Assignment 1

OPS435 Assignment 1

2020 Fall Semester

Raymond Chan

Due Date

Oct 23, 2020

Before End of Day

Post your progress to github.com

Upload your algorithm, python script, and test results to Blackboard by the due date

Problem Statement

Take **a date of birth** as a string in one of the following format at the command line:

- `YYYYMMDD`
- `YYYY-MM-DD`
- `YYYY/MM-DD`
- `YYYY.MM.DD`

And convert the given date of birth to the following format and send to the standard output, e.g if 20200301 is given, the output should be:

- Mar 1, 2020

Script Development Cycle

- Design an algorithm (step-by-step instruction) which solve a given computation problem
- Convert the algorithm into a scripting language (one task at a time if the language support function)
- Formula test cases and execute each test and document the results
- Document the scripts and functions (in Python, you should use the built-in docstring to make it easy for other to access and use your codes, examples to follow)
- Release and maintenance

Rephrase the Computation problem of Assignment 1

- We are provided with one single data item:
  - A given date of birth as s a string in one of the three formats:
    - YYYYMMDD (e.g 20200301)
    - YYYY/MM/DD (e.g 2020/03/01), or
    - YYYY-MM-DD (e.g. 2020-03-01)
- We are asked to
  - Convert the given date of birth to the following format:
    - mmm d, yyyy (e.g. Mar 1, 2020) where mmm is one of Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, or Dec.

Algorithm for Assignment 1 First try

- Take the first data item (either YYYYMMDD, YYYY/MM/DD, YYYY-MM-DD, or YYYY.MM.DD), remove all the non-digits and extract the year, month, and day from the input data.
- Convert the 2 digit month to a three letter name for the given month
- Rearrange the year, month, and day into the required format → mmm d, yyyy

Convert the algorithm to Python Codes

```
#! /usr/bin/env python3
import sys
dob = sys.argv[1].replace('/','')
month_name = [‘Jan’,’Feb’,’Mar’,’Apr’,’May’,’Jun’,
              ‘Jul’,’Aug’,’Sep’,’Oct’,’Nov’,’Dec’]
year = dob[0:4]
month = int(dob[4:6])
day = dob[6:]
new_dob= month_name[month-1]+’ ‘+day+’, ‘+year
print(“Your date of birth is:”, new_dob)
```

Test and test results

```
>python3 a1_rchan.py 2019/01/20 1
2019/01/21
>python3 a1_rchan.py 2019/01/20 2
2019/01/22
>python3 a1_rchan.py 2019/01/20 15
2019/01/35
```

Test and test results

```
>./dob.py 20200301
Your date of birth is: Mar 01, 2020
>./dob.py 2020/03/01
Your date of birth is: Mar 01, 2020
…
```

More tests

```
>./dob.py 20200230

????

>./dob.py 20201301

????

>./dob.py 20190229

????
```

More tests

```
>./dob.py 2020/10/1
????
>python3 2020-01-01
????
>python3 2020/03/301
????
```

Errors?

Test results and conclusion:

- Syntax error? – No
- Runtime error? – Yes!
- Logical error? – Yes!

Debug and Update

- The original algorithm works only if the user provides valid data
- Runtime errors should be caught and be fixed.
- Logical error should be identified and be fixed
- The original algorithm does not pay attention to the maximum number of days each month has, and the maximum number of days of February depends even on the year of the given date.
- The algorithm needs to be refined.

Lesson learned – input data must be validated FIRST and FOREMOST

To reduce the complexity of input data validation, we should identify different type of data requirements and perform each type of data checking. We should handle one type of checking at a time and create a function for each checking task:

- (a) maximum and minimum check
- (b) format check
- (c) value check

Checking functions

Possible data validation functions:

- size_check(): number of input characters
- leapyear(): given year is a leap year or not
- value_check(): the given day is greater than the maximum number of days for a given month

Documentation

Doctstring – ''' documentation text '''

- Script level
- Function level

Examples on how to do it in class

Doctstring – single line

```
#!/usr/bin/env python3
'''put your docstring here for your script '''

def func1():
    '''put your docstring for func1 here'''
    ...
    return

Def func2():
    '''put your docstring for func2 here'''
    ...
    Return

if __name__ == '__main__':
    ...
    Main block of code
```

Docstring – multi-lines

```
#!/usr/bin/env python3
'''
put your multi-lines docstring for your script
Here, as many lines as you need. Tell the user
mainly what your script can do, and how to use
it.
'''
Def func1():
    ''' multi-lines docstring for func1 here
        As many lines as you need.
    '''
```

Releasing your code

Add codes to your script to enable other users to reuse the functions that you have created for this assignment by importing your script.

The use of the

“**`if __name__ == '__main__':`**”

block

Questions / Answers