top of the file we must import the `re` module:

```
import re
```

Then we can use the `search()` function to test whether a string matches a pattern. In the example below, the regular expression has been saved in a string called `pattern` for convenience:

```
phone = '123-456-7890'
pattern = r'^\d{3}-\d{3}-\d{4}$'
if re.search(pattern, phone):
    print('The string matches the pattern.')
else:
    print('The string does not match the pattern.')
```

The `r` that precedes the pattern string is not a typo. Rather, the `r` indicates that the string is a "raw" string. In a raw string, as opposed to a "normal" string, any backslash character is interpreted as simply a backslash, as opposed to defining an escape sequence like `\n` or `\t`. Make sure you use raw strings in your Python code when defining regular expressions.

The `^` and `$` at the beginning and end of the regular expression indicate that the entire string must match the regular expression, and not just part of the string. Make sure you include these symbols in your regular expressions too!

Part I: Validate Street Addresses (10 points)

Write a function `street_addresses()`, which takes a single argument `addresses`, which is a list of strings. The function analyzes each string, and, if the string's format matches the pattern of a valid street address as described below, the function appends the string's index in `addresses` into a returned list.

A validly formatted address is defined by the following components, in the given order:

1. one or more digits
2. exactly one space
3. an uppercase letter, followed by any combination of uppercase letters, lowercase letters and spaces
4. exactly one space
5. the word `'Street'`, `'Road'` or `'Path'`, capitalized exactly as given
6. optionally, the following text:
    a. exactly one space
    b. the string `'Apt.'`, capitalized exactly as given
    c. exactly one space
    d. exactly one uppercase letter

Examples of validly formatted addresses are `'58 Gnarled Oak Street'` and `'173 East Main Road Apt. Q'`.

Suppose the following list were passed as `addresses`:

```
['21 Main Street', '12 Main Drive', '5 Elm Apt. E', '6 Elm Path Apt. 5']
```

The function would return `[0, 3]` because only the first and last strings in this example match the pattern defined above.

Examples:

Function Call	Return Value
`street_addresses(['58 Gnarled Oak Street', '12 Chirping Bird Road', '173 East Main Road Apt. Q', '194 Sheep Street Apt. W'])`	`[0, 1, 2, 3]`
`street_addresses(['Main Street', '12 Main Drive', '5 Elm', '6 Elm Apt. 5', '11 Elm Street Apt.', '6 main Street', '19 Dark Horse street'])`	`[]`
`street_addresses(['12 Main Road', '12 Main Drive Apt. G', '100 Circle Road', '5 Elm', '6 Elm Road Apt. G', '12 Elm Street Apt. X', '44 Dark Hollow Road', '6 main Street', '19 Dark Horse street'])`	`[0, 2, 4, 5]`
`street_addresses(['4 High ', '5 GNARLED OAK ROAD', '4 Chirping Bird Path', '57 Soaring Bald Eagle Path', '98 Gnarled Oak Road', '29 Street Apt. b', '28 SHEEP PATH APT. Y', '31 Gnarled Oak Road Apt. B', '52 GNARLED OAK PATH APT. S'])`	`[2, 3, 4, 7]`
`street_addresses(['56 SHEEP PATH APT. P', '67 STREET', 'East Main Path Apt. h', '11 Sheep Street Apt. y', '83 GNARLED OAK ROAD', 'Road', '10 High Path Apt. f', '89 Street Apt. f', '55 east main street'])`	`[]`
`street_addresses(['34 Soaring Bald Eagle Street', '98 Dark Elm Street Apt. m', '40 East Main Apt. Y', '96 East Main ', '51 Road', '54 Gnarled Oak Path Apt. o', '15 Apple Path Apt. h', '56 Apple Path', '66 Apple Path', 'Dark Elm Street Apt. a', '82 Street Apt. A', '42 Soaring Bald Eagle Road'])`	`[0, 7, 8, 11]`
`street_addresses(['97 apple street', 'Chirping Bird Street Apt. g', '26 Path Apt. n', '92 East Main Street', '73 Gnarled Oak Street', '42 DARK ELM PATH APT. H', '14 Chirping Bird Road Apt. Z', '59 Dark Elm '])`	`[3, 4, 6]`
`street_addresses(['IUhwiV', 'FCaXcjUg', 'JEwrdGDrZocSKIn', 'QTkTNbin', 'TdiuIQev'])`	`[]`

Part II: Determine a Package's Destination (10 points)

