LAB1-PYTHON

OPERATING SYSTEMS-DR. LÉONARD JANER

2021-2022

JANUARY 2022

TecnoCampus
Escola Superior
Politècnica

Centre adscrit a la

Universitat
Pompeu Fabra
Barcelona

LAB1-PYTHON

Objectives

Part 1: PIP and Python Virtual Environment

Part 2: Processing Ex1 file

Part 3: Processing Ex2 file

Part 4: Processing Ex3 file

Background / Scenario

In this lab, you review Python installation, PIP, and Python virtual environments. Then you will write a Python script to extract information from some configuration and topology files into a PDF file.

Required Resources

- 1 PC with operating system of your choice
- Virtual Box or VMWare
- DEVASC Virtual Machine

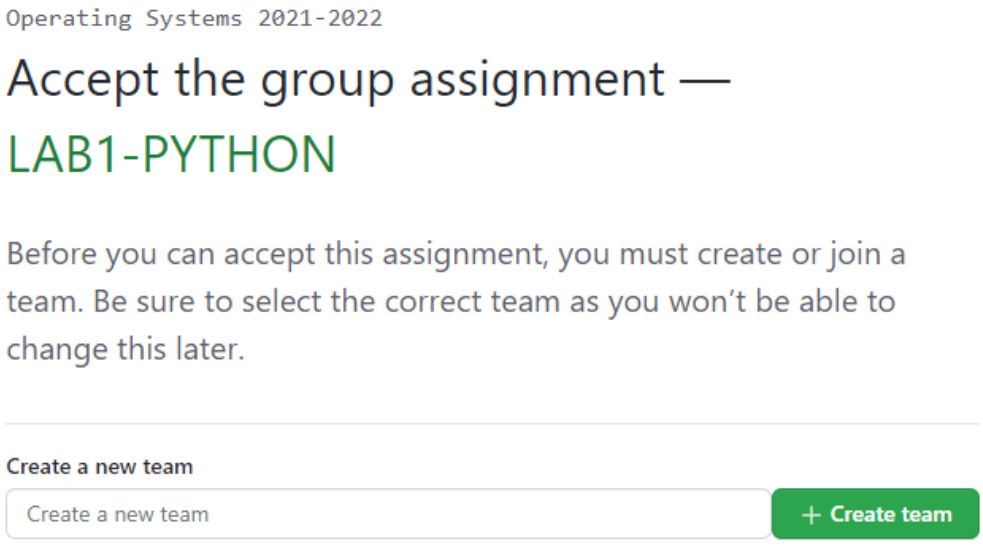
Instructions

Part 1: **Group and repository**

Using the link: https://classroom.github.com/a/TMWLu36d

Create a group with your teammate, and on that repository is where the LAB must be delivered

You have to write on the title page of your REPORT the name of the group / classroom



One of the teammates has to create the group and to other has to join the group

PIP and Python Virtual Environments

The purpose of this part is just to show you how to work with environments. When delivering the task you must provide the environment, so packages can be imported to the test environment.

PIP stands for **Pip Installs Packages**. Many people first learn about PIP and start using **pip3 install** commands on the system wide Python installation. When you run **pip3 install** command on your system, you might introduce competing dependencies in your system installation that you may or may not want for all Python projects. Therefore, the best practice is to enable a Python virtual environment. Then install only the packages that are needed for the project in that virtual environment. That way, you know exactly which packages are installed in a given setting. You can switch those package dependencies easily when switching to a new virtual environment, and not break or cause problems due to competing versions of software.

To install a Python virtual environment, use the **venv** tool in Python 3 and then activate the virtual environment, as shown in the following steps.

Step 1: Create a Python 3 virtual environment.

Inside the DEVASC VM, change to the **labs/devnet-src/python** directory. This is just an example, to show you how to create your own environment.

```
devasc@LJG:~$ cd labs/devnet-src/python/
devasc@LJG:~/labs/devnet-src/python$
```

Enter the following command to use the **venv** tool to create a Python 3 virtual environment with the name **lab1-ljg**. The **-m** switch tells Python to run the **venv** module. The name is chosen by the programmer.

```
devasc@LJG:~/labs/devnet-src/python$ python3 -m venv lab1-ljg
devasc@LJG:~/labs/devnet-src/python$
```

Step 2: Activate and test the Python 3 virtual environment.

