|  |
| --- |
| Resource monitoring in Python on Linux |
|  |
| May 30  №201219041  Authored by: Yuliy Bodurski |



## First Phase

Concept and preparation

The main goal of my project is to create a program, based on Python, that works on Linux and displays utilization of different system details in real-time. My version of Python is Python 3.9.

At first, I had to search for a function that helps me display that type of system information. I found out that a module in Python called ‘psutil’ is used to access system details and process utilities. It keeps track on various resources like CPU, disks, network and memory.

To access this library, I had to install it in my IDE (PyCharm). I made that by going to Settings -> Project: ProjectFDIBA -> Python: Interpreter and I installed the Package.

Now I can make use of all the features that this library offers. Now that I had the functions that help me display the different resources, I needed to find a way to make it in real-time. This is where the library ‘tqdm’ comes into play. ‘Tqdm’ helps me create a smart progress meter for my loops (in my case – the output of the results from ‘psutil’).

To make It work in my IDE, I had to follow the steps from the third paragraph.

# Second Phase

## Implementation

When I got into the Linux machine, the first that I had to do was to download and install the modules that I mentioned above. To do this task, I used the following commands in the Terminal:

sudo apt-get install -y python3-psutil  
sudo apt-get install -y python3-tqdm

I had to make sure that my Linux machine has read the files, so I updated the file system:

sudo apt-get update -y

Then all that I had to do was to test whether my code runs on the Unix terminal. I navigate to my destination folder and run the script command:

python3 swtchcse.py

Important part of the script is to change the value of the variable “inf=” depending on the machine we are working on. If we are working on a Windows machine, the default value should be “inf=Ethernet”. If we are working on a Unix machine – “inf=eth0”. Based on the nature of my task, I am using “eth0”.

# Third Phase

## Convert .py script to .exe application

To create an executable from my Python script, I had to install PyInstaller Package. Instead of following the steps that I had made in the First Phase, I used the Command Prompt. Using the command ‘pip3 install pyinstaller’, I managed to seal the deal. To convert my script, I had to implement the package that I’ve just installed. I navigated to the destined folder and typed ‘pyinstaller --onefile swtchcse.py’. This helped me convert my script into an executable application file.

If this method does not work, the program could be executed by typing the following commands in this order:  
 $cd Desktop ( If the .py file is placed on the Desktop )  
 $python3 swtchcse.py

P.S. To test the networking usage, open something in the browser, for instance a YouTube video or a streaming application. To change between the options, the user must hit Ctrl + C and then run the program again.