LINUX VOICE TUTORIAL

PYTHON 3: BUILD A PASSWORD CHECKER

LES POUNDER

Use programming logic, variables and functions to check the strength of your passwords.

WHY DO THIS?

- We will learn how to accept user input via a graphical user interface and then check the input against an algorithm to ensure that the password input meets a criteria.

TOOLS REQUIRED

- Python 3 installed on your machine.
- EasyGUI installed on your machine.

DISCLAIMER

This project should not be used to check any "real world" passwords for example online banking or shopping.

Keeping our data private is a major concern, and one way to keep our data safe is via a good password. But being human we rarely choose a secure password; instead use insecure data such as the name of our pet, our date of birth or other such details. Data such as this is not really suitable in a password, and we are encouraged to create harder to guess passwords with a mix of both upper and lower case characters along with numbers.

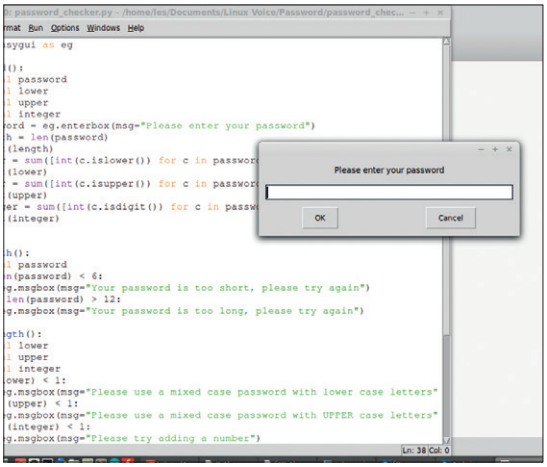
In this project we will be using the popular programming language Python, and we will be writing the code using the Python 3 syntax, which also enables this project to be run on Python 2 systems. This project can be created using any computer, including a Raspberry Pi.

For this project you will need

- **Idle 3** A Python editor for Python 3
- **pip3** A Python package manager
- **EasyGUI** A GUI library for Python

For this tutorial we're using Linux Mint 17, which is based on Ubuntu, and so we will install software using the **apt** package manager. The software is also available for other Linux distributions.

To install *Idle 3* open a terminal and type in the following.