Activate the virtual environment. The prompt changes to indicate the name of the environment you are currently working in, which is **lab1-ljg** in this example. Now when you use the **pip3 install** command form here, the system will only install packages for the active virtual environment.

```
devasc@LJG:~/labs/devnet-src/python$ source lab1-ljg/bin/activate
(lab1-ljg) devasc@LJG:~/labs/devnet-src/python$
```

Run the **pip3 freeze** command to verify that there are no additional Python packages currently installed in the **lab1-ljg** environment.

```
(lab1-ljg) devasc@LJG:~/labs/devnet-src/python$ pip3 freeze
(lab1-ljg) devasc@LJG:~/labs/devnet-src/python$
```

Now, you can install the Python **requests** package (this is just an example installing that package; you must install the packages required for your solution) within the **lab1-ljg** environment.

```
(lab1-ljg) devasc@LJG:~/labs/devnet-src/python$ pip3 install requests
Collecting requests
  Using cached requests-2.25.1-py2.py3-none-any.whl (61 kB)
Collecting certifi>=2017.4.17
  Using cached certifi-2020.12.5-py2.py3-none-any.whl (147 kB)
Collecting urllib3<1.27,>=1.21.1
  Using cached urllib3-1.26.2-py2.py3-none-any.whl (136 kB)
Collecting idna<3,>=2.5
  Using cached idna-2.10-py2.py3-none-any.whl (58 kB)
Collecting chardet<5,>=3.0.2
  Downloading chardet-4.0.0-py2.py3-none-any.whl (178 kB)
     |████████████████████████████████| 178 kB 164 kB/s
Installing collected packages: certifi, urllib3, idna, chardet, requests
Successfully installed certifi-2020.12.5 chardet-4.0.0 idna-2.10 requests-2.25.1 urllib3-1.26.2
(lab1-ljg) devasc@LJG:~/labs/devnet-src/python$
```

Re-enter the **pip3 freeze** command to see the packages now installed in the **lab1-ljg** environment.

```
(lab1-ljg) devasc@LJG:~/labs/devnet-src/python$ pip3 freeze
certifi==2020.12.5
chardet==4.0.0
idna==2.10
requests==2.25.1
urllib3==1.26.2
(lab1-ljg) devasc@LJG:~/labs/devnet-src/python$
```

To deactivate the virtual environment and go back to your system, enter the **deactivate** command.

```
(lab1-ljg) devasc@LJG:~/labs/devnet-src/python$ deactivate
devasc@LJG:~/labs/devnet-src/python$
```

Step 3: Check the current packages installed in the system environment.

Enter the system wide **python3 -m pip freeze** command to see what packages are installed in the system environment.

Note: Because Python 3 is invoked with the following command, you only use **pip** instead of **pip3**.

```
devasc@LJG:~/labs/devnet-src/python$ python3 -m pip freeze
aiohttp==3.6.2
ansible==2.9.9
apache-libcloud==2.8.0
appdirs==1.4.3
```

```
<output omitted>
```

If you want to quickly find the version of a package installed, pipe the output to the **grep** command. Enter the following to see the version of the requests package currently installed.

```
devasc@LJG:~/labs/devnet-src/python$ python3 -m pip freeze | grep requests
requests==2.22.0
requests-kerberos==0.12.0
requests-ntlm==1.1.0
requests-toolbelt==0.9.1
requests-unixsocket==0.2.0
devasc@LJG:~/labs/devnet-src/python$
```

Step 4: Sharing Your Virtual Environment

The output of the **pip3 freeze** command is in a specific format for a reason. You can use all the dependencies listed so that other people who want to work on the same project as you can get the same environment as yours.

A developer can create a requirements file, such as **requirements.txt**, by using the **pip3 freeze > requirements.txt** command. Then another developer can, from another activated virtual environment, use this **pip3 install -r requirements.txt** command to install the packages required by the project.

You must always deliver with your Python labs the **requirements.txt** file.

Re-activate the **lab1-ljg** virtual environment.

```
devasc@LJG:~/labs/devnet-src/python$ source lab1-ljg/bin/activate
(lab1-ljg) devasc@LJG:~/labs/devnet-src/python$
(lab1-ljg) devasc@LJG:~/labs/devnet-src/python$
```

Send the output of the **pip3 freeze** command to a text file called **requirements.txt**.

```
(lab1-ljg) devasc@LJG:~/labs/devnet-src/python$ pip3 freeze > requirements.txt
(lab1-ljg) devasc@LJG:~/labs/devnet-src/python$
```

Just to simulate the process, you can deactivate the **lab1-ljg** virtual environment. You can use the **ls** command to see that the **requirements.txt** file is in the **/python** directory.

```
(lab1-ljg) devasc@LJG:~/labs/devnet-src/python$ deactivate
devasc@LJG:~/labs/devnet-src/python$ ls requirements.*
requirements.txt
devasc@LJG:~/labs/devnet-src/python$ cat requirements.txt
certifi==2020.12.5
chardet==4.0.0
idna==2.10
requests==2.25.1
urllib3==1.26.2
devasc@LJG:~/labs/devnet-src/python$
```

Now, you can create and activate a new Python virtual environment called **lab1-test**.

```
devasc@LJG:~/labs/devnet-src/python$ python3 -m venv lab1-test
devasc@LJG:~/labs/devnet-src/python$ source lab1-test/bin/activate
(lab1-test) devasc@LJG:~/labs/devnet-src/python$
```

Use the **pip3 install -r requirements.txt** command to install the same packages that are installed in the **lab1-ljg** virtual environment.

```
(lab1-test) devasc@LJG:~/labs/devnet-src/python$ pip3 freeze
(lab1-test) devasc@LJG:~/labs/devnet-src/python$ pip3 install -r requirements.txt
Collecting certifi==2020.12.5
  Using cached certifi-2020.12.5-py2.py3-none-any.whl (147 kB)
Collecting chardet==4.0.0
  Using cached chardet-4.0.0-py2.py3-none-any.whl (178 kB)
Collecting idna==2.10
  Using cached idna-2.10-py2.py3-none-any.whl (58 kB)
Collecting requests==2.25.1
  Using cached requests-2.25.1-py2.py3-none-any.whl (61 kB)
Collecting urllib3==1.26.2
  Using cached urllib3-1.26.2-py2.py3-none-any.whl (136 kB)
Installing collected packages: certifi, chardet, idna, urllib3, requests
Successfully installed certifi-2020.12.5 chardet-4.0.0 idna-2.10 requests-2.25.1 urllib3-1.26.2
(lab1-test) devasc@LJG:~/labs/devnet-src/python$ pip3 freeze
certifi==2020.12.5
chardet==4.0.0
idna==2.10
requests==2.25.1
urllib3==1.26.2
(lab1-test) devasc@LJG:~/labs/devnet-src/python$
```

This will be the process to test and execute your python programs. Installing the required packages according to your **requirements.txt** file on a new environment.

When delivering your lab, you must deliver an **requirements.txt** file so when evaluating the import (and install) can be done, and the testing environment could be exactly the same as yours, with exactly the same libraries, version number, and so. If the lab works properly on your environment it must work the same way on my imported environment. This is also the way to share environment with your teammate. If the importing process fails then the labs will not be evaluated.

If you want to import/export your environment using **VS Code** you can also.

You have always to write the import process to guarantee that it is clearly written how to import your environment. If it is not written, no import will be done, and so, no evaluation will be undertaken.

Part 2: **Processing Ex1 file**

The purpose of this second part is to write a Python Script to process a file named EX1.PDF and process it

The task report must include:

- Information about how the environment has been saved by the student.
- Information about how the environment must be imported into the instructor system.
- Information about all the packages that must be used on the lab (with a brief description of the package purpose for the lab)
- Information about the structure of the program, and description of the most relevant parts of the code. It is not just a matter of coding, but you must write a good supporting documentation for your projects.
- Screenshots with the execution of the program.

Step 1: Information on input file

The input file is a kind of exam file. You have to read the content of the file, and extract some relevant information:

- The number of questions
- The number of words per question
- For the whole document, you have to create a table with the number of times each letter in the English alphabet is in the document

- The number of pages of the document
- The header of the document
- The footer of the document
- The script must be named **step1.py**, and the **requirements1.txt** file must be committed also.

Step 2: Write a new PDF file with the information re-structured

You have to write a new PDF file, named EX1-NOBREAK.PDF with the following requirements:

- The script must be named **step2.py**, and the **requirements2.txt** file must be committed also.
- The name of the file must be the name of the input file EX1.PDF and your program must add the text "-NOBREAK" to the incoming name and add the extension automatically. The file name COULD NOT be hardcoded on your program
- The script must prompt the user for the date of the exam (select the best way to ask for the information) to be written on the first page
- The content of the file, must be the same as the input file, with the same number of questions but:
    - For the first page:
        - The header must be the same, but the text "Full Student Name: " must be written below the header in a format with a box where the student can write his name. Something similar to:

Full Student Name:

        - You have to add below that a box for the QUALIFICATION of the exam, something similar to:

Qualification:

    - For the remaining pages the header must be the same without the text for student's name neither the qualification
    - The footer must be the same (but with the correct page numbering) and the date must be the one according with the date of the exam (with the same format as the input file)
    - No extra space must be added in between questions
    - The questions now must:
        - Instead of being numbered as 1, 2, 3… must be numbered Q1, Q2, Q3…
        - Instead of having the question mark (1 POINT…) at the end of the question must have it at the beginning of the question: Q1 (POINTS: 1)
        - You must have a red line as a separator between each question, and a double red line as a separator for every 5 questions (then only that line and not the previous one).
        - After the last question no red line must be written

Step 3: Write a new PDF file with the information re-structured

You have to write a new PDF file, named EX1-T1.PDF with the following requirements:

- The script must be named **step3.py**, and the **requirements3.txt** file must be committed also.
- The name of the file must be the name of the input file EX1.PDF and your program must add the text "-T1" to the incoming name and add the extension automatically. The file name COULD NOT be hardcoded on your program

- The content of the file must be similar (same requirements as the previous one) but now the questions (the ordering) must be randomized

Part 3: **Processing a group of files**

Step 4: Write a group of new PDF files with the information re-structured

You have to write a group of new PDF files, named EX1-i.PDF with the following requirements:

- The script must be named **step4.py**, and the **requirements4.txt** file must be committed also.
- The name of the file must be the name of the input file EX1.PDF and your program must add the text "i" to the incoming name and add the extension automatically. The file name COULD NOT be hardcoded on your program, where the value of "I" must be 1, 2, 3…. according to the number of files to write.
- The script must ask (in the way you select) the number of randomize questionaries to write, and generate them with the naming detailed in previous item
- The content of the file must be similar (same requirements as the previous one) but now the questions (the ordering) must be randomized

Step 5: Final Task1

The Final task1 must:

- The script must be named **step5.py**, and the **requirements5.txt** file must be committed also.
- Read all PDF files from a folder
- Extract from them the questions
- Write a new PDF file name FOLDERNAME-1-NUMBEROFFILES.PDF
- The header:
    - You have to select a new LOGO image (select the preferred way to choose the image, it can be hardcoded or prompted, …)
    - You have to select the degree (select the preferred way to choose the degree, it can be hardcoded or prompted, …)
    - You have to select the date (select the preferred way to choose the date, it can be hardcoded or prompted, …)
    - You have to write the Full Student Name and Qualification information as in the previous requirements
- The following pages header must be the same as the first page without the name and qualification information
- The footer:
    - Must have the date and numbering information (no logo)
- You have to write all the questions, from all the exams, ordered based on the length (number of words of the question)

Step 6: Final Task2

The Final task2 must generate a file with the same requirements as previous section apart from:

- The script must be named **step6.py**, and the **requirements6.txt** file must be committed also.
- Write a new PDF file name FOLDERNAME-2-NUMBEROFFILES.PDF
- Just before the content on the previous section
    - Write a title page: with your names (Full Students name of the members of the group), the date (current date), a vertical yellow line (all page long) and a

disclaimer (select the one you preferred, but must be on a file named **disclaimer.txt**)

- Write a group of pages (with an index, numbered with the number of questions, something similar to:

Q1 1

Q2 1

Q3 2

…

- Then you have to write the same content type as the Final Task 1. The purpose of this Final Task 2 is just to add the title and index pages