CNT 4603: System Administration
Spring 2015

Project Eight – Python Scripting

Instructor : Dr. Mark Llewellyn

markl@cs.ucf.edu
HEC 236, 407-823-2790
Office Hours: M&W 1:00-3:00pm, T&Th 10:30am-12:30pm

Department of Electrical Engineering and Computer Science
Computer Science Division
University of Central Florida

Project Eight

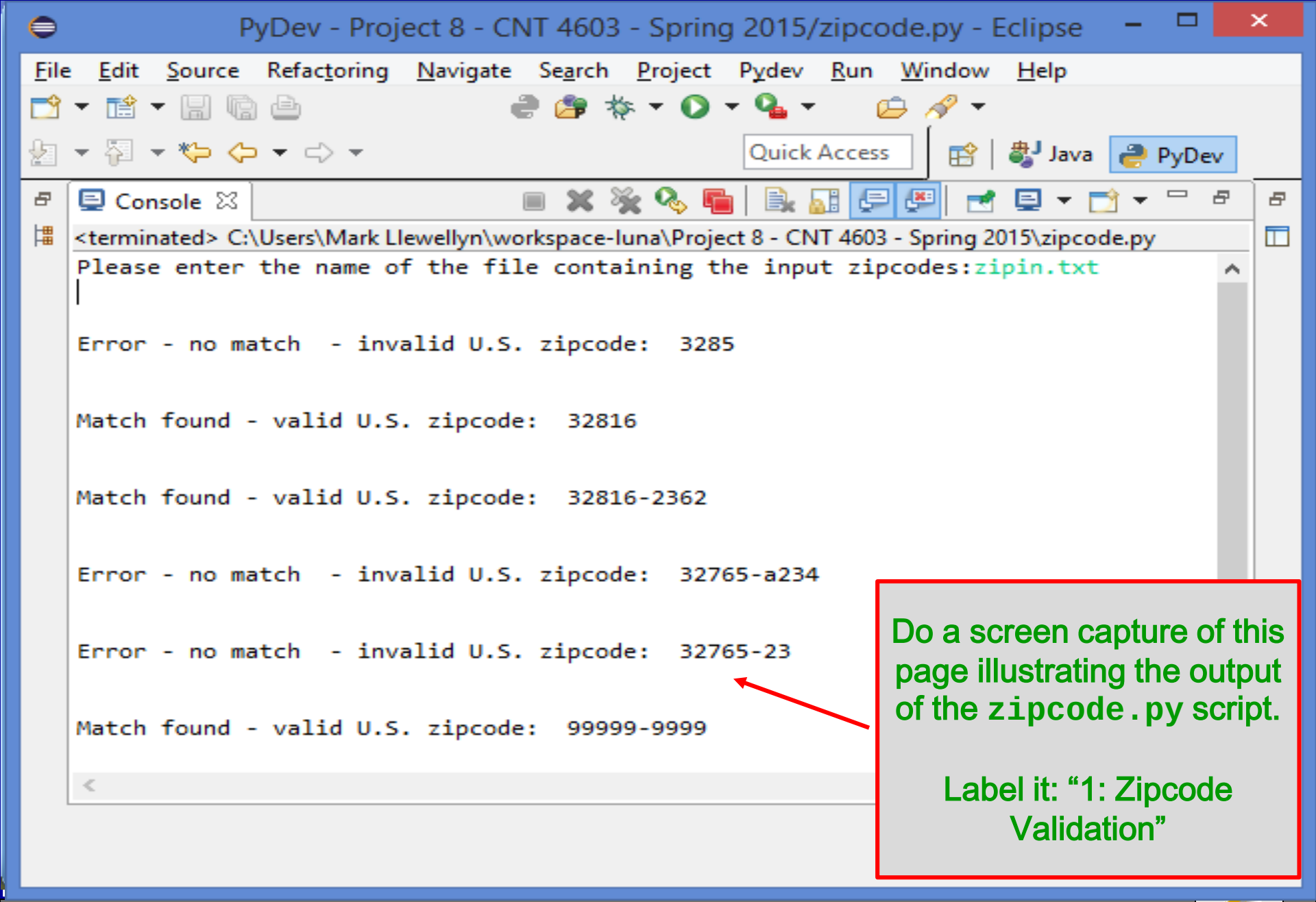
- **Title:** "Project Eight: Python Scripting"
- **Points:** 40 points (10 points per script and output screen shot)
- **Due Date:** Monday April 27th by 11:59 pm WebCourses time.
- **Objectives:** This project focuses on scripting using the Python language. You will write four scripts that will use regular expressions as implemented in Python to recognize/validate certain patterns in input strings.
- **Deliverables:**
  - 1. Screen shots as shown on pages 5, 7, 10, and 13.
  - 2. Your Python script code files (all 4 of them).

