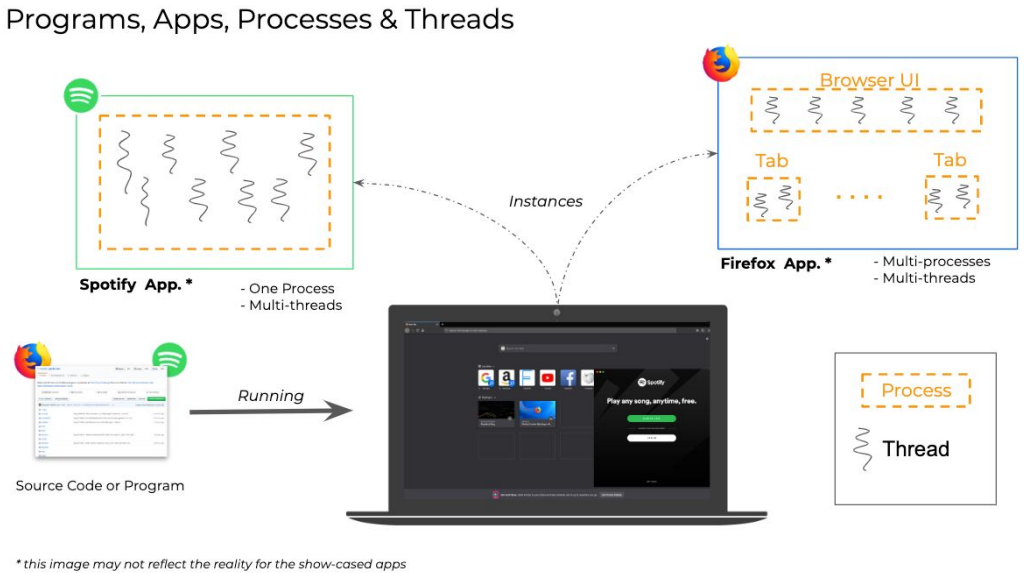
Data Science Survival Skills

Homework 10

Homework 10: Code profiling, multiprocessing and multithreading.

Hello guys! We will be using code profiling, multithreading and multiprocessing this week. Your task is to find a way to improve the speed of our programs.



Homework 10: Task 1/3

- Use code profiling to find the bottleneck.
- We provide you with the following function:

```
from skimage import data, color
from skimage.transform import resize

imgs = np.uint8(data.lfw_subset()*255)

def res_skimage(imgs):
   new_size = (imgs[1].shape[0]//2, imgs[1].shape[1]//2)
   res_im = []
   for im in imgs:
       image_resized = resize(im, new_size, anti_aliasing=True)
       res_im.append(image_resized)
   return np.asarray(res_im)
```

Homework 10: Task 1/3

- If we run "line-by-line" profiling, we will find that resize is an inefficient function.

```
%lprun -f res_skimage res_skimage(imgs) # Line-By-Line profiling
```

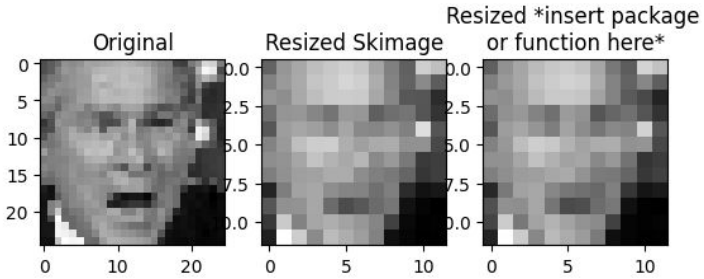
```
Timer unit: 1e-09 s

Total time: 0.134479 s
File: /tmp/ipykernel_11691/2918951301.py
Function: res_skimage at line 1

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
     1                                           def res_skimage(imgs):
     2         1       7473.0   7473.0      0.0      new_size = (imgs[1].shape[0]//2, imgs[1].shape[1]//2)
     3         1        430.0    430.0      0.0      res_im = []
     4       200     299405.0   1497.0      0.2      for im in imgs:
     5       200  133741691.0 668708.5     99.5          image_resized = resize(im, new_size, anti_aliasing=True)
     6       200     311429.0   1557.1      0.2          res_im.append(image_resized)
     7         1     118820.0 118820.0      0.1      return np.asarray(res_im)
```

Homework 10: Task 1/3

- Your task is to find a way to improve the speed of the code.

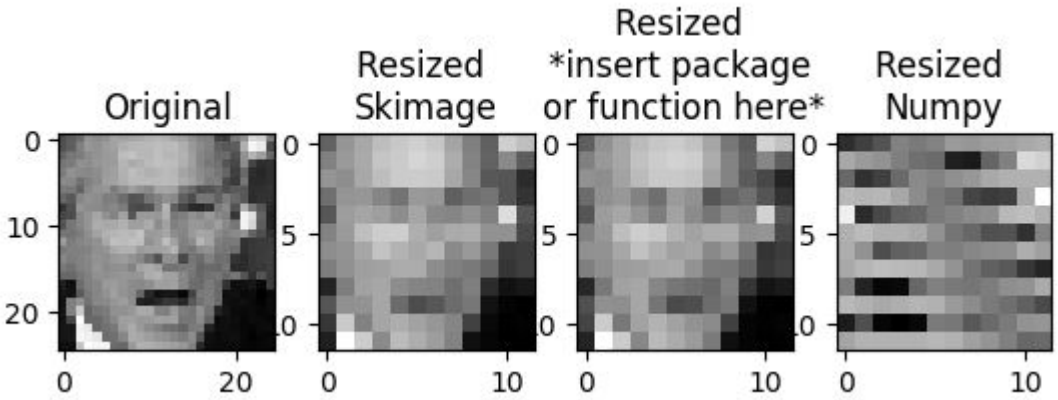
➜ **Slide: Screenshot of your results (line-by-line profiling only with the faster function)**

