1. nofluffjobs.com is job offer portal that focuses on IT jobs in Poland.

2. I chosen interactive method of interacting with page as I wanted to use in site search form. By doing so, I was able to search for offers with custom parameters such as Technology, Location, Category and More (where user can choose seniority level and other benefits/perks of offers). To do that I used selenium with chromium driver.
After getting urls of given search, I used static method to scrap offer site.

3. Module get_profile_urls.py uses
Scraping starts with getting urls of offers, module get_profile_urls.py can be used to get them. It works by reading file profile.txt and getting from it parameters that will be used to fill search form. profile.txt can be modify to create new profiles for scraping (for everything to work properly use similar structure as other profiles (and the same keys), please do not use underscore in name of profile, all values are case sensitive).
User can select profile by changing variable PROFILE_NAME at beginning of get_profile_urls.py, for this projects requirements profile "python job in warsaw" have more than 100 offers (variable MAX_100_PAGES if true will limit getting more that 100 urls).
By running this module, program will select appropriate parameters in search form and get urls of displayed offers from all pages. After that, urls will be saved in folder urls where can be further utilized by module scrap_offers.py.

To make use of newly created urls file and to scrap offers we run module scrap_offers.py, by changing value of variable URLS_FILE_NAME we can determine from which urls file is going to start scraping offers (if value is equal None program will use the newest urls file). If selected urls file contain more than 100 items, we can limit scraping more than 100 sited by changing MAX_100_PAGES to true.
Running module scrap_offers.py will start scraping each urls and append the result of individual scrap to newly created cvs file.

Three other modules was created to structure the code.
class_items.py – defines all scrapy items to which scraper will assign scraped items of offer,
column_names – collection of all column names to which data can be fitted,
xpath_list – all xpath that will be used for scrapy,

4.Output of both modules is saved in two seperate folders:
urls – in this folder each file consist of list of urls, seperated with new line, format – plain text.

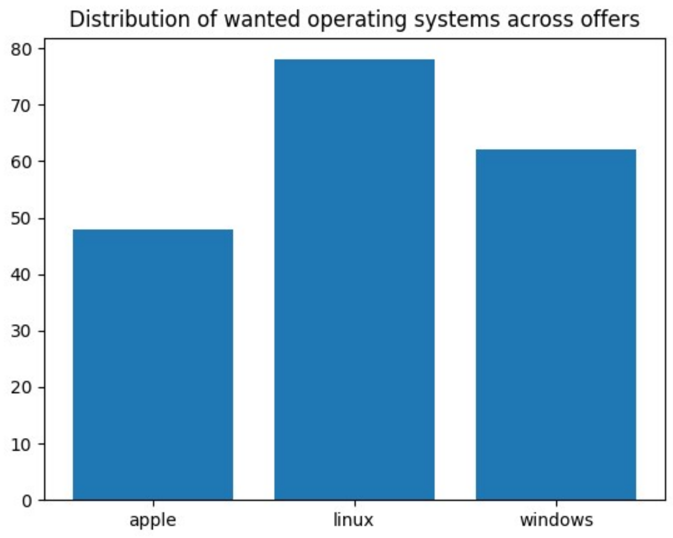
offers – in this folder each file consist of job offers data, columns separated with semicolon.
Columns are loosely categorized into few categories (list of all column and their category in file column_names.py): general, work_methodology, equipment_supplied_category, specs_category.
This categorization was created as offer site is divided in this way.

Some columns can hold multiple elements within (separated with comma). it was done to save space and maintain readability of already crowded table, as for example offer can have up to 18 must_have positions.

Columns that hold aggregated values are: must_have, nice_to_have, perks_in_the_office, benefits, job_location, integration_tests, all in category work_methodology and operating system

Not scraped elements are display is csv as empty space.

5.Data analysis for profile ‘python job in warsaw’ (script available in file analysis.py):



Mean salary across seniority (only B2B salary offers, in PLN)
Senior : 17362.27
Mid : 12743.16
Expert : 17000.0
Junior : 7875.0

Created by:
Piotr Świercz 372756