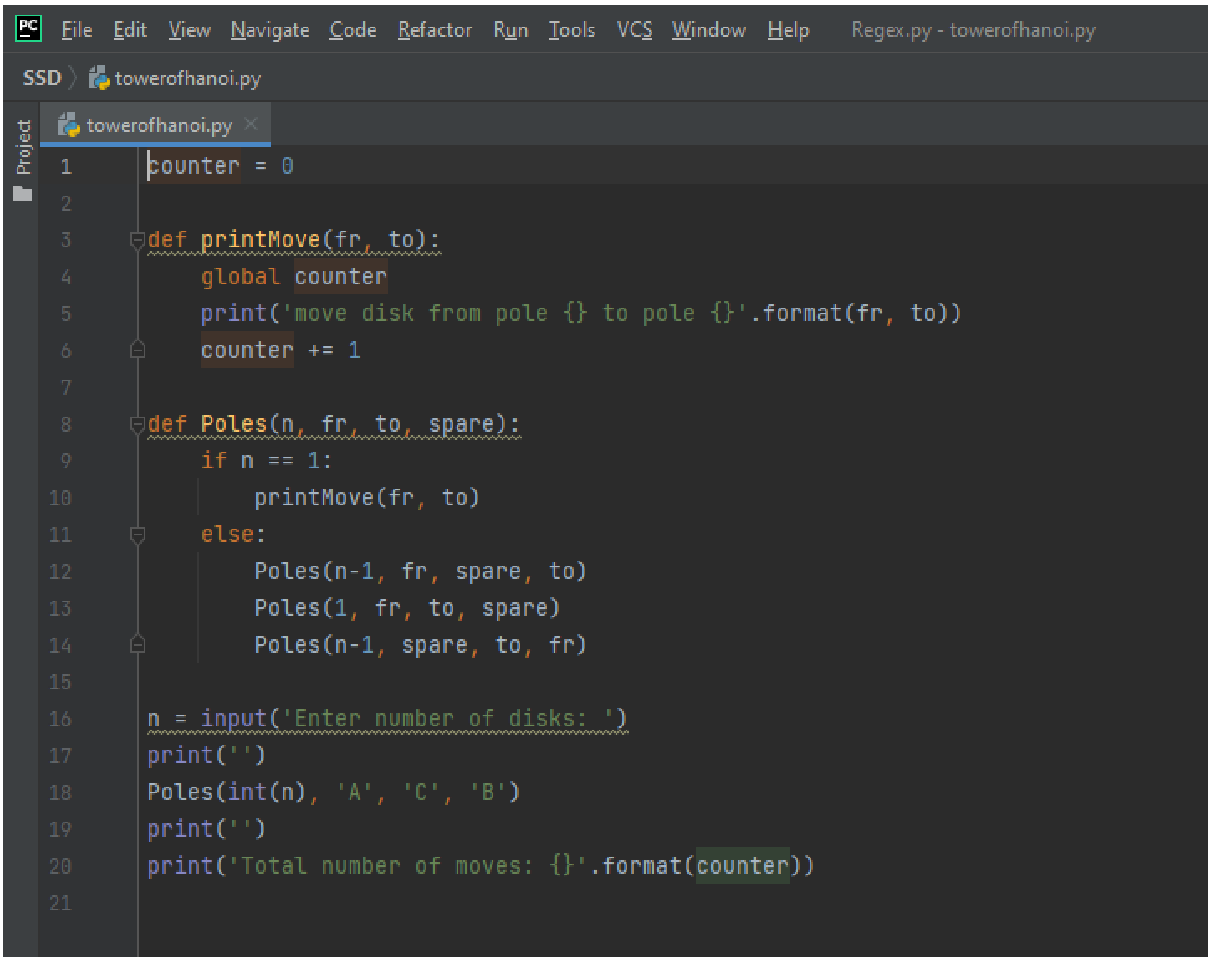
Seminar 2 Preparation: Practical Activity

