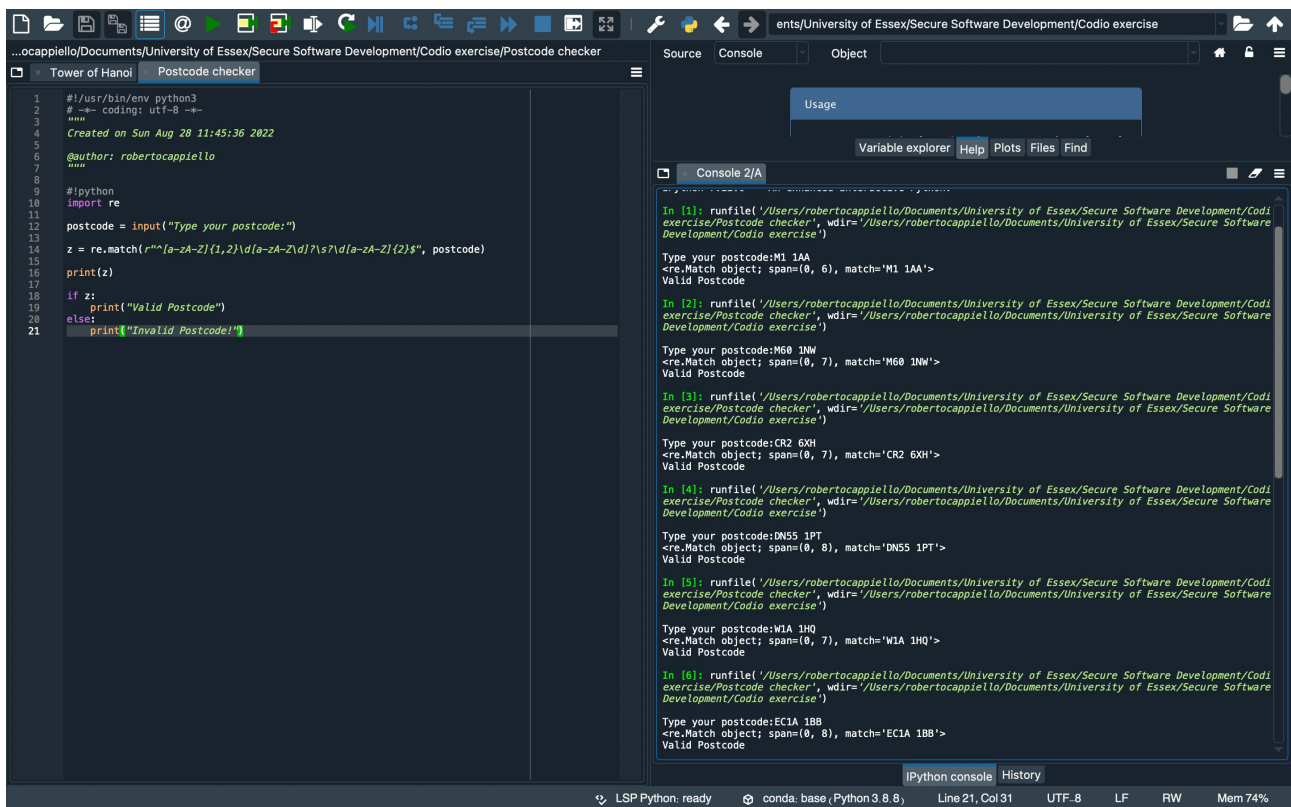


## Regex

The second language concept we will look at is regular expressions (regex). We have already presented some studies on their use, and potential problems, above. The lecturecast also contains a useful link to a tutorial on creating regex. Re-read the provided links and tutorial (Jaiswal, 2020) and then attempt the problem presented below:

- The UK postcode system consists of a string that contains a number of characters and numbers – a typical example is ST7 9HV (this is not valid – see below for why). The rules for the pattern are available from [idealpostcodes \(2020\)](#).
- Create a python program that implements a regex that complies with the rules provided above – test it against the examples provided.
- Examples:
  - M1 1AA
  - M60 1NW
  - CR2 6XH
  - DN55 1PT
  - W1A 1HQ
  - EC1A 1BB