Project Eight – Details

- In this project you will create four different, small Python scripts that will each read strings from an input file of unknown size.
- The strings will be operated upon using regular expressions as implemented in Python to check for certain patterns in the input strings. Scripts such as these would be useful for a system administrator performing various searches through Active Directory user account listings or validating server IP addresses.
- You should properly document your scripts following the same conventions we discussed when looking at scripting in PowerShell. You do not need to digitally sign these scripts. You can use any of the Python development environments that we detailed in the notes (Python interpreter, IDLE, PyScripter, or PyDev in Eclipse).
- Each of the scripts that you need to create are detailed on the following pages. Please note the names for the input files used for each script. Points will be deducted if you do not use the input file names as specified! Your script should prompt the user to enter the name of the input file.

Project Eight – Script One – `zipcode.py`

- This script will validate U.S. zipcodes using either the five or nine digit format.
- Example: 32816 is valid, 32816-2362 is valid, 3281 is invalid, 32186*2345 is invalid.
- Source file name: `zipcode.py`.
- Input file name: `zipin.txt`.
- Submit your script source file `zipcode.py` and the screen shot shown on page 5.

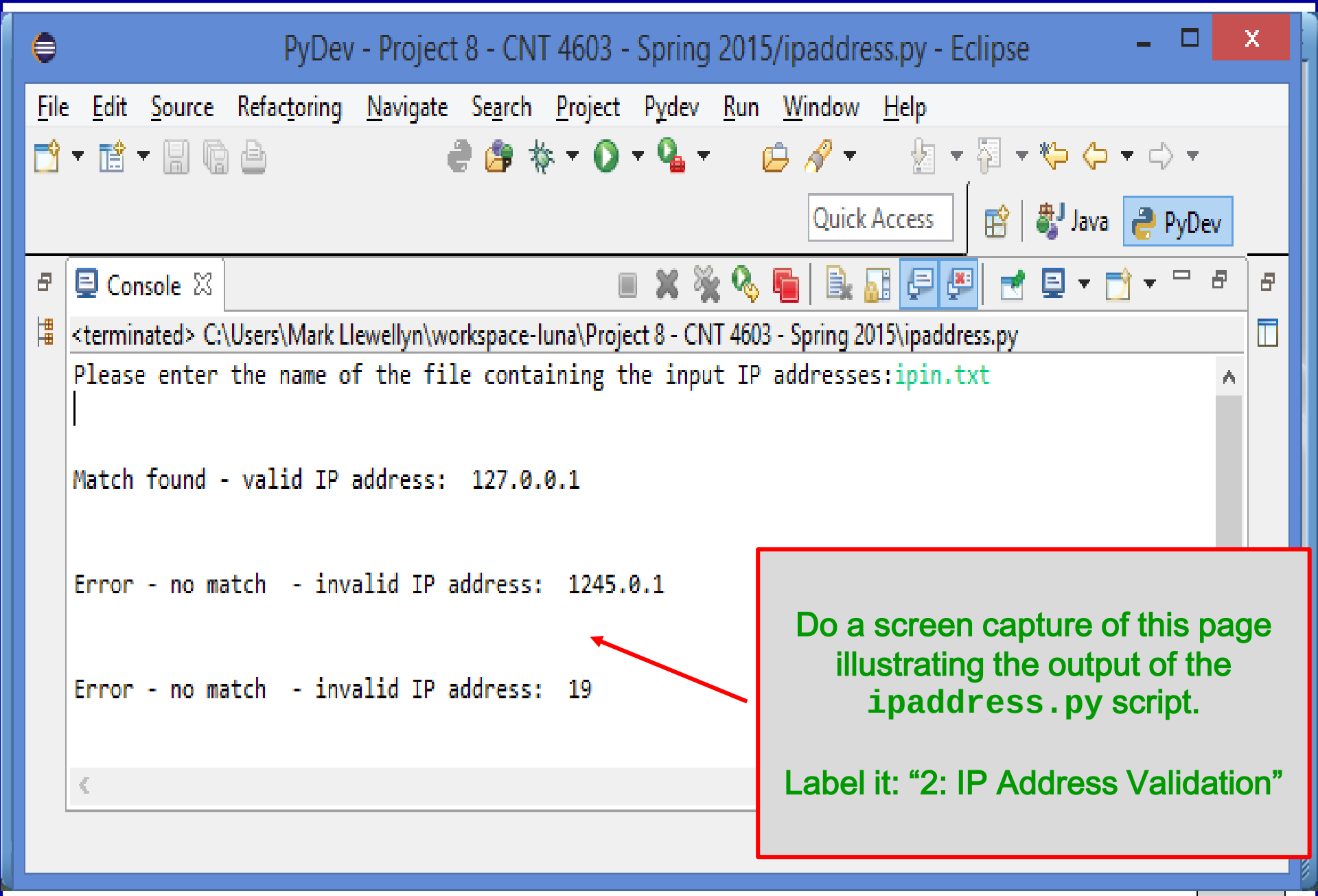


Do a screen capture of this page illustrating the output of the `zipcode.py` script.

Label it: "1: Zipcode Validation"

Project Eight – Script Two – `ipaddress.py`

- This script will validate IP addresses.
- IP addresses are made up of four bytes (each with a valid range of 0-255). They are typically formatted as four sets of digits (each being 1 to 3 digits in length) separated by "." characters.
- Example: 127.0.0.1 is valid, 1245,0.1 is invalid.
- Source file name: `ipaddress.py`.
- Input file name: `ipin.txt`.
- Submit your script source file `ipaddress.py` and the screen shot shown on page 7.



Do a screen capture of this page illustrating the output of the `ipaddress.py` script.

Label it: "2: IP Address Validation"

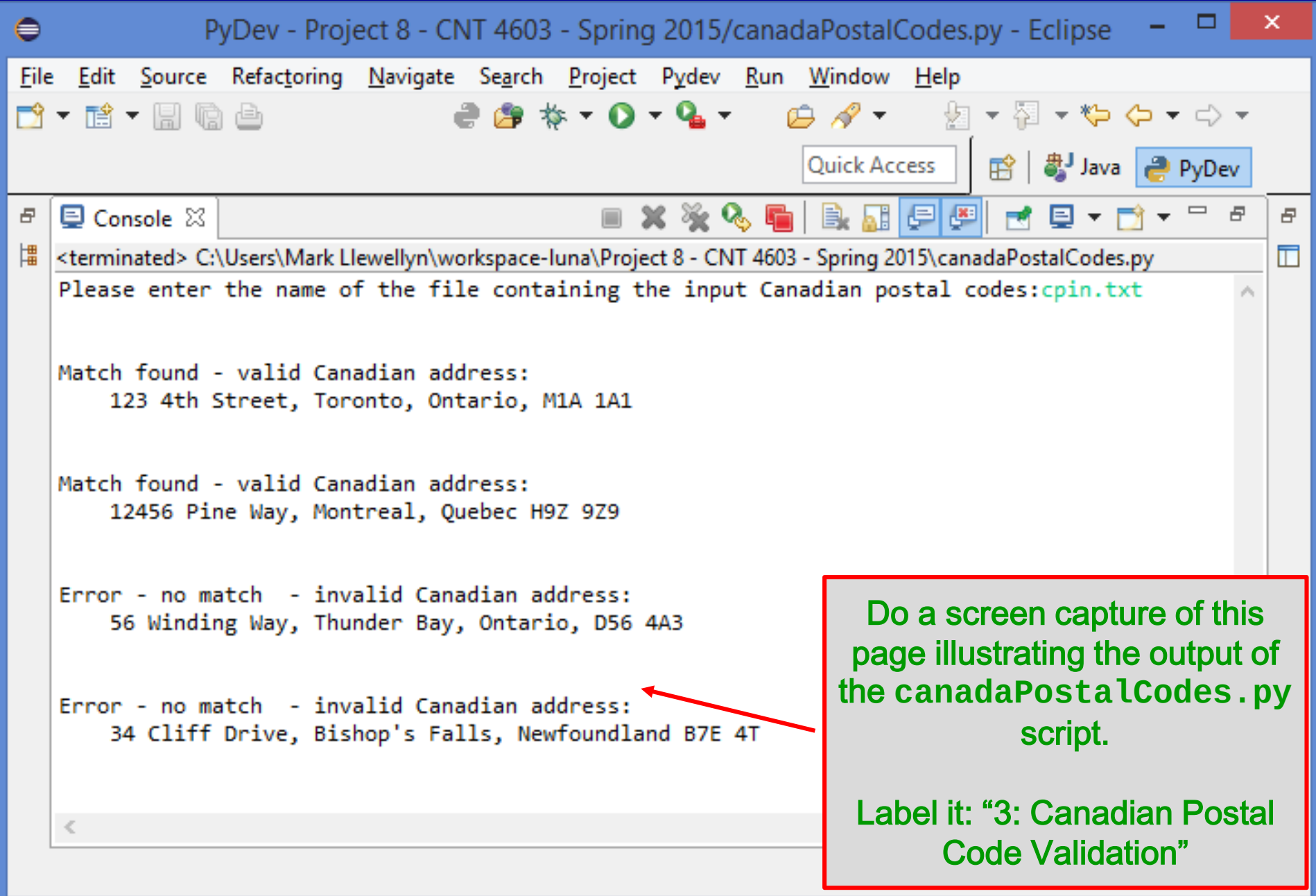
Project Eight – Script Three – `canadaPostalCodes.py`