Recursion

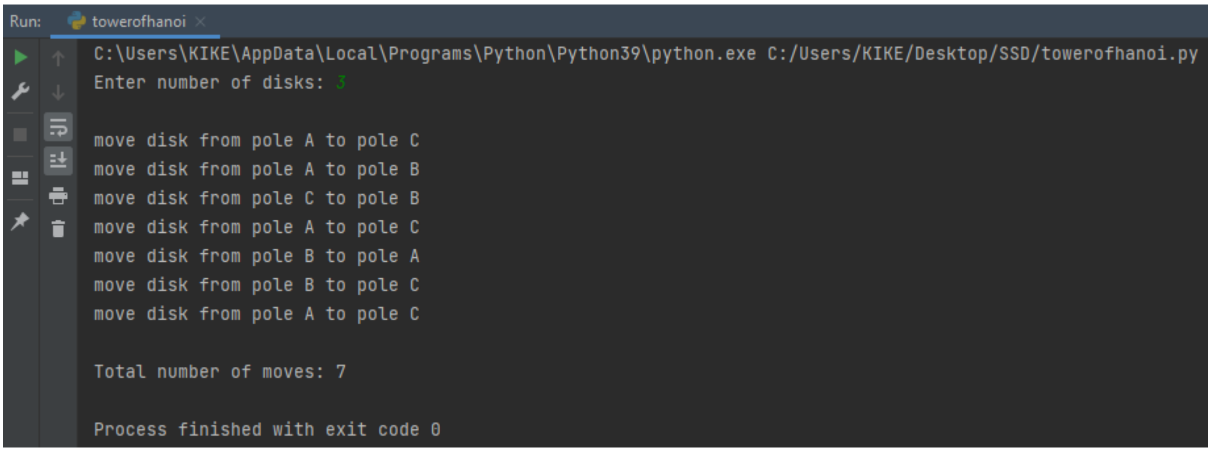
- Tower of Hanoi: create a version that asks for the number of disks and then executes the moves, and then finally displays the number of moves executed.

  [code]



```python
counter = 0

def printMove(fr, to):
    global counter
    print('move disk from pole {} to pole {}'.format(fr, to))
    counter += 1

def Poles(n, fr, to, spare):
    if n == 1:
        printMove(fr, to)
    else:
        Poles(n-1, fr, spare, to)
        Poles(1, fr, to, spare)
        Poles(n-1, spare, to, fr)

n = input('Enter number of disks: ')
print('')
Poles(int(n), 'A', 'C', 'B')
print('')
print('Total number of moves: {}'.format(counter))
```

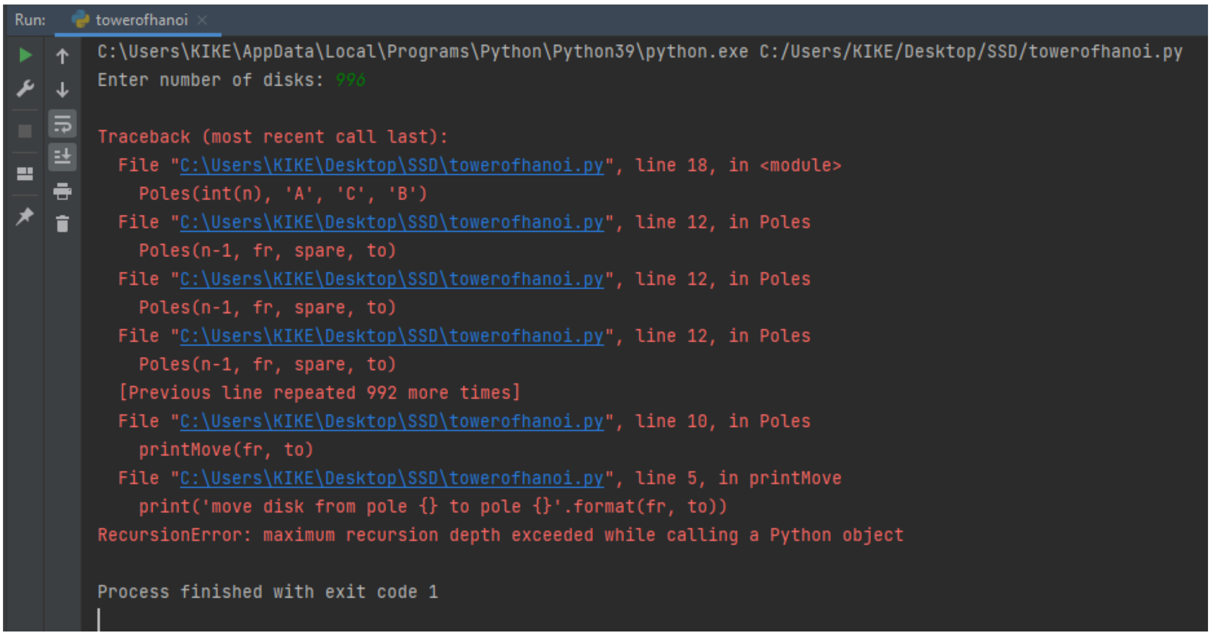
  [Output]



```
C:\Users\KIKE\AppData\Local\Programs\Python\Python39\python.exe C:/Users/KIKE/Desktop/SSD/towerofhanoi.py
Enter number of disks: 3

move disk from pole A to pole C
move disk from pole A to pole B
move disk from pole C to pole B
move disk from pole A to pole C
move disk from pole B to pole A
move disk from pole B to pole C
move disk from pole A to pole C

Total number of moves: 7

Process finished with exit code 0
```

- What is the (theoretical) maximum number of disks that your program can move without generating an error? 995 disks
- What limits the number of iterations? What is the implication for application and system security?

