## Python Exercises – by Mgcini Phuthi

## Flow Control and Loops

1. Define a function called strip that removes the white spaces at the beginning and end of a string no matter how many there are. For example, if the string is '  hat  ', the function should return 'hat'. Do it two different ways, using a loop and using recursion.
2. Remember the number guessing game? Write a program that solves the number guessing game i.e. The computer should play against itself. (Hint: Change the guessing game so that it returns booleans that you can test with. [Solution added]
3. Define a function that allows a user to do the basic math calculations. This should not be done directly on the shell. The function should ask for a basic math statement like '1+1' and then find the answer. The key here is that input() always returns a string so your program should figure out what operation (+,-,/ and * ) the user wants to do and find the answer. Do this only for the case where there are no spaces in the users input but it should work for any number. The function should print the answer and return None.
4. Define a function that takes in a string as a parameter and counts how many number digits are in the string e.g. If the string is 'dfsd78dsf90' it should return 4. Make the function print the position of each digit.

## Data structures

5. Define a function that stores a students marks in a list from input and returns the list.
6. Define a function that takes in a list of floats as a parameter and returns the average of the numbers in the list.

## 7. Good example: Email verification

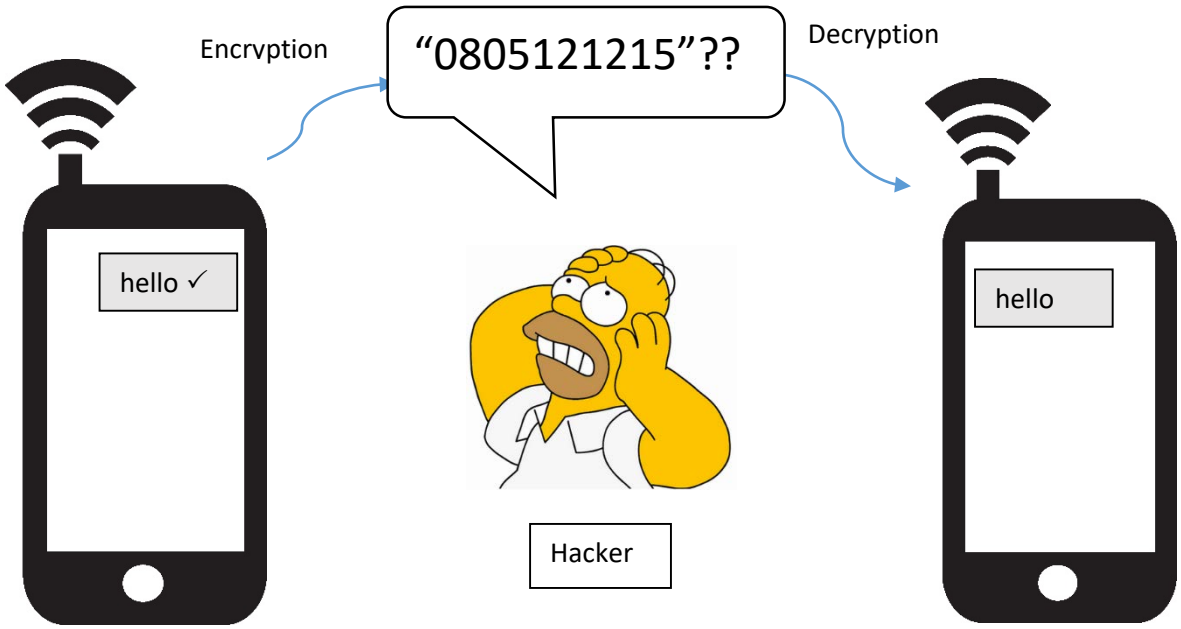
When a website asks you to enter an email, you will notice that if you enter a bad email address, it tells you that you are wrong. An email address must always take the form : 'username@domain' where 'username' is for the client(the person who owns the address) and the 'domain' is for the mail provider (e.g. gmail or yahoo)  e.g. admire@zimcode.org. The domain should have a '.' somewhere, it can have many dots e.g. thamsanqa@email.co.zw but the whole address can only have 1 @ symbol. Define a function that takes in a string and checks if the string is a valid email address. For example if the parameter is 'rubbishemail' or 'rubbishemail.com' or rubbish@email@rubbish.co.zw' or 'rubbish.email' it will return False. If the email is valid like 'not@rubbish.email' it should return True. Your function should use indexing and flow control.

**Side Story –** This is yet another practical example. Websites and apps need verification systems to make sure people enter their details correctly otherwise people would make mistakes and it would waste everyone's time. This process of making sure input data is correct is called data 'validation'. This has to be done for everything that has to be stored in a database (A very big data structure!). You wrote a function that validates email addresses but this is not how it's actually done anymore. Today they use something called 'regex' which stands for 'regular expressions'. Continue learning Python and one day they'll be your best friend.

```
def validate(email):
    '''
    Validates an email addreess by checking that it contains only 1 @ sign
    and that there is at least one period '.' after the @ sign
    email -> str
    returns bool
    '''
    #First check that there is only one '@' symbol
    if email.count('@') > 0 and email.count('@'):
        #Find the position of the '@' symbol
        symbol_index = email.index('@')
        #Make sure there is a username before the @ symbol
        if symbol_index > 0:
            if '.' in email[symbol_index:] and email[symbol_index+1] != '.' and email[-1] != '.' :
                #Use slicing to check if there is a '.' after the @ symbol
                #Then check that it is not directly after the '@' symbol
                #Then check that the '.' is not at the end of the email
                return True
    #If any of the tests are False, return False
    return False
```

## 8. Good Example: Encryption and Decryption

You may have heard the word encryption before e.g. your WhatsApp messages are encypted. This means that Whatsapp changes the message so that other people (hackers) can't read it. For example, when you send a message like 'hello', WhatsApp can encrypt the message so that the message looks like '0805121215'. If you saw this, you cannot know what this means unless you know how to 'decrypt' the message, this is what happens on your friend's phone so that he/she can read the message.




How do we encrypt then? We encrypt using a key. A key tells you how to change a message to an encrypted message and how to change an encrypted message back to the original message. For example, the key used above uses the position of the letter in the alphabet as a 2 digit code. For example 'h' is the 8th letter in the alphabet so it is letter number '08', 'e' is the 5th letter so it is '05', 'l' is number '12' etc.

The important thing when encrypting a message is to use a good key. For our simple case, we want our key to have a code matching each letter of the alphabet e.g '08' matches 'h', 08 does not match

any other character and ‘h’ does not match any other code. We say our key should be “unambiguous” (You don’t need to know this or how to pronounce it!”). The key we used above is shown below:

| a | b | c | d | e | f | g | h | i | j | k | l | m |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 |
| n | o | p | q | r | s | t | u | v | w | x | y | z |
| 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 |

**Class exercise**

1. Write a function called **encrypt(**message**)** that takes in a message and encrypts it using the order of the characters on your keyboard i.e. 'qwertyuiopasdfghjklzxcvbnm' replaces 'abcdefghijklmnopqrstuvwxyz'. Make sure your function ignores all other characters and leaves them unchanged e.g. periods (fullstops) should still be periods.
2. Write a function that decrypts an encrypted message using the same key (otherwise it doesn’t work!)
3. Send secret messages to your friends and ask them to use their decryption function to figure out what they mean. You can even edit your code so that it asks for input from the user to encrypt/decrypt.

**The solution is given below**

**Student Exercise**

1. Use the encryption key in the example (the numbers) to write a function to make your own encryptions but this time make sure it works for uppercase characters e.g. ‘h’ should be different from ‘H’. To do this you can just make character 27 == ‘A’, 28 == ‘B’ etc.
2. Use the same key to write a function that decrypts the message.

## Story about encryption

Data encryption is a big thing in computer science, the field is called “cryptography”. Recently the Harare Institute of Technology was hacked with the “WannaCry” ransomware. The ransomware encrypted all the data on the computer so that it becomes unusable unless it is decrypted. The hackers have the key and wanted money so that they can decrypt the data.

Encryption is not as easy as we made it look here. In fact, most encryption software used today makes sure that no one can ever guess the key. For example I can make rules like:

a) The symbol for ‘a’ is ‘$j39’
b) If the ‘a’ is after an ‘e’ or ‘f’ change the code to ‘2##$’
c) Every 3rd ‘a’ in a message should have its code reversed
d) Any ‘a’ that is second in a sentence should have the numbers replaced by a symbol given by another key.

These rules are all for the letter ‘a’ only and I can keep adding more, there is no way anyone can ever guess. Encryption used today is still far better than this!

You can read about the Wannacry ransomware now that you know more about encoding!

## Solution

```
def encrypt(message, key = 'qwertyuiopasdfghjklzxcvbnm'):
    '''
    Encrypts a message using the key defined here. The key can be
    anything but must have 26 distinct characters so that
    the encryption is not ambiguous.
    message -> str
    returns -> str
    '''
    alphabet = 'abcdefghijklmnopqrstuvwxyz'
    key = 'qwertyuiopasdfghjklzxcvbnm'
    enc_message = ''

    #Take each character in message and find its position in the
    #alphabet
    for char in message:
        if char in alphabet:
            #Find the index of the character in the key
            char_index = alphabet.index(char)
            #Take the char in the same position from the key and add
             #it to the encrypted message
            enc_message += key[char_index]
        else:
            #If the char is not in our key, ignore it
            enc_message += char

    return enc_message

print(encrypt("hello"))

def decrypt(enc_message, key = 'qwertyuiopasdfghjklzxcvbnm'):
    '''
    Decrypts a message using the key defined here, the key must be
    the
    same as the one used to encrypt the message
    enc_message -> str
    message -> str
    '''
    alphabet = 'abcdefghijklmnopqrstuvwxyz'
    key = 'qwertyuiopasdfghjklzxcvbnm'
    message = ''   #Empty string to store decrypted message

    #Take each char in enc_message and find its position in the key
    for char in enc_message:
        if char in key:
            #Find the index of the character in the key
            char_index = key.index(char)
            #Take the char in the same position from the alphabet
             #and add it to the message
            message += alphabet[char_index]
        else:
            #If the char is not in our key, ignore it
            message += char
    return message

print(decrypt("itssg"))
```