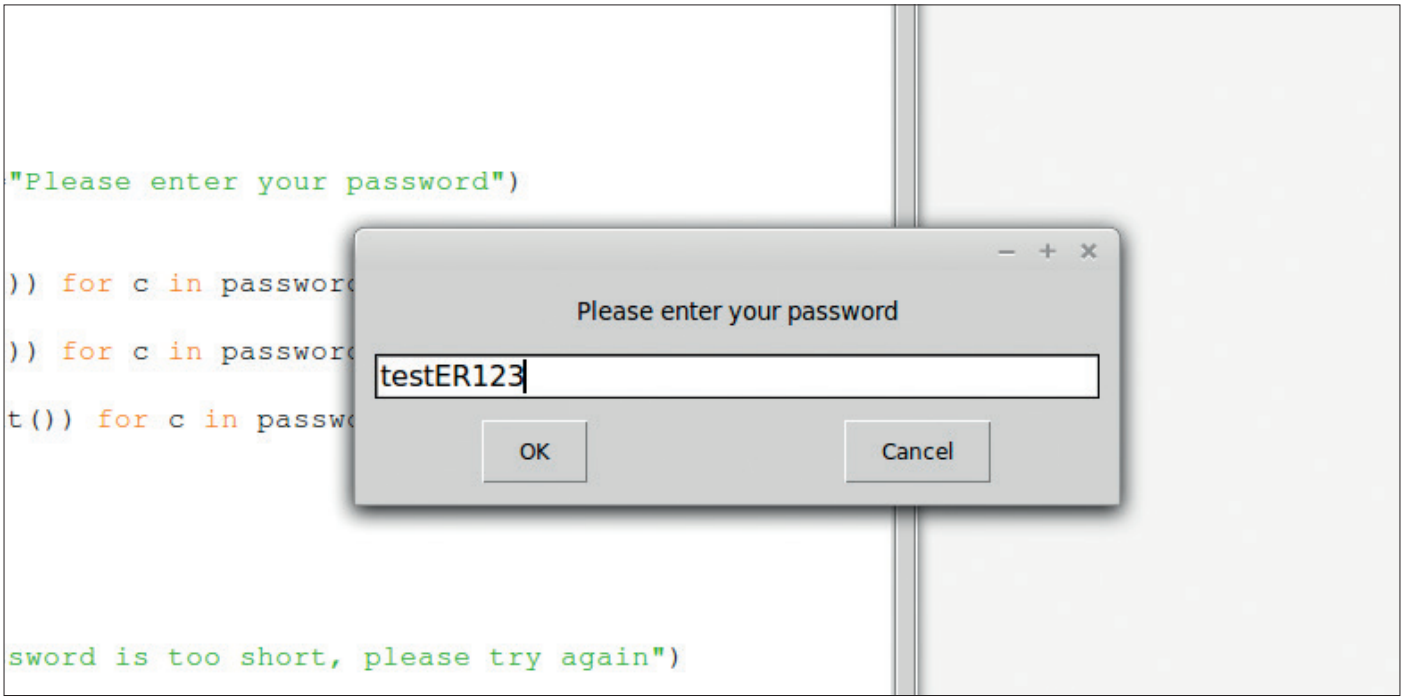


The password checker in action. Our simple application has but one job to do: to keep our passwords strong.

```
sudo apt-get install idle3
```

New to Python 3 is the bundling of the **pip** Python package manager, which works in a similar manner to other Linux package managers. This should be installed by default for your distribution, but if this is not the case open a terminal and type in the following.

```
sudo apt-get install python3-pip
```



Using *EasyGUI* we can easily create a graphical user interface to capture the user's password.

With **pip** for Python 3 installed now we use it to install *EasyGUI*, in a terminal type in the following

```
sudo pip3 install easygui
```

After a few moments *EasyGUI* will be installed and ready for use in our project.

The project

In this project we use Python 3 to write a program that captures the user's password and checks it against a series of conditions. These conditions are:

- The password must have six or more characters.
- The password must have less than or equal to 12 characters.
- There must be at least one integer in the password.
- There must be at least one upper case character.
- There must be at least one lower case character.

We start the project by importing additional modules to expand the abilities of our Python code.

```
import easygui as eg
```

In this example we import the **easygui** module and rename it as **eg**, which makes it a lot easier to work with. With the imports completed we now create a series of functions that handle the input and testing of the password.

Firstly we create the function and name it **pword()**:

```
def pword():
```

Next we create four global variables that are used by our function and other functions later in the project. These global will contain the data for our project.

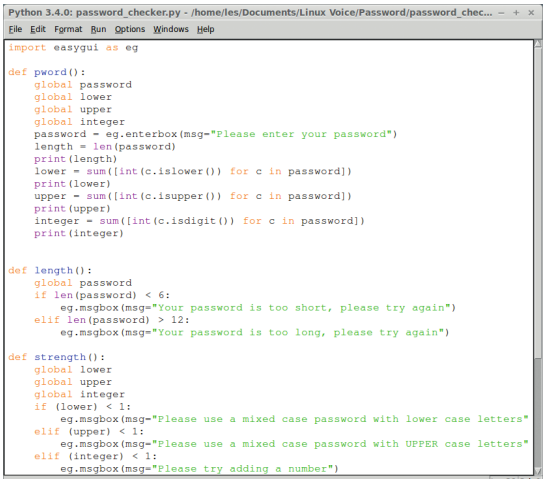
```
    global password
    global lower
    global upper
    global integer
```

With the variables created we now use one of the *EasyGUI* functions to create a dialog box that will enable us to enter our password and then save it as the global variable **password**.

