# COMS W2132 Lab 3

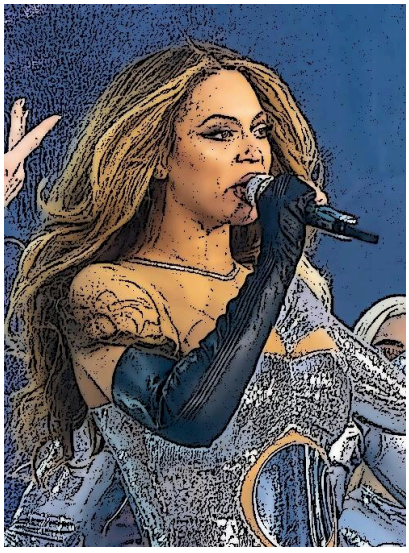
*Parts of this lab were adapted from Jan Janak's notes.*

In this lab, you'll create a basic Python program that, given the name of a celebrity (or brand, etc.), generates a cartoon image of that individual/entity.

## Example usage:

```
$ python cartoon.py
Enter name of a personality: Beyonce
Select a name from the following list:
1. Beyoncé
2. Beyoncé (album)
3. Tina Knowles
Enter the number of the desired name: 1
Cartoon image of Beyoncé saved as Beyoncé.jpeg

$ open Beyoncé.jpeg
```



To accomplish this, we will use multiple Python packages, as well as starter code that has been provided to you.

Start by opening a terminal, and go to your `cs2132` directory. Use the `cd` and `pwd` commands to do so.
Then, create a directory for lab3 using `mkdir lab3` . Then, `cd` into this directory.

There are two ways to install Python packages: system-wide, and in a virtual environment. Installing a Python package system-wide will make it available to all Python programs running on your computer. **This approach is generally discouraged because system-wide packages are difficult to manage.** We will not cover this method here.

## Creating a Virtual Environment

We will create a new folder called `myenv` that will provide a "modified" Python environment for our program. This modified environment will have the necessary packages installed. This folder is called a Python virtual environment. We can ask the Python interpreter to create it for us as follows:

```
$ python -m venv myenv
```
*You can replace "myenv" with another name if you prefer.*

Here, we call the Python interpreter with the optional argument "-m venv"3 which instructs the Python interpreter to run the venv module from the Python standard library. The last argument "myenv" is the folder name for our virtual environment. The folder must not exist.

**WARNING: Do not check virtual environment folders in Git or GitHub! Virtual environments are not meant to be managed in Git because when another developer clones your Git repository with a virtual environment in it, it will not work for them (folder names will be different, they might be running on a different platform, etc.).**

## Activating a Virtual Environment

Next, we need to activate the virtual environment. Activating a virtual environment enables it for the duration of your terminal session. In other words, running the "python" program will look for additional resources in your newly created virtual environment myenv instead of the entire system. The activation step depends on your platform.
On MacOS and Linux, you can run:

```
$ source myenv/bin/activate
(myenv) julien@Juliens-MacBook-Pro ~ $
```

On Windows:
`myenv\Scripts\activate`

## Fork and Clone the GitHub Repository

Fork the following repository to your own GitHub account:
github.com/coms2132-sp25/lab-3

Clone your fork to your local computer using `git clone [ssh url of your fork]`

## Installing Python packages inside the virtual environment

Look around the boiler plate code that has already been provided to you. There are 2 files `cartoon.py` and `images.py` . In this lab, you will only need to modify `cartoon.py` . However, you will still need to invoke prewritten functions from `images.py` so it's a good idea to check them out (for the purposes of this lab, you don't need to understand how they work, but more importantly what they do). You might notice that `images.py` imports a package called `requests`. This package can be used to connect to websites on the web (we will use it to download pictures of celebrities from Wikipedia–more on that later).

Recall the warning about not checking virtual environments into Git. So if we cannot check the virtual environment folder in Git, how do we make it possible for our collaborators or users to recreate the virtual environment so that they could run our program?

The answer lies in another file you might have also noticed in the repo called `requirements.txt` . It's a plain text file that lists all the additional packages and their versions that we should install in our virtual environment.

Currently, it contains the following:

```
certifi==2025.1.31
charset-normalizer==3.4.1
idna==3.10
requests==2.32.3
urllib3==2.3.0
```

You might recognize `requests`, among other packages (the rest are dependencies of `requests`, that is, `requests` use them in its own code). Each line also contains the version of

each package, your collaborators and users will be able to install the precise version of each package you used. If the `requests` developers release a new version of the package in the meantime with a breaking change, your users and collaborators will install a version that is known to work with your program.

It's now very easy to install all the necessary packages in your environment. All you need to do is run the following command :

```
$ pip install -r requirements.txt
```

## Looking up pictures

Now, it's your turn to code! We'll use Wikipedia to identify celebrities and find their pictures. Thankfully, someone created a python wikipedia package to make this process much easier.

You will start by completing the `prompt_for_image()` function inside `cartoon.py` . `prompt_for_image` should prompt the user for the name of a celebrity and search Wikipedia for pages matching that name (have a look at the documentation for the wikipedia.search() method). Print the top 3 results to the user and ask them to choose which one they would like to generate a cartoon for. Then you need to obtain the URL of a picture on that wikipedia page. Luckily, a function called `get_wikipedia_page_thumbnail_url()` was already implemented for you in `images.py`. Finally, `prompt_for_image` should return the URL of the image, as well as the name of the page.

Here are a few sample runs of different functions to help you understand how they work:
`wikipedia.search('Beyonce', results=3)` will return :

```
['Beyoncé', 'Beyoncé (album)', 'Tina Knowles']

images.get_wikipedia_page_thumbnail_url("Albert Einstein") will return :
'https://upload.wikimedia.org/wikipedia/commons/thumb/3/3e/Einstein_1921_by_
F_Schmutzer_-_restoration.jpg/500px-Einstein_1921_by_F_Schmutzer_-_restorati
on.jpg'

 >>> prompt_for_name()
Enter name of a personality: Einstein
Select a name from the following list:
1. Albert Einstein
2. Einstein family
```

```
3. Bob Einstein
Enter the number of the desired name: 1
('https://upload.wikimedia.org/wikipedia/commons/thumb/3/3e/Einstein_1921_by
_F_Schmutzer_-_restoration.jpg/500px-Einstein_1921_by_F_Schmutzer_-_restorat
ion.jpg', 'Albert Einstein')
```

Test your implementation so far by calling `prompt_for_image()` inside the if statement at the end of the file. Store the values returned by the function in variables, as they will be needed in the next section. For debugging, you can temporarily print them to verify that everything is producing the expected output.

## Downloading an image from the internet

Now that you have the url of the image you would like to download, use the prewritten `download_image_from_url(url, save_name)` function to save the image to disk. Here, save_name will be the name of the file (without the extension) to save the image as. The easiest is to pass the Wikipedia page name. This function will return the final filename used to save the image (spaces will have been removed and a file extension added at the end).

## Adding a cartoon effect to an image

To do so, we will use the opencv-python package. After installing, start by opening up the image inside `convert_image_to_cartoon(image_path)` using `img = cv2.imread(image_path)` .

There are multiple ways to edit a picture to make it look like a cartoon. You are free to experiment with different methods inside the `cv2` package. However, if you need a little inspiration, have a look at the code below.

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
edges = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, 9, 9)
color = cv2.bilateralFilter(img, 9, 200, 200)
cartoon = cv2.bitwise_and(color, color, mask = edges)
```

*Source: https://dev.to/ethand91/cartoon-filter-using-opencv-and-python-3nj5*

Once you have applied the necessary transformations, save the image using `cv2.imwrite(image_path, cartoon)` .

You should now have a functioning cartoon generator! Try it out to make sure everything works correctly.

## Updating requirements.txt

As you may have noticed, we have installed two new packages in the virtual environment. To reflect these changes, we need to update `requirements.txt`. Python provides a convenient command for this: try running `pip freeze`. This will generate an updated list of all packages used in this lab.

As you might recall from lab 2, the > redirection operator allows us to redirect a command's output to a file. By running:
`pip freeze > requirements.txt`
we can update `requirements.txt` accordingly.

## Committing and pushing to GitHub

Now that you are done, you can commit your changes and push them to GitHub. You do not need to commit the generated images.

## Deactivating a Virtual Environment

A virtual environment activation is valid for the duration of the terminal session. If you close the terminal window (or log out from the server), the virtual environment will be deactivated. When you open a new terminal window or SSH into your server again, you will have to repeat the activation step. If you wish to deactivate a virtual environment without closing the terminal window or logging out, run the command deactivate: `deactivate`

The command "`deactivate`" is a shell function that the activation step created internally. Running pip now will install packages system-wide.

## Deleting a Virtual Environment (optional)

Everything related to a particular virtual environment is stored in its folder. You can simply delete the folder to delete the virtual environment. Make sure that the environment is not active while you are deleting the folder. Nothing bad would happen, but running pip or other programs will not work until you deactivate it (or activate another virtual environment).