Write a function `package_destination()`, which takes two arguments in the following order:

1. `package_code`: a string that *might* represent a valid shipping code on a package
2. `country_codes`: a dictionary that matches a country's code (a string) to the name of a country

A validly formatted address is defined by the following components, in the given order:

1. three digits

2. a dash
3. three digits
4. a dash
5. one uppercase letter
6. one digit

If the string is validly formatted, the function extracts the three-digit country code at the start of the string and uses it as a key in the `country_codes` dictionary. The country name associated with the key is returned by the function. If the `package_code` is invalidly formatted, or it is validly formatted but the extracted country code is not in the dictionary, the function returns the string `'error'`.

Hint: after verifying that the argument string matches the expected pattern, use the `split()` method to extract the part of the string needed for the dictionary lookup.

Examples:

Function Call	**Return Value**
`package_destination('898-524-G0', {'898': 'Kuwait', '312': 'Russia', '709': 'Bangladesh', '400': 'Slovenia', '905': 'Yemen', '256': 'Slovakia', '085': 'Burundi'})`	`'Kuwait'`
`package_destination('787-195-F5', {'167': 'Peru', '873': 'Brazil', '539': 'Greece', '787': 'Mongolia'})`	`'Mongolia'`
`package_destination('328-168-I4', {'328': 'Equatorial Guinea', '378': 'Malawi', '319': 'Hungary', '612': 'Liberia', '839': 'Dominican Republic', '565': 'Mongolia', '036': 'Zimbabwe', '620': 'Luxembourg', '766': 'Panama', '562': 'India', '976': 'New Zealand', '904': 'Egypt', '603': 'Nepal', '013': 'Lebanon', '863': 'France'})`	`'Equatorial Guinea'`
`package_destination('911-361-N2', {'991': 'Italy', '130': 'Jamaica', '380': 'Greece', '626': 'Qatar', '788': 'Slovenia', '126': 'Oman', '504': 'Zambia', '430': 'Rwanda', '911': 'Honduras', '355': 'Colombia', '796': 'Lithuania'})`	`'Honduras'`
`package_destination('861-135-H6', {'160': 'Equatorial Guinea', '259': 'Faroe Islands', '328': 'Democratic Republic of the Congo', '861': 'South Sudan'})`	`'South Sudan'`

`package_destination('162-855-N9', {'921': 'Peru', '696': 'Uzbekistan', '571': 'Kyrgyzstan', '273': 'Chile', '164': 'Ghana', '349': 'Nigeria', '478': 'Bulgaria', '603': 'Paraguay', '642': 'Bahrain', '301': 'Turkmenistan', '863': 'Czech Republic', '126': 'Burkina Faso', '155': 'Iraq', '554': 'Laos', '907': 'Israel'})`	`'error'`
`package_destination('11-456-W5', {'331': 'Libya', '664': 'Fiji', '229': 'Cape Verde', '002': 'Albania', '449': 'Hungary', '278': 'Turkey', '328': 'Niger', '633': 'Togo', '558': 'Lesotho', '861': 'Jordan', '251': 'Spain', '403': 'Algeria', '809': 'Chad'})`	`'error'`
`package_destination('752-9194-M0', {'752': 'Sri Lanka', '815': 'Fiji', '141': 'Bulgaria', '992': 'Jamaica', '915': 'Lebanon', '903': 'Mali', '671': 'Central African Republic', '929': 'Uzbekistan', '893': 'Costa Rica', '197': 'Equatorial Guinea', '443': 'Brazil', '382': 'Burkina Faso', '650': 'Hong Kong'})`	`'error'`
`package_destination('595-37-V1', {'595': 'Kyrgyzstan', '516': 'Spain', '310': 'Singapore', '391': 'Zimbabwe', '211': 'Austria', '081': 'Uruguay', '215': 'Sri Lanka', '640': 'Qatar', '218': 'Italy', '817': 'Georgia', '356': 'Sri Lanka'})`	`'error'`
`package_destination('102-052-0', {'569': 'Fiji', '865': 'Denmark', '199': 'Bhutan', '951': 'Tanzania', '460': 'Rwanda', '046': 'Kenya', '340': 'Guyana', '922': 'Morocco', '416': 'Niger', '102': 'Andorra', '161': 'Lithuania', '921': 'Turkmenistan', '066': 'Togo', '593': 'Equatorial Guinea', '436': 'Egypt'})`	`'error'`
`package_destination('382-525-N', {'151': 'Bhutan', '107': 'Jamaica', '991': 'Poland', '089': 'Japan', '382': 'Turkey', '427': 'Iceland', '276': 'New Zealand', '859': 'South Sudan', '613': 'Guyana', '170': 'Papua New Guinea'})`	`'error'`

Note: The quotation marks displayed in the example return values are there to emphasize that the return values are strings. You should not add quotation marks to your return values.