➜ **Slide: Show the resized images with both scikit-image and your solution.**

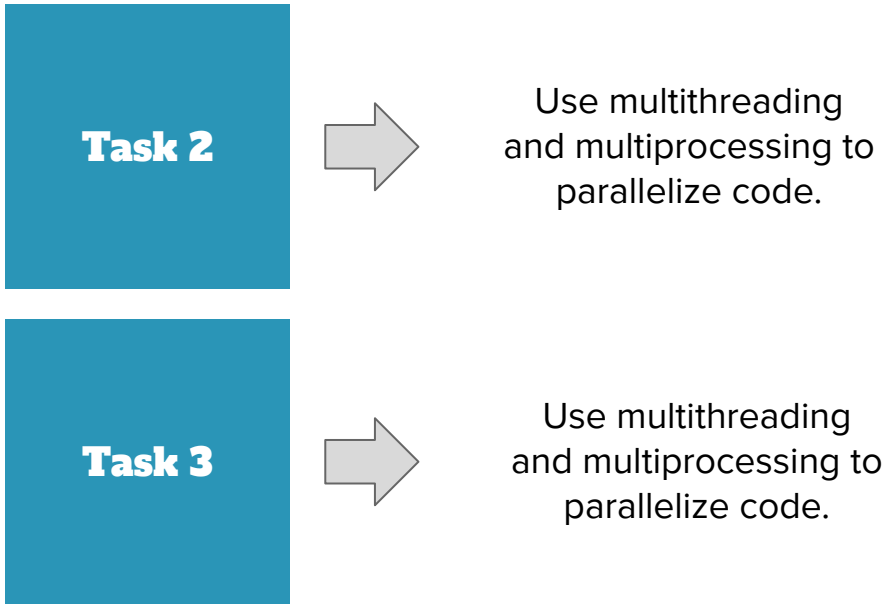


Homework 10: Task 1/3

Note: Even though a transformation using NumPy is faster, the result of resize() does not look adequate. NumPy is then not suitable for this purpose. Such solutions will be marked as **failed**.



Homework 10: Multiprocessing and multithreading



Homework 10: Task 2/3

We will provide you with the following function. Your task is to approximate pi the fastest way possible using MT and MP for different values of the argument n.

```
## Task 2
def approximate_pi(n):
    pi_2 = 1
    nom, den = 2.0, 1.0
    for i in range(n):
        pi_2 *= nom / den
        if i % 2:
            nom += 2
        else:
            den += 2
    return 2*pi_2

# Data to pass to the above function
nums = [1822725,
        22059421,
        32374695,
        88754320,
        9716266]
```

Homework 10: Task 3/3

We will provide you with some Numpy files. Your task is to load them the fastest way possible using MT and MP. You can download them from here.

```
## Use this function for task 3
def load_array(filename):
   return np.load(filename)

# Load all files
```

Homework 10: Tasks 2-3/3

- Use multithreading **and** multiprocessing for tasks 2 **and** 3. Check the performance (Hint: the *magic* command *%%time* might come in handy).

➜ **Slide: Screenshot of your results (time spent using each method)**

➜ **Slide: Code snippet of MT and MP for each function**

➜ **Slide: Answer for tasks 2 and 3: Why is MT or MP faster for this tasks? Please write a well-argued answer. Simple or false justifications will be graded as failed. A couple of sentences should do it.**

Homework 10: Example

Tasks 2 and 3:

```
## Task 2
# Multiprocessing
YOUR CODE


#Multithreading
YOUR CODE
```

```
## Task 3
# Multiprocessing
YOUR CODE


#Multithreading
YOUR CODE
```

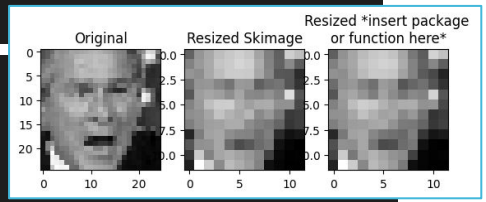
The code for task 2 benefits from Multi... because of this reason. It runs 60% faster than using Multi...

The code for task 3 benefits from Multi... because of this reason. It runs 60% faster than using Multi...

Task 1:
I used the function X that is 90% faster than skimage:

```
# Line-By-Line profiling
%lprun -f your_function your_function(imgs)
```

```
Total time: 0.134479 s
File: /tmp/ipykernel_11691/2918951301.py
Function: res_skimage at line 1

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
     1                                           def res_skimage(imgs):
     2         1       7473.0   7473.0      0.0      new_size = (imgs[1].shape[0]//2, imgs[1].shape[1]//2)
     3         1        430.0    430.0      0.0      res_im = []
     4       200     299405.0   1497.0      0.2      for im in imgs:
     5       200  133741691.0 668708.5     99.5          image_resized = resize(im, new_size, anti_aliasing=True)
     6       200     311429.0   1557.1      0.2          res_im.append(image_resized)
     7         1     118820.0 118820.0      0.1      return np.asarray(res_im)
```



Homework: Requirements

You must complete **all** homework assignments (**unless otherwise specified**) following these guidelines:

- **One** slide/page.
- **PDF** file format only.
- It has to contain your **name** and **student** (**matriculation**) **number** in the down-left corner.
- Font: **Arial,** Font-size: **> 10 Pt.**
- Answer **all** the questions and solve all the tasks requested.
- Be careful with **plagiarism**. Repeated solutions will not be accepted!