Hyperion Development

Task: Input and Output in Python

www.hyperiondev.com

Introduction

Welcome to the Input and Output Task!
Overview:

Until now, the Python code you've been writing comes from one source and only goes to one place. You type it in at the keyboard and its results are displayed in the console. But what if you want to read information from a file on your computer, and/or write that information to another file? This process is called file I/O (the "I/O" stands for "input/output"), and Python has a number of built-in functions that handle this for you. In this Task, we will look at different ways of achieving this in Python.

What makes a password strong against crackers?

Using only letters or numbers in your password makes your password weak and easy to crack. It turns out that making your password stronger is just a matter of adding a few numbers and special characters. This can make a weak password like 'password' a strong password like 'P@$$word1'.

How do organisations store passwords?

Some organisations store their password as hashes. This means that they run a special one-way hash function on a client password that turns it into an encrypted form that cannot be converted back to original form. They do this for security reasons and a password can even be hashed iteratively(many times) for extra safety. Additionally organisations can add a special encrypted character(that can be variable) to each password hash to make things even harder for crackers.

Since it is impossible to go backwards to the password from the hash, a hacker must brute

force(try every possibility) password generation using the same hash function on the guessed password to generate the correct hash.

How long does it take to try every possibility for password cracking?

A weaker password obviously takes a shorter time period to crack and it depends on the computing capability i.e. using servers and parallelised password generation.

Say that a computer can generate over 1 billion password guesses a second. Let's say that you have an 8 character password that only uses lower-case letters. Doing some maths you get:

> There are 26 letters in the English language.
> The number of password possibilities are 26^8.
> If you divide this number by a billion and then by 60 to get minutes ; you get approximately 3.5 minutes to guess the password.

You can see how adding more characters and making your password longer makes it harder to crack. The table below shows the progression in time when you make your password longer and add more character possibilities. [1]

	8 characters	9 characters	10 characters	11 characters	12 characters
LC	208 seconds	90 minutes	39 hours	42 days	3 years
LC&UC	14 hours	32 days	4.5 years	238 years	12,394 years
LC&UC&DIGITS	2.5 days	5 years	26 years	1,650 years	102,304 years
LC&UC&DIGITS&SC	70 days	18 years	1,707 years	169,546 years	15,091,334 years

Websites like http://passfault.com/ give you estimates about the strength of your password and how long it could take to crack.

- **The Hyperion Team**

[1] Don't Be Cracked: The Math Behind Good Online Passwords

Instructions

First read **example.py**, open it using Notepad++ (Right click the file and select 'Edit with Notepad++').

- example.py should help you understand some simple Python. Every task will have example code to help you get started. Make sure you read all of example.py and try your best to understand.
- You may run example.py to see the output. The instructions on how to do this are inside the file. Feel free to write and run your own example code before doing the Task to become more comfortable with Python.
- You are not required to read the entirety of Additional Reading.pdf, it is purely for extra reference.

Compulsory Task

Follow these steps:

Now, create a Python file called **forgetful.py** . Imagine your friend was very forgetful and always seemed to enter his email password incorrectly. You want to write a Python program that takes all his incorrect password entries, stores it in a list, then records all his incorrect password entries in a text file called **wrongpasswords.txt**.

Example: your friends password is 'rusty'. But he enters 'rusty123', 'Rusty', 'rustless' before finally remembering his password is 'rusty' and then entering it correctly.

In this situation **wrongpasswords.txt** should read this exactly:

Incorrect password 1: rusty123
Incorrect password 2: Rusty
Incorrect password 3: rustless
Correct password entered on 4th entry.

The program should ask the user for input by saying 'Please enter your password: '. The correct password will always be 'rusty' but the user can of course enter any String.

Bonus Optional Task:

Edit your completed program so that the number of characters your friend gets wrong is also stored for each incorrect password.

Task Statistics

Last update to task: 07/01/2016.
Course Content Manager: Riaz Moola.
Task Authors: Riaz Moola and Sinead Urisohn.
Task Feedback link: Hyperion Development Feedback.