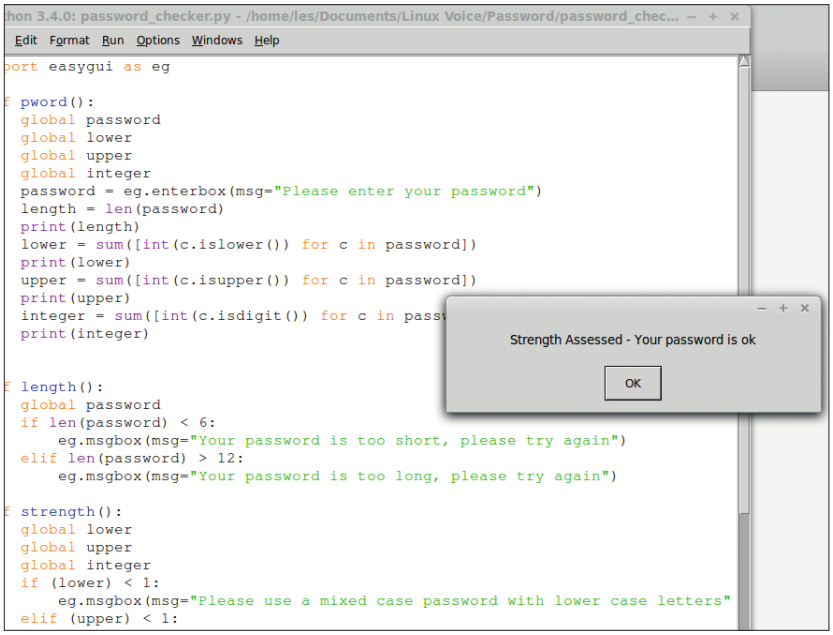
```
    password = eg.enterbox(msg="Please enter your password")
```

With the password captured and stored as a variable we can now do a lot of checks.

```
    length = len(password)
```



There isn't a great deal of code to this project but it is a great meaty project to reinforce your learning and brush up your skills.



If your password meets the criteria then it will be evaluated as a strong password, an essential part of keeping you safe.

```
print(length)
```

Our first check is get the length of the password, which is done using the **len()** function and giving it the name of the variable that we would like to check. The output of this is stored as the variable length, which we then print to the Python shell. We do not need to print the output to the shell, but it can really help to debug a project quickly.

Now that we know how long the password is we need to find out what the password is made of, which will generally be a mix of lower- and upper-case letters along with numbers. We create three variables: **lower**, **upper** and **integer** and they handle lower-case and upper-case characters, and **integers** relates to any numbers in the password. For each of these variables we do the following.

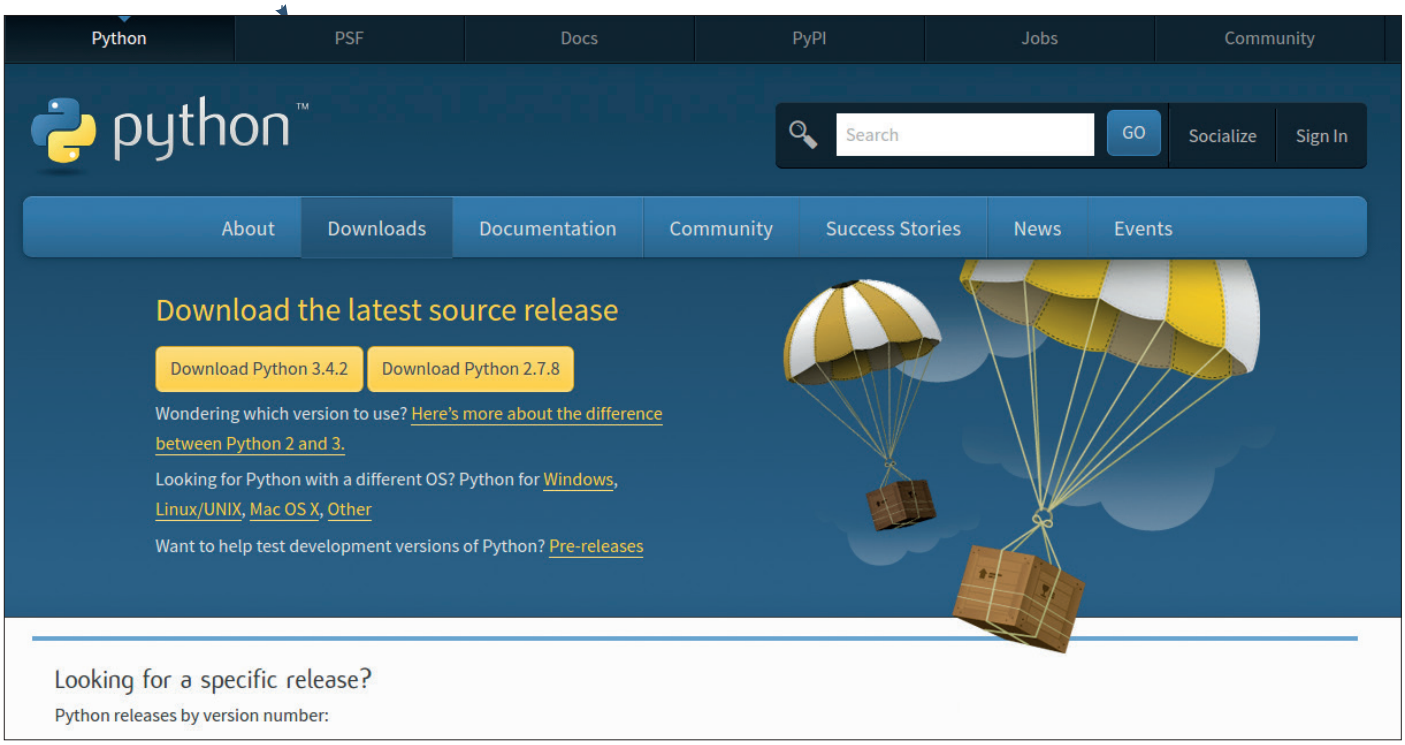
> **"Our first check is to get the length of the password, with the len() function."**

```
Count every lower-case letter in the password and save it as a variable called lower.
Count every upper-case letter in the password and save it as a variable called upper.
Count every integer in the password and save it as a variable called integer.
```

