Sig•Mal

Meeting 7

Basic Terminology

Environment Variables → An environment variable is a dynamic-named value that can affect the way running processes will behave on a computer.

Credentials → Verification of Identity

Abstraction → “A general idea rather than one relating to a particular object, person, or situation.”

~ The word will arise in your mind when you need it the most to describe something

Time Complexity → how long it takes to run

Space Complexity → how much space it holds in memory

Macros → “macro instruction” : a programmable pattern which translates a certain sequence of input into a preset sequence of output.

Code Design (and some “Dark” Arts)

- Typedef
- Macros
- Credentials

Credentials on Github

```
- credentials = oauth2.SpotifyClientCredentials(client_id='6564646465asdfasdf546asdf56', client_secret='684asdfa8949werh9g4j6b')
+ credentials = oauth2.SpotifyClientCredentials(client_id='', client_secret='')
  token = credentials.get_access_token()
  token = util.prompt_for_user_token(
          username='bruh_chat_dude',
          scope='user-top-read',
-         client_id='6564646465asdfasdf546asdf56',
-         client_secret='684asdfa8949werh9g4j6b',
+         client_id='',
+         client_secret='',
          redirect_uri='http://google.com')
```

Version 1 + Version 2

Credentials out in the open (Version 1) and No credentials (Version 2)

Credentials on Github (cont)

or if you are reluctant to immortalize your app credentials in your source code, you can set environment variables like so (use `SET` instead of `export` on Windows):

```
export SPOTIPY_CLIENT_ID='your-spotify-client-id'
export SPOTIPY_CLIENT_SECRET='your-spotify-client-secret'
export SPOTIPY_REDIRECT_URI='your-app-redirect-url'
```

Version 3

Environment Variables!

Version 4

.gitignore exists

```
[redacted]ba$ cat config.py
# Fill out for API
USER=""
CLIENT_ID=""
CLIENT_SECRET=""
```

Put the config file in a ignored directory or in .gitignore and set stuff to their values.

Maybe more clout to use YAML but you decide to do that

```
credentials = oauth2.SpotifyClientCredentials(client_id=CLIENT_ID, client_secret=CLIENT_SECRET)
credentials = oauth2.SpotifyClientCredentials(client_id='', client_secret='')
token = credentials.get_access_token()
token = util.prompt_for_user_token(
        username=USER,
        scope='user-top-read',
        client_id=CLIENT_ID,
        client_secret=CLIENT_SECRET,
        client_id='',
        client_secret='',
        redirect_uri='http://google.com')
```

Environment Variables

Case against them ← (Article to proof my bias)

~ “System-wide Global Variable”

- Do for static applications, not dynamic.
- Not great for use in production. (so why do them for a project?)
- On scrubbed systems, you will have to constantly set the env variables.
- If you want to change configurations, version 4 will allow for easier editing.

While you can unset then or maybe have a virtualenv, why increase the complexity of the app / project?

Simplicity <3

Why this matters?

Spotipy

Spotipy is a lightweight Python library for the Spotify Web API. With *Spotipy* you get full access to all of the music data provided by the Spotify platform.

Assuming you set the `SPOTIPY_CLIENT_ID` and `SPOTIPY_CLIENT_SECRET` environment variables, here's a quick example of using *Spotipy* to list the names of all the albums released by the artist 'Birdy':

```
import spotipy
from spotipy.oauth2 import SpotifyClientCredentials

birdy_uri = 'spotify:artist:2WX2uTcsvV5OnS0inACecP'
spotify = spotipy.Spotify(client_credentials_manager=SpotifyClientCredentials())

results = spotify.artist_albums(birdy_uri, album_type='album')
albums = results['items']
while results['next']:
```

Why this matters?

Spotipy

user_playlist_add_tracks(*user, playlist_id, tracks, position=None*)

user_playlist_change_details(*user, playlist_id, name=None, public=None, collaborative=None, description=None*)

user_playlist_create(*user, name, public=True, collaborative=False, description=''*)

Creates a playlist for a user

Parameters:

- user - the id of the user
- name - the name of the playlist
- public - is the created playlist public
- collaborative - is the created playlist collaborative
- description - the description of the playlist

Forbidden Arts of C++

- Macro Dark Magic → Works and can compile.
- Internal OS API /* Building a Keylogger in c++ */
  - Windows → GetAsyncKeyState() → #include <winuser.h>
  - Linux → #include <linux/input.h>
  - Mac → #include <ApplicationServices/ApplicationServices.h>

Malware is normally tested in Python as an idea. Then it using a more general purpose programming language like C/C++, where they can write small yet efficient programs.

- C/C++ are popular for malware, that is not to say malware is only written in those languages. Half of Mirai is written in Golang and Emotet in VBA, so fun!

Good for PPL interested in Prog Team and Competitive Programming

More Forbidden Arts of C++

Macros → “Copy and Paste”

```
#define REP(i,a,b) for (int i = a; i <= b; i++)
```

```
for (int i = 1; i <= n; i++) {
    search(i);
}
```

```
REP(i,1,n) {
    search(i);
}
```

However, don’t think of these as lambda functions or functions. Think as copy and paste. Bc:

```
#define cube_num(x) x * x * x
```

→ cube_num(1+2) = (1 + 2 * 1 + 2 * 1 + 2) => 7 . Not 27

Good for PPL interested in Prog Team and Competitive Programming

More Forbidden Arts of C++

- Typedef

ex : long long ll → 9 chars to 2

```
typedef long long ll;
```

```
long long a = 123456789;
long long b = 987654321;
cout << a*b << "\n";
```

```
ll a = 123456789;
ll b = 987654321;
cout << a*b << "\n";
```

src: Competitive Programming Handbook

Sources

- Competitive Programmers Handbook
- Abstraction
- Dark Magic
- Spotify Docs API
- Environment Variables