At disk no 996, the error below is shown



```
C:\Users\KIKE\AppData\Local\Programs\Python\Python39\python.exe C:/Users/KIKE/Desktop/SSD/towerofhanoi.py
Enter number of disks: 996

Traceback (most recent call last):
  File "C:\Users\KIKE\Desktop\SSD\towerofhanoi.py", line 18, in <module>
    Poles(int(n), 'A', 'C', 'B')
  File "C:\Users\KIKE\Desktop\SSD\towerofhanoi.py", line 12, in Poles
    Poles(n-1, fr, spare, to)
  File "C:\Users\KIKE\Desktop\SSD\towerofhanoi.py", line 12, in Poles
    Poles(n-1, fr, spare, to)
  File "C:\Users\KIKE\Desktop\SSD\towerofhanoi.py", line 12, in Poles
    Poles(n-1, fr, spare, to)
  [Previous line repeated 992 more times]
  File "C:\Users\KIKE\Desktop\SSD\towerofhanoi.py", line 10, in Poles
    printMove(fr, to)
  File "C:\Users\KIKE\Desktop\SSD\towerofhanoi.py", line 5, in printMove
    print('move disk from pole {} to pole {}'.format(fr, to))
RecursionError: maximum recursion depth exceeded while calling a Python object

Process finished with exit code 1
```

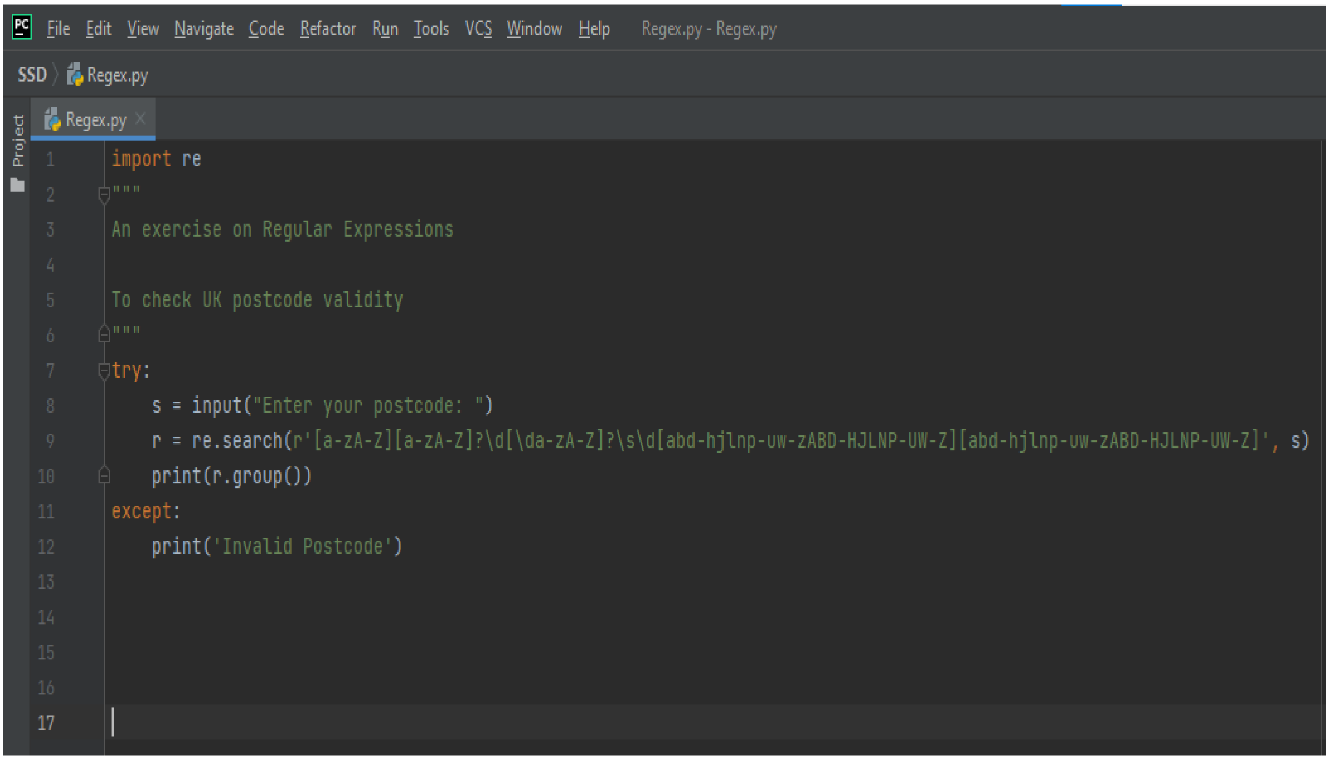
RecursionError: maximum recursion depth exceeded while calling a Python object

