Exercise Set 1

To-Do Date: Oct 27 at 11:59pm

Submissions

Submit your solutions to **CodePost** (https://codepost.io). Some notes on the differences with this assignment:

- All exercises will be submitted to the same CodePost assignment "Exercise Set 0"
- You can still submit incrementally (one exercise at a time), but tests for files which have not been submitted will fail.
- Descriptions for each exercise is only available here in Canvas.

ex_1_0.py

Write a Python module named `ex_1_0.py` that implements the following two functions. In this exercise, you will use Python selection structures `if-else` statements in conjunction with conditional expressions to implement solutions.

Note: your module should not print anything to the console.

1. Write a function `in_range` that takes three positional arguments: `value`, `vmin`, `vmax`, (in that order). Your function should return `True` if

   `vmin <= value <= vmax`.

   Your function should have the following signature.

   ```
   def in_range(value, vmin, vmax)
   ```

   Note that in Python, this function will work with a wide variety of data types and objects.

2. Write a function named `is_even` that takes in a single positional argument `n` and returns `True` if `n` is **even** or `False` if `n` is **odd**.

   For now, it is safe to assume that only integer types will be passed to this function.

ex_1_1.py

Write a Python module named `ex_1_1.py` that implements a function named `get_username` that accepts a single positional `str` argument `email` (with form `username@domain`). Your function should extract and return (not print) the username portion of the input string–in lowercase.

Note: Your function should handle *invalid* input strings by returning `None`. Recall that `None` is a Python type (equivalent to null in other languages).For now you can assume input strings will never have more than one `@` character in them.

Hint: use the `str.split()` *(https://docs.python.org/3/library/stdtypes.html#str.split) method for this one.*

Here's an example run at the Python interactive prompt:

```
>>> user = get_username('addison@example.com')
>>> type(user)
<class 'str'>
>>> print(user)
addison
>>> user_2 = get_username('invalid-email.com')
>>> type(user_2)
<class 'NoneType'>
>>> print(user_2)
None
>>> user_3 = get_username('JOHN@EXAMPLE.COM')
>>> print(user_3)
john
```

ex_1_2.py

This exercise uses the function that you implemented in `ex_1_1.py`. Add `get_username` to this module and implement a new function called `last_user_login` that takes in a list of the following form and returns a `tuple` `username, last_login` (in that order).

```
account = ['Kaspar', 'Braidon', 'kbraidon3@gnu.org', '10/9/2021']
```

Where the fields are arranged in this order:

```
<first_name>, <last_name>, <email>, <last_login>
```

To accomplish this, your function should do the following (note that the variable names here are simply suggestions):

1. Extract the username from the `email` field in the input list and save it to a variable `username`
2. Collect the last login date and save it to a variable `last_login`.
3. Return `username, last_login` (in that order)

Here's an example test of the function at the interactive prompt:

```
>>> account = ['Kaspar', 'Braidon', 'kbraidon3@gnu.org', '10/9/2021']
>>> user, last_login = last_user_login(account)
>>> user
'kbraidon'
>>> last_login
'10/9/2021'
```

ex_1_3.py

When we have related information in a list as in `ex_1_2.py` it is often best to store the data in a dictionary. This makes lookups more straightforward. In this exercise, you will implement a function `last_logins` that takes in a nested list like the one below and returns a dictionary. This function and any other required code, should be saved in a Python module named `ex_1_3.py`.

Note that each element of the list has the same structure as the list in `ex_1_2.py`.

```
accounts = [
    ['Mitchell', 'Mustarde', 'mmustarde0@boston.com', '10/1/2021'],
    ['Federico', 'Woehler', 'fwoehler1@ftc.gov', '10/21/2021'],
]
```

For an input list of the form above, your function should:

1. Create an empty dictionary using the `dict()` builtin.

2. Loop over all items in the list and:
    a. Get the username and last login date using your `last_login` function.
    b. Save the last login date to the dictionary using the username as the key.
3. Return the resulting dictionary

Here's an example test of the function for the sample `accounts` list above:

```
>>> account_dict = last_logins(accounts)
>>> type(account_dict)
<class 'dict'>
>>> account_dict
{'mmustarde0': '10/1/2021', 'fwoehler1': '10/21/2021'}
```

Note that the key, value pairs here are username, last_login.

ex_1_4.py

This last exercise in the set is very straightforward. You will implement a function named `word_count` that takes in an input string and returns the integer count of words within the string. Save your `word_count` function to a Python module named `ex_1_4.py`.

Tip: The function body can be implemented in one line of Python code using the appropriate string method.

Below is an example test of this function from the interactive prompt:

```
>>> text = "There is a way out of every box, a solution to every
puzzle; it's just a matter of finding it."
>>> num_words = word_count(text)
>>> num_words
20
```