Here is a screenshot proving that the code is working and the above postcodes are being tested and valid UK postcodes



The screenshot shows a Jupyter Notebook interface with a file named 'Postcode checker'. The code in the notebook is as follows:

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Sun Aug 28 11:45:36 2022
5
6 @author: robertocappiello
7 """
8
9 #python
10 import re
11
12 postcode = input("Type your postcode:")
13
14 z = re.match(r"[a-zA-Z]{1,2}[d[a-zA-Z]{1,2}]s?[d[a-zA-Z]{2}]s", postcode)
15
16 print(z)
17
18 if z:
19     print("Valid Postcode")
20 else:
21     print("Invalid Postcode!")
```

The console output shows the following interactions:

```
In [1]: runfile('/Users/robertocappiello/Documents/University of Essex/Secure Software Development/Codio exercise/Postcode checker', wdir='/Users/robertocappiello/Documents/University of Essex/Secure Software Development/Codio exercise')
Type your postcode:M1 1AA
<re.Match object; span=(0, 6), match='M1 1AA'>
Valid Postcode

In [2]: runfile('/Users/robertocappiello/Documents/University of Essex/Secure Software Development/Codio exercise/Postcode checker', wdir='/Users/robertocappiello/Documents/University of Essex/Secure Software Development/Codio exercise')
Type your postcode:M60 1NW
<re.Match object; span=(0, 7), match='M60 1NW'>
Valid Postcode

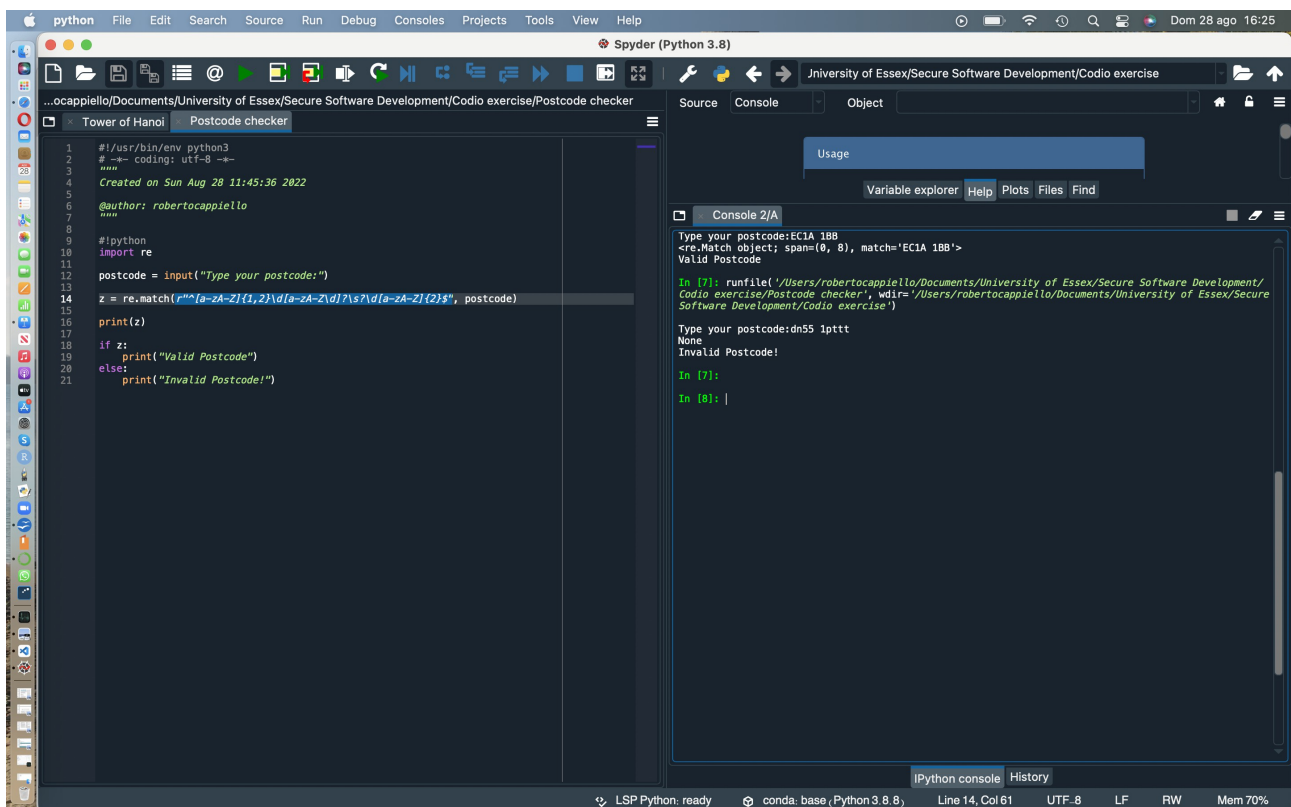
In [3]: runfile('/Users/robertocappiello/Documents/University of Essex/Secure Software Development/Codio exercise/Postcode checker', wdir='/Users/robertocappiello/Documents/University of Essex/Secure Software Development/Codio exercise')
Type your postcode:CR2 6XH
<re.Match object; span=(0, 7), match='CR2 6XH'>
Valid Postcode

In [4]: runfile('/Users/robertocappiello/Documents/University of Essex/Secure Software Development/Codio exercise/Postcode checker', wdir='/Users/robertocappiello/Documents/University of Essex/Secure Software Development/Codio exercise')
Type your postcode:DN55 1PT
<re.Match object; span=(0, 8), match='DN55 1PT'>
Valid Postcode

In [5]: runfile('/Users/robertocappiello/Documents/University of Essex/Secure Software Development/Codio exercise/Postcode checker', wdir='/Users/robertocappiello/Documents/University of Essex/Secure Software Development/Codio exercise')
Type your postcode:W1A 1HQ
<re.Match object; span=(0, 7), match='W1A 1HQ'>
Valid Postcode

In [6]: runfile('/Users/robertocappiello/Documents/University of Essex/Secure Software Development/Codio exercise/Postcode checker', wdir='/Users/robertocappiello/Documents/University of Essex/Secure Software Development/Codio exercise')
Type your postcode:EC1A 1BB
<re.Match object; span=(0, 8), match='EC1A 1BB'>
Valid Postcode
```

here is an example of recognising an invalid postocde



The screenshot shows the same Spyder Python IDE interface with the 'Postcode checker' script. The console output now includes an invalid postcode:

```
In [7]: runfile('/Users/robertocappiello/Documents/University of Essex/Secure Software Development/Codio exercise/Postcode checker', wdir='/Users/robertocappiello/Documents/University of Essex/Secure Software Development/Codio exercise')
Type your postcode:dn55 1pttt
None
Invalid Postcode!

In [7]:
In [8]:
```

- How do you ensure your solution is not subject to an evil regex attack?

According to OWASP it is possible to avoid Evil Regex pattern by avoiding:

- Grouping with repetition
- Inside the repeated group:
- Repetition
- Alternation with overlapping

#### Examples of Evil Regex:

- `(a+)+`
- `([a-zA-Z]+)*`
- `(a|aa)+`
- `(a|a?)+`
- `(.*a){x}` for `x > 10`

All the above are susceptible to the input `aaaaaaaaaaaaaaaaaaaaaaa!` (The minimum input length might change slightly, when using faster or slower machines).

#### References

owasp.org. (n.d.). *Regular expression Denial of Service - ReDoS* | OWASP. [online] Available from [https://owasp.org/www-community/attacks/Regular\\_expression\\_Denial\\_of\\_Service\\_-\\_ReDoS](https://owasp.org/www-community/attacks/Regular_expression_Denial_of_Service_-_ReDoS)