The python interpreter limits the number of recursions to avoid an application running an infinite number of recursions leading to stack overflow (Geekforgeeks, 2021). If this vulnerability is not addressed, an attacker can inject a malicious code that will overflow the program stack and interrupt its normal processes giving the attacker control over the system (Rahman et al., 2020)

Regex

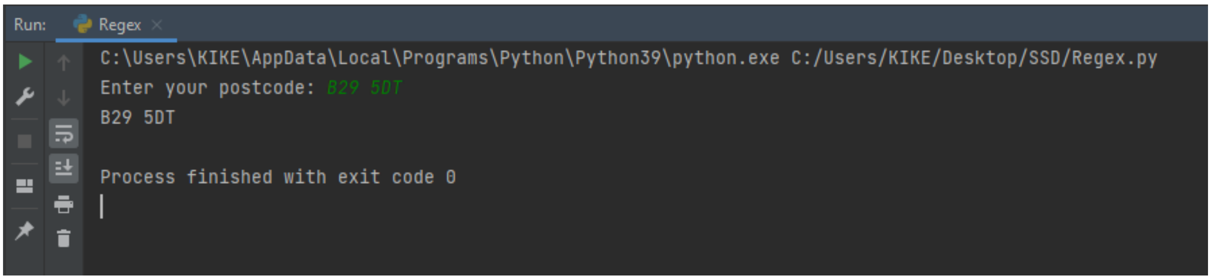
- The UK postcode system consists of a string that contains a number of characters and numbers – a typical example is ST7 9HV (this is not valid – see below for why). The rules for the pattern are available from idealpostcodes (2020).
- Create a python program that implements a regex that complies with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

[Code]


File Edit View Navigate Code Refactor Run Tools VCS Window Help
Regex.py - Regex.py
SSD
Regex.py
Project
Regex.py

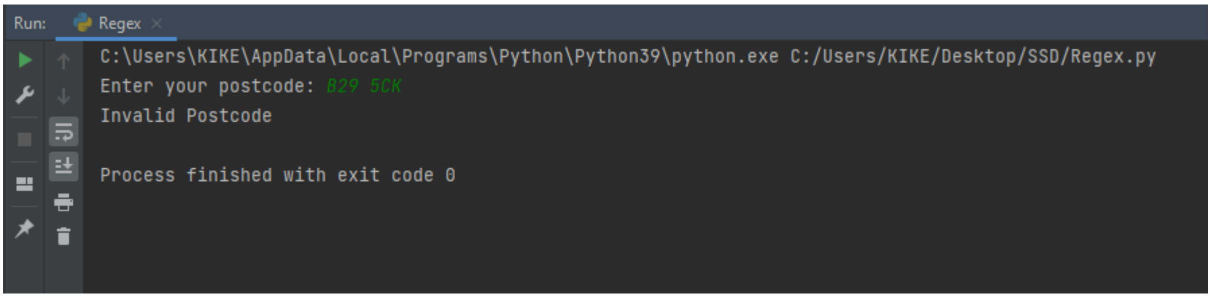
```python
import re
"""
An exercise on Regular Expressions

To check UK postcode validity
"""
try:
    s = input("Enter your postcode: ")
    r = re.search(r'[a-zA-Z][a-zA-Z]?\d[\da-zA-Z]?\s\d[abd-hjlnp-uw-zABD-HJLNP-UW-Z][abd-hjlnp-uw-zABD-HJLNP-UW-Z]', s)
    print(r.group())
except:
    print('Invalid Postcode')
```

[Output 1]: Valid postcode



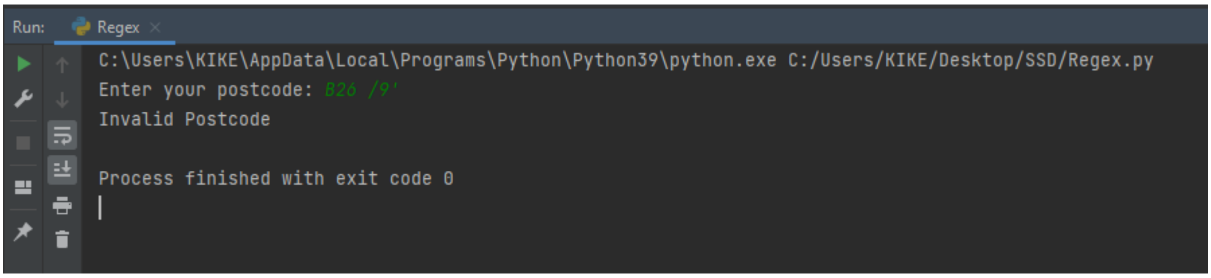
```
C:\Users\KIKE\AppData\Local\Programs\Python\Python39\python.exe C:/Users/KIKE/Desktop/SSD/Regex.py
Enter your postcode: B29 5DT
B29 5DT

Process finished with exit code 0
```

[Output 2]: Invalid postcode with any of the “CIKMOV” letters



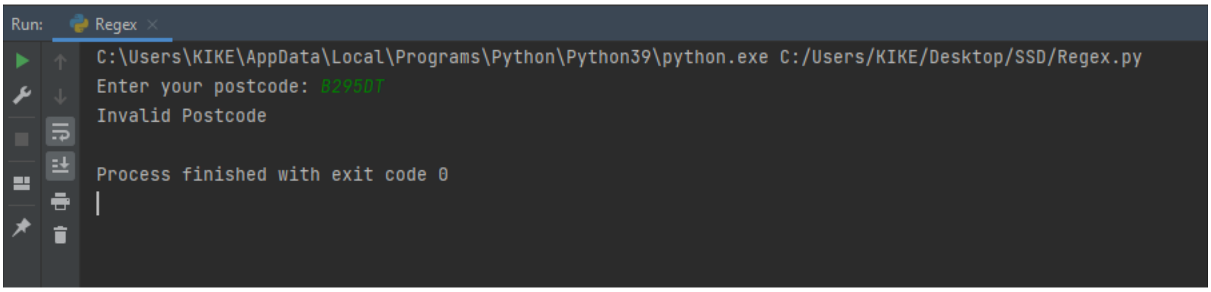
```
C:\Users\KIKE\AppData\Local\Programs\Python\Python39\python.exe C:/Users/KIKE/Desktop/SSD/Regex.py
Enter your postcode: B29 5CK
Invalid Postcode

Process finished with exit code 0
```

[Output 3]: Invalid postcode with an invalid character



```
C:\Users\KIKE\AppData\Local\Programs\Python\Python39\python.exe C:/Users/KIKE/Desktop/SSD/Regex.py
Enter your postcode: B26 /9'
Invalid Postcode

Process finished with exit code 0
```

[Output 4]: Invalid post code with missing space



```
C:\Users\KIKE\AppData\Local\Programs\Python\Python39\python.exe C:/Users/KIKE/Desktop/SSD/Regex.py
Enter your postcode: B295DT
Invalid Postcode

Process finished with exit code 0
```

How do you ensure your solution is not subject to an evil regex attack? By ensuring that the code does not contain characters (or combination of characters) prone to evil regex attacks as explained below.

Evil Regex

Evil Regex is defined as a pattern that could get stuck on crafted input due to contents such as (Weidman, n.d):

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

Some examples are (a+)+, ([a-zA-Z]+)*, (a|aa)+, (a|a?)+, (.*a){x} for x\>10

References

Geekforgeeks (2021) Python: Handling Recursion Limit. Available from: https://www.geeksforgeeks.org/python-handling-recursion-limit/ [Accessed 03 September 2021]

Rahman, M.M., Satter, A. and Hossain, B.M., (2020) An Empirical Study on Stack Overflow Security Vulnerability in Well-known Open Source Software Systems. International Journal of Computer Applications. 176 (39): 0975-8887.

Weidman A. (n.d) Regular Expression – Denial of Service. Available from: https://owasp.org/www-community/attacks/Regular_expression_Denial_of_Service_-_ReDoS [Accessed 02 September 2021]