Part III: Create a Business Card (20 points)

Write a function `business_card()`, which takes a single argument `contact_info` that contains the contact information for a person. After validating the format of the argument, the function creates a new string that reformats the input in a particular way (described later).

A validly formatted `contact_info` string is defined by the following components, in the given order:

1. the string `'*N'`
2. exactly one space
3. a person's name, defined as a single uppercase letter, followed by any combination of uppercase letters, lowercase letters and spaces
4. exactly one space
5. the string `'*P'`
6. exactly one space
7. exactly 10 digits
8. exactly one space
9. the string `'*T'`
10. exactly one space
11. a person's job title, defined as a single uppercase letter, followed by any combination of uppercase letters, lowercase letters and spaces

If the `contact_info` argument is not validly formatted, the function returns the string `'error'`. Otherwise, it returns a new string that contains the contact information reformatted as illustrated in the example below.

Suppose that the argument is the following validly formatted string:

```
'*N Milana Howser Ross *P 1006344742 *T Systems Administrator'
```

The return valid would be this:

```
'Name: Milana Howser Ross, Title: Systems Administrator, Phone: (100) 634-4742'
```

Note that the returned string has the following format:

1. the string `'Name:'`
2. exactly one space
3. the person's name
4. a comma
5. exactly one space
6. the string `'Title:'`
7. exactly one space
8. the person's job title
9. a comma
10. exactly one space
11. the string `'Phone:'`

12. exactly one space
13. the first three digits of the person's phone number, enclosed by parentheses
14. exactly one space
15. the next three digits of the person's phone number
16. a dash
17. the last four digits of the person's phone number

Examples:

Function Call:
`business_card('*N Kaitlin Mcneill *P 1006344742 *T ')`
Return Value:
`'error'`

Function Call:
`business_card('*N Dennis Mata P 3490338670 *T Claims Adjuster')`
Return Value:
`'error'`

Function Call:
`business_card('*N Milana Howser Ross *P 1006344742 *T Systems Administrator')`
Return Value:
`'Name:  Milana Howser Ross, Title:  Systems Administrator, Phone:  (100) 634-4742'`

Function Call:
`business_card('*N *P 5091466032 *T Investor Relations Officer')`
Return Value:
`'error'`

Function Call:
`business_card('*N Junior Senior Pickle Spears *P 0663929680 * Hedge Fund Trader')`
Return Value:
`'error'`

Function Call:
`business_card('*N Milana Howser Ross *P 1800670728 *T Marketing Manager')`
Return Value:
`'Name:  Milana Howser Ross, Title:  Marketing Manager, Phone:  (180) 067-0728'`

Function Call:
`business_card('*N Junior Senior Pickle Spears * 1240284278 *T Investigator')`
Return Value:
`'error'`

Function Call:
`business_card('*N Eugene Gillespie *P 1800670728 *T Employee Specialist')`
Return Value:
`'Name:  Eugene Gillespie, Title:  Employee Specialist, Phone:  (180) 067-0728'`

Function Call:
`business_card('*N Anisa Haines P 9644700679 *TWeb Developer')`
Return Value:
`'error'`

Function Call:
```
business_card('*N Shaurya Nguyen *P 3490338670 *T Insurance Appraiser')
```
Return Value:
```
'Name:  Shaurya Nguyen, Title:  Insurance Appraiser, Phone:  (349) 033-8670'
```

Note: The quotation marks displayed in the example return values are there to emphasize that the return values are strings. You should not add quotation marks to your return values.

How to Submit Your Work for Grading

To submit your `.py` file for grading:

1. Login to Blackboard and locate the course account for CSE 101.
2. Click on "Assignments" in the left-hand menu and find the link for this assignment.
3. Click on the link for this assignment.
4. Click the "Browse My Computer" button and locate the `.py` file you wish to submit. Submit only that one `.py` file.
5. Click the "Submit" button to submit your work for grading.

Oops, I messed up and I need to resubmit a file!

No worries! Just follow the above directions again. We will grade only your last submission.