In Python code it looks like this – after each line we also print the value as a debug measure.

```
    lower = sum([int(c.islower()) for c in password])
    print(lower)
    upper = sum([int(c.isupper()) for c in password])
    print(upper)
    integer = sum([int(c.isdigit()) for c in password])
    print(integer)
```

If you are familiar with spreadsheets you will see that the line is not too dissimilar to the syntax needed for *LibreOffice* or *Excel*. The term **c** refers to each character in the password, and we can instruct Python to look for characters that match **islower** **isupper** or **isdigit**. We wrap the value returned in a



Getting the latest version of Python 3 is really simple – you can download it via your distribution's package manager or download the latest package from the Python website.

helper function that converts the data type returned into an integer value **int(c.islower())**. This is not strictly necessary as the value returned is already an integer, but it helps to sanitise the data just in case anyone tries to intentionally break the project. Data sanitisation is best practice and a great skill to learn, as it is used a lot when working with content on the web and in databases.

Our next function has one purpose: to check the length of the password against a strict set of criteria. Its length must be greater than or equal to six characters but it must also be less than or equal to 12 characters in length. A good password should be of a reasonable length, but a long password does not equal a secure password.

In the code snippet following we first define a new function called **length()** and then instruct Python that we wish to use the global variable **password** that we created in our first function.

```
def length():
    global password
```

Next we create an **if**..**elif** conditional statement that will first check to see if the length of the password is less than six characters. If this condition is true we will use EasyGUI's **msgbox** function to generate a pop-up dialog box that advises the user that their password is too short and that they should try again.

```
    if len(password) < 6:
        eg.msgbox(msg="Your password is too short, please try again")
```

Similar to the previous condition we tested, we now use **else if**, which is shortened to **elif** in Python to run another test on the password, which in this case is to see if the password is longer than 12 characters. If this is the case, a dialog box pops up to advise the user to try again.

```
    elif len(password) > 12:
        eg.msgbox(msg="Your password is too long, please try again")
```

Our last function runs a series of tests against the password. These tests are there to assess that the password meets our criteria that we defined earlier in the project.

To start we name our function **strength()** and instruct Python that we wish to use the global variables that we created earlier.

```
def strength():
    global lower
    global upper
    global integer
```

We now reuse an **if..elif** conditional statement that we used in our **length()** function to run three conditional tests.

Firstly we compare the value stored in the variable lower to see if it is lower than 1, which means there are no lower-case characters in the password. If this is correct then we use the *EasyGUI* **msgbox** function to create a dialog box that informs the user that there are no lower-case characters in their password.

```
    if (lower) < 1:
        eg.msgbox(msg="Please use a mixed case password with lower case letters")
```

After the lower-case character test is complete we

Project code

You can find the complete code for this project at our GitHub repository **https://github.com/lesp/LinuxVoice_Issue12_Password_Checker**. For those of you unfamiliar with *Git* you can download the complete package as a Zip file from **https://github.com/lesp/LinuxVoice_Issue12_Password_Checker/archive/master.zip**

Python 3

Python has come a long way since its début in 1989, when it started as the personal project of a chap called Guido van Rossum, who needed a project to keep him busy over the Christmas period. In the passing years Python has improved with each release, and since around 2001 Python 2 has been the default version. In recent years, however, there has been a strong move towards leaving the Python 2 series and moving on to version 3, which has been supported by such organisations as the Raspberry Pi Foundation.

But why should we move to Python 3? Well the most obvious reason is that the Python core developers are no longer working on any code or projects for Python 2. In fact they are so focused on Python 3 that the Python team have created an "un-release schedule for Python 2.8", or in other words, Python 2.7 series is the last of that series. There will be bugfixes and updates for Python 2 for a few years yet, but now is the time to migrate your projects to Python 3.

There are a few changes to Python 3 syntax for example, in Python 2, print is a statement, and will pick up whatever is inside the quotation marks, like so:

```
print "Hello World"
```

In Python 3, print is now a function that comes with a number of arguments to make it a lot more functional, including options to control how content is separated, ended and error control.

```
print("Hello World")
```

User input

In Python 2 **raw_input** is used to capture the keyboard input via the Python shell.

```
raw_input("What is your name?")
```

In Python 3 this has been renamed as **input** but it performs the same actions as **raw_input**.

```
input("What is your name?")
```

Working with lists

In Python 2, working with a list called "names" we pass it two strings, in this case the name of my dog and I. To check the contents we print the list to the shell. Now using del we delete the contents of the list and then use print to check that the list has been deleted.

```
names = ["Les", "Dexter"]
print(names)
del(names)
print(names)
```

In Python 3 lists have been refined with functions that handle printing the contents of the list, in this example **names.copy()** prints the contents of the list. To clear the list we use the **names.clear()** function.

```
names = ["Les", "Dexter"]
names.copy()
names.clear()
names.copy()
```

These are but a few changes made to Python 3. Head over to **https://docs.python.org/3** for a full list of all the changes and additions to Python.

now run the same test looking for upper-case characters.

```
elif (upper) < 1:
    eg.msgbox(msg="Please use a mixed case password with UPPER case letters")
```

Our last test is exactly the same, but this time we are looking for integers in our password.

```
elif (integer) < 1:
    eg.msgbox(msg="Please try adding a number")
```

To close the **if..elif** conditional statement we use an **else** condition which requires no condition to be used. For example, if all of the previous statements evaluate as false then our **else** condition must be true.

```
else:
    eg.msgbox(msg="Strength Assessed - Your password is ok")
```

With our last function completed our attention now turns to threading the functions together into a sequence.

Firstly we instruct Python that we wish to run the following code in an infinite loop, which in Python is **while True**:

```
while True:
```

Inside our infinite loop we call the three functions that we created earlier.

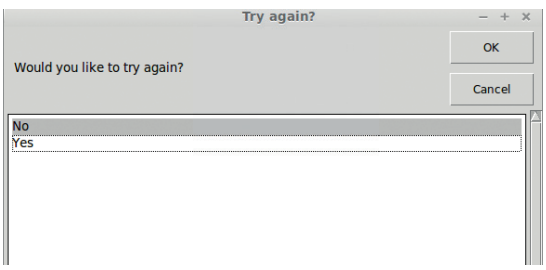
```
pword()
length()
strength()
```

With the functions called and their contents executed, the next part of the loop is a question to the user, asking if they would like to run the program again, and their answer to this question is captured using *EasyGUI*'s **choicebox** dialog. Their choices are limited to Yes and No via the choices argument. The answer is then stored inside a variable called answer.

```
answer = eg.choicebox(title="Try again?", msg="Would you like to try again?", choices=("Yes","No"))
```

In our last section of code we use an **if** statement to check the value of the variable answer. If it does not match "Yes", which in Python can be written as **!=**, then the loop is broken via break and the program ends.

```
if answer != "Yes":
    break
```



The program is designed to loop infinitely enabling many passwords to be checked.

What have we learnt?

In this project we have used Python 3 to develop a simple program which evaluates a password.

- We provided a graphical user interface to capture the password.
- We created a function to evaluate the contents.
- We tested using programming logic to ascertain the number of lower- and upper-case characters along with the number of integers present.
- We used conditional logic to compare the actual results against what we expected to find and where they were different we advised the user as such.

"These tests are there to assess that our password meets the criteria that we defined earlier."

Les Pounder divides his time between tinkering with hardware and travelling the United Kingdom training teachers in the new IT curriculum.