- This script will validate Canadian postal codes within Canadian addresses.

- Canadian postal codes contain six characters made up of alternating characters and digits. The first series of three characters and digits identifies the forward sortation area (or FSA), and the second series of three characters and digits identifies the local delivery unit (or LDU). The first character of the FSA identifies the province, territory, or region. Only 18 different characters are valid in this first position, A is for Newfoundland and Labrador, B is for Nova Scotia, and so on. The 18 valid characters are: A,B,C,E,G,H,J, K, L, M, N, P, R, S, T, V, X, and Y.

Project Eight – Script Three – `canadaPostalCodes.py`

- Validation of Canadian mailing addresses should check that the first character is from this set of valid characters in addition to the formatting of the six characters that make up the postal code.
- The FSA and the LDU are separated by a space.
- Examples: 12 Pine Street, Toronto, Ontario M1A 1A1 is valid, 12 Pine Street, Montreal, Quebec D1E 7R7 is invalid.
- Source file name: `canadaPostalCodes.py`.
- Input file name: `cpcin.txt`.
- Submit your script source file `canadaPostalCodes.py` and the screen shot shown on page 10.



Do a screen capture of this page illustrating the output of the `canadaPostalCodes.py` script.

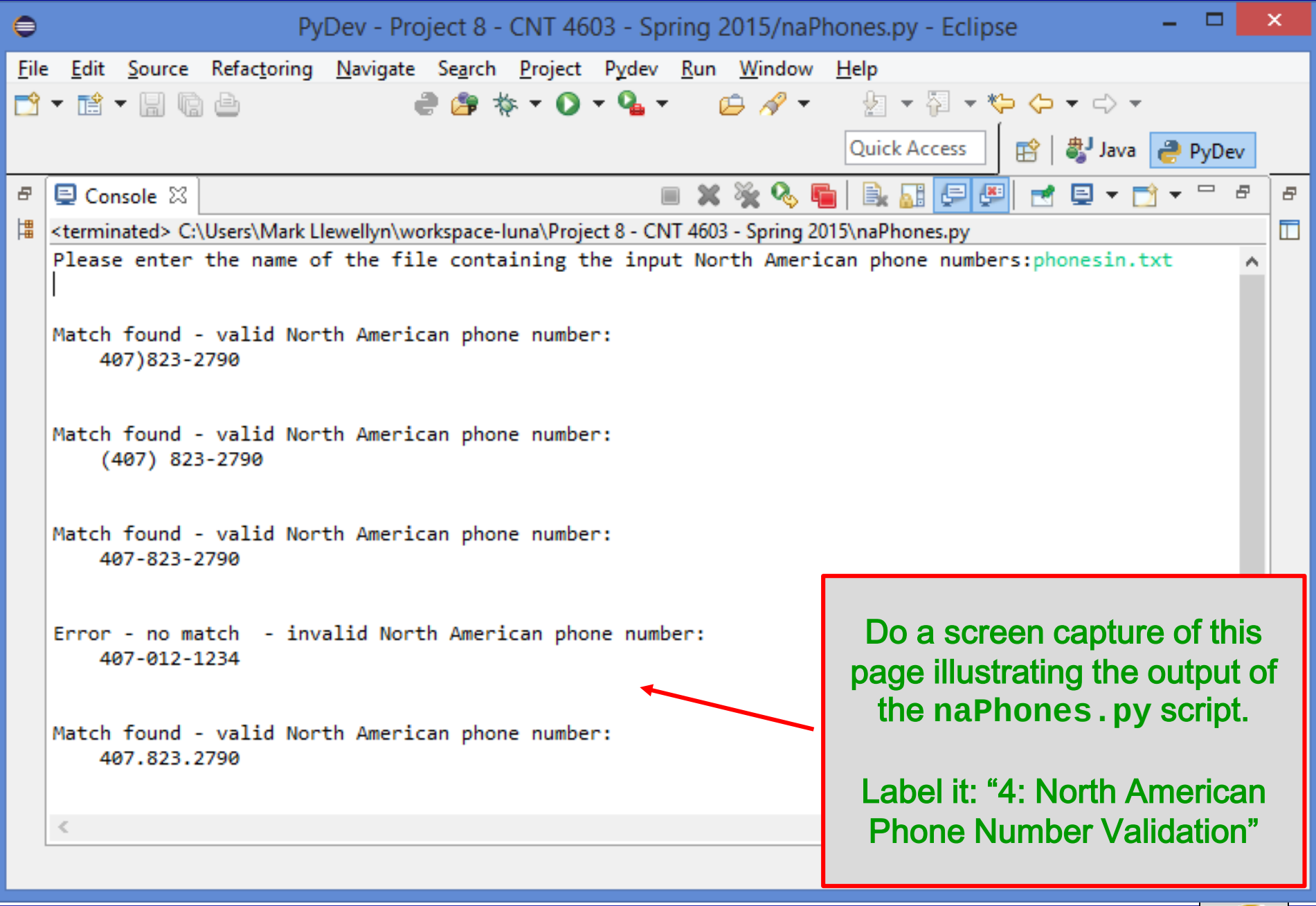
Label it: "3: Canadian Postal Code Validation"

Project Eight – Script Four – naPhones.py

- This script will validate North American phone numbers.
- The North American Numbering Plan defines how North American telephone numbers are formatted. As per the plan, telephone numbers in the US, Canada, much of the Caribbean, and other areas, are made up of a three-digit area code (technically, the NPA or numbering plan area) and then a seven-digit number, which is formatted as a three-digit prefix followed by a hyphen and a four-digit line numbers.
- Any digits may be used in a phone number with two exceptions: The first digit of the area code and the first digit of the prefix may not be 0 or 1.

Project Eight – Script Four – naPhones.py

- The area code is often represented enclosed in parentheses, or separated from the actual phone number by a hyphen.
- Additionally, phone numbers are sometimes represented with periods between each sequence of the digits.
- Examples: (407)823-2790, (407) 823-2790, 407-823-2790, and 407.823.2790 are all valid phone numbers in North America. (012) 111-2345, 112-223-456, and (821) 045-9876 are all invalid.
- Source file name: `naPhones.py`.
- Input file name: `phonesin.txt`.
- Submit your script source file `naPhones.py` and the screen shot shown on page 13.



Do a screen capture of this page illustrating the output of the naPhones.py script.

Label it: "4: North American Phone Number Validation"