# Homework 4 Regular expressions Answers

*NB: You can edit the first two questions to match your own school’s tutor group naming conventions if required.*

1. Julia has written some Python code, but it won’t run. She has made 3 syntax errors – highlight them and explain each one to Julia. [3]

import re  
tutorGroup = input(“Enter your tutor group – e.g. 10MC: ”)  
valid = re.match(“[0-9][0-9][A-Z][A-Z]”,tutorGroup)  
if valid:  
 print(“OK!”)  
else:  
 print(“Invalid!”)

1. Line 1: Incorrect name for the library, should be ‘import re’.

2. Line 3: 4th element in the pattern said [a-Z] but should be [A-Z]. If upper and lower case is acceptable then [a-zA-Z] should be used.

3. Line 3: tutorGroup variable should not be in speech marks unless you are checking against the **string** “tutorGroup” rather than the variable. The speech marks should be around the regular expression.

1. George wants to alter Julia’s program so that it will accept tutor groups with 1 or more numbers (e.g. 7MC, 10MC, 12MC). Write just the one line of code that George needs to edit and explain what he needs to do. [2]  
     
   valid = re.match([0-9]+[A-Z][A-Z],tutorGroup)  
   [0-9]+ will match one **or more** digits, allowing 1 or 2 digit tutor groups.

1. Darcey has been looking at a program that will collect and then validate IP addresses   
   (e.g. 192.168.0.37). IP addresses are always made up of 4 block of digits, separated by full stops. Add comments to explain how the program works.  
   NB: The program is only meant to check for the correct pattern, it doesn’t check if each element is between 0 and 255) [5]  
     
   # The first line imports the regular expression library which  
   # is needed to run the regular expression match  
   import re  
   ip = input(“Enter IP address: ”)  
   # This line matches the IP address typed in with a pattern:  
   # [0-9]+ means “at least one number”  
   # \. means “a full stop”  
   valid = re.match(“[0-9]+\.[0-9]+\.[0-9]+\.[0-9]”,ip)  
   # If the match has been successful, ‘valid’ will have the value True  
   # If it hasn’t then the value of ‘valid’ will be False  
   if valid:  
    print(“IP address OK”)  
   else:  
    print(“IP address invalid”)

4. There are four valid telephone number formats in the UK. These are:

(01nnn) nnnnnn

(01n1) nnn nnnn

(011n) nnn nnnn

(02n) nnnn nnnn

Write regular expressions for each of these formats. The first one is done for you. Note that [0-9]{6} means “exactly 6 instances of any digit between 0 and 9”. [3]

pattern1 = “(01[0-9]{3}) [0-9]{6}”

pattern2 = "(01[0-9]1) [0-9]{3} [0-9]{4}"

pattern3 = "(011[0-9]) [0-9]{3} [0-9]{4}"

pattern4 = "(02[0-9]) [0-9]{4} [0-9]{4}"

Write a Python statement to test whether a phone number conforms to one of the patterns. If it does, print “valid phone number”. If it does not, print “Invalid phone number”. (Tip: You could use an IF..ELIF” statement) [5]

if re.match(pattern1, phonenumber):

print("valid phone number")

elif re.match(pattern2, phonenumber):

print("valid phone number")

elif re.match(pattern3, phonenumber):

print("valid phone number")

elif re.match(pattern4, phonenumber):

print("valid phone number")

else:

print("invalid phone number")  
 [Total 18 Marks]