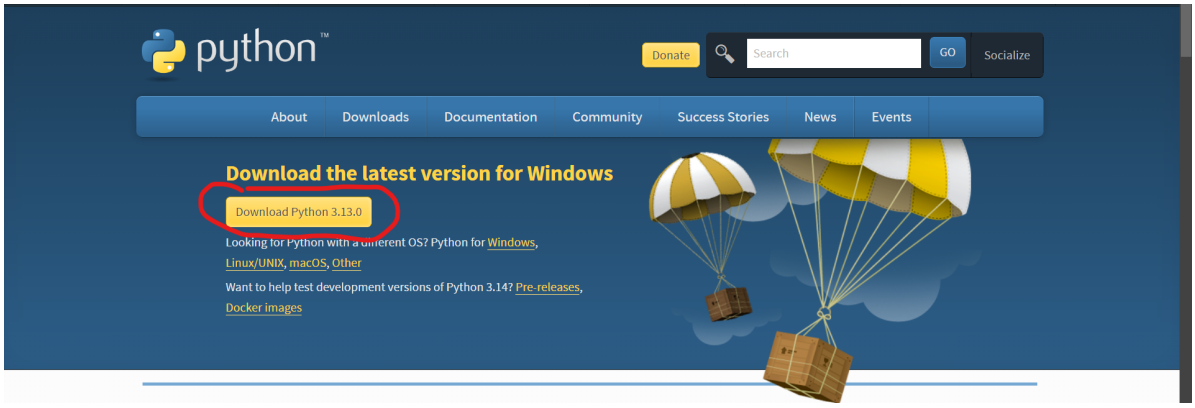
BootCamp ATLAS #1

Author : Mateusz Walo & Dominila Stępniewska

Wstęp do Pythona ze specjalizacją w uczeniu maszynowym

Instalacja Pythona

Należy wejść na stronę internetową https://www.python.org/downloads/ i wybrać wersję odpowiednią dla Twojego systemu operacyjnego i wersję która Ci odpowiada. Załóżmy że interesuje nas `Python 3.13.0` i korzystamy z systemu operacyjnego Windows



Po pobraniu pliku przechodzi przez każdy elementn instalki zostawiając basic ustawienia.

Aby sprawdzić czy Python zainstalował się poprawnie, otwórzmy wiersz poleceń (`Win+R` + `cmd`)

i wpiszmy następującą komendę

`python --version` alternatywnie `py --version` zależnie od wersji z jakiej korzystamy, jeśli korzystamy z Pythona3 `python3 --version` analogicznie

```
C:\Users\Mateusz Walo>py --version
Python 3.10.9
```

Jupyter Lab

Aby w pełni wykorzystać potencjał Pythona w branżach Data Science i Machine Learning potrzebujemy frameworku z którego będziemy korzystać w celu generowania plików `.ipynb`. Na sam początek sugeruję aby tym frameworkiem był `Jupyter Lab` aby go zainstalować należy otworzyć ponownie wiersz poleceń i wpisać polecenie `pip install jupyter lab` w przypadku gdy nie zadziała ta komenda i masz inną wersję Pythona wpisz `py -m pip install jupyter lab`. Wpisujemy polecenie i klikamy `Enter` jeśli posiadamy połączenie z siecią powinno się rozpocząć pobieranie które będzie wyglądać jak cyferki z matrixa . Jak zakończy się instalacja należy otworzyć naszego Jupytera poleceniem w terminalu `jupyter lab` ewentualnie `py -m jupyter lab` zależy z której opcji instalowania korzystaliśmy. Powinno to wyglądać nasstępująco.

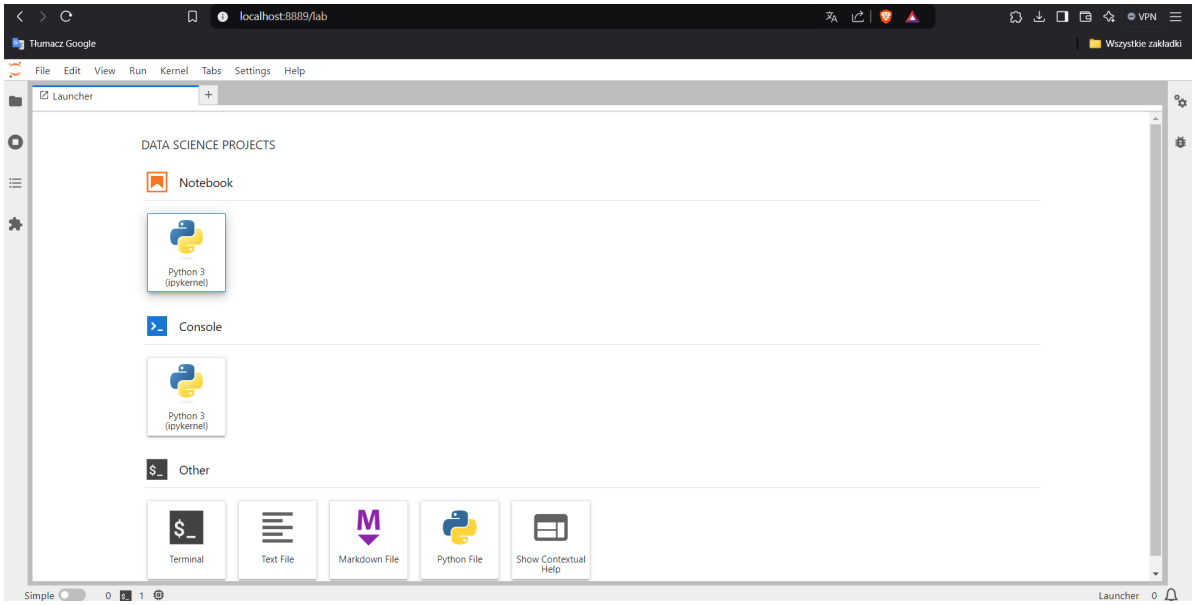
```
C:\Users\Mateusz Walo>py -m jupyter lab
[I 2024-10-28 07:30:02.025 ServerApp] jupyter_lsp | extension was successfully linked.
[I 2024-10-28 07:30:02.031 ServerApp] jupyter_server_terminals | extension was successfully linked.
[I 2024-10-28 07:30:02.037 ServerApp] jupyterlab | extension was successfully linked.
[I 2024-10-28 07:30:02.052 ServerApp] notebook | extension was successfully linked.
[I 2024-10-28 07:30:02.354 ServerApp] notebook_shim | extension was successfully linked.
[I 2024-10-28 07:30:02.384 ServerApp] notebook_shim | extension was successfully loaded.
[I 2024-10-28 07:30:02.400 ServerApp] jupyter_lsp | extension was successfully loaded.
[I 2024-10-28 07:30:02.400 ServerApp] jupyter_server_terminals | extension was successfully loaded.
[I 2024-10-28 07:30:02.405 LabApp] JupyterLab extension loaded from C:\Users\Mateusz Walo\AppData\Local\Programs\Python\Python310\lib\site-packages\jupyterl
ab
[I 2024-10-28 07:30:02.405 LabApp] JupyterLab application directory is C:\Users\Mateusz Walo\AppData\Local\Programs\Python\Python310\share\jupyter\lab
[I 2024-10-28 07:30:02.405 LabApp] Extension Manager is 'pypi'.
[I 2024-10-28 07:30:02.416 ServerApp] jupyterlab | extension was successfully loaded.
[I 2024-10-28 07:30:02.426 ServerApp] notebook | extension was successfully loaded.
[I 2024-10-28 07:30:02.426 ServerApp] The port 8888 is already in use, trying another port.
[I 2024-10-28 07:30:02.432 ServerApp] Serving notebooks from local directory: C:\Users\Mateusz Walo
[I 2024-10-28 07:30:02.432 ServerApp] Jupyter Server 2.12.5 is running at:
[I 2024-10-28 07:30:02.432 ServerApp] http://localhost:8889/lab?token=b10831651983d22c2860da61e0f2a567f442fc77f1e33e91
[I 2024-10-28 07:30:02.432 ServerApp]     http://127.0.0.1:8889/lab?token=b10831651983d22c2860da61e0f2a567f442fc77f1e33e91
[I 2024-10-28 07:30:02.432 ServerApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 2024-10-28 07:30:02.464 ServerApp]

    To access the server, open this file in a browser:
        file:///C:/Users/Mateusz%20Walo/AppData/Roaming/jupyter/runtime/jpserver-1932-open.html
    Or copy and paste one of these URLs:
        http://localhost:8889/lab?token=b10831651983d22c2860da61e0f2a567f442fc77f1e33e91
        http://127.0.0.1:8889/lab?token=b10831651983d22c2860da61e0f2a567f442fc77f1e33e91
[I 2024-10-28 07:30:02.538 ServerApp] Skipped non-installed server(s): bash-language-server, dockerfile-language-server-nodejs, javascript-typescript-langse
rver, jedi-language-server, julia-language-server, pyright, python-language-server, r-languageserver, sql-language-server, texlab, typescript-language-serve
r, unified-language-server, vscode-css-languageserver-bin, vscode-html-languageserver-bin, vscode-json-languageserver-bin, yaml-language-server
[W 2024-10-28 07:30:04.711 LabApp] Could not determine jupyterlab build status without nodejs
[I 2024-10-28 07:30:05.912 ServerApp] Kernel started: 70a69109-1be8-41ce-8c35-ddeea71d7347
[I 2024-10-28 07:30:07.007 ServerApp] Connecting to kernel 70a69109-1be8-41ce-8c35-ddeea71d7347.
```

Po wykonaniu się polecenia na lokalnym porcie powinien się nam otworzyć Jupyter Lab. Przeważnie defaultowo jest to port 8888. (U mnie ze względu na ustawienia Dockerowe jest to 8889 bo port 8888 zajmuje mi jeden notatnik z innego kontenera)

Jeśli widzisz na swoim monitorze to co powyżej to Gratuluję! Zainstalowałeś i uruchomiłeś Jupyter Lab.

Wstęp do bibliotek w Pythonie

Już macie doświadczenie w instalacji czegoś za pomocą metody `pip` więc teraz przez analogię do instalacji Jupyter Lab instalujemy biblioteki Pandas i Numpy przez komende `pip install`



`pandas numpy` albo `py -m pip install pandas numpy` możemy robić to w wierszu poleceń albo wykorzystać do tego Jupter Lab, wpisując te same polecenie tylko z wykrzyknikiem `!` przed poleceniem w nowym chunku kodowym i klikając `Ctrl+Enter`. Dzięki tej metodzie możemy wykonywać polecenia bashowe z poziomu Pythona, jest to wygodne rozwiązanie jeśli pracujemy lokalnie

Biblioteki `Pandas` i `Numpy` odowiadają za podstawowe manipulacje macierzowe, operacje na ramkach danych, arytmetyke i algebrę macierzy, posiadają przydatne funckje statystyczne które przydają się w późniejszych etapach pracy.

Dla dociekliwych:

W przyszłości zapoznamy się z biblioteką ***`sckit-learn`*** *dzięki której będizemy mogli tworzyć klasyczne modele uczenia maszynowego, omówieniem tej biblioteki zajmiemy się na* ***`BootCampie #2`*** *lecz zainstalować możecie ją już teraz lokalnie za pomocą metody* ***`pip`****.*

Opcjonalnie dla dociekliwych:

Jeśli posiadasz więcej czasu i wolnego interentu możesz zaintslować w lokalnym środowisku również bibliotekę ***`tensorflow`*** *która posłuży nam na* ***`BootCamp#3`*** *do toworzenia modeli uczenia głębokiego - sieci neuronowych, dzięki swojemu API* ***`Keras`***

Dla super dociekliwych:

W przyszłości możecie spotkać się z biblioteką ***`Polars`*** *(* https://pola.rs *)która pełni analogiczne funkcje jak* ***`Pandas`*** *i umożliwia na szybszą pracę na większych zbiorach danych, dociekliwi mogą zpaoznać się z nią i dowiedzieć się czemu jest w stanie szybciej wczytywać pliki i operacje na dużych ramkach wykonuje szybciej*

Zacznijmy pracę nad danymi w Pythonie!

```
#Importowanie bibliotek w Python z wykorzystaniem aliasów

import pandas as pd
import numpy as np
```

Tworzenie listy numerycznej typu np. `int`

```
numbers = range(1,100,5)

#Funckja range() od kąd zaczynamy, do ile, co ile
```

```
#Funkcja series tworzy nam szereg z listy numbers

pd.Series(numbers)
```

```
0      1
1      6
2     11
3     16
4     21
5     26
6     31
7     36
8     41
9     46
10    51
11    56
12    61
13    66
14    71
15    76
16    81
17    86
18    91
19    96
dtype: int64
```

Tworzenie listy wyrażeń typu `string`

```
#Analogicznie możemy postępować z napisami ;)

strings = "Koło" , "ATLAS", "to", "najlepsze", "koło","na","Pollub"
pd.Series(strings)
```

```
0         Koło
1        ATLAS
2           to
3    najlepsze
4         koło
5           na
6       Pollub
dtype: object
```

```
#oraz z listami zawierającymi dowolne elementy ;) róznych typów

abstract_list= "ATLAS" , 12, 96.69, -3.345, "Pollub", 12.432
pd.Series(abstract_list)
```

```
0     ATLAS
1        12
2     96.69
3    -3.345
4    Pollub
5    12.432
dtype: object
```

Zaraz przed wami pierwsze kolokwia więc zróbmy tabele obrazującą wasze możliwe przyszłe wyniki i na tej podstawie się czegoś nauczmy

```
procenty_z_kolosa = [93,50,73,98]
przedmioty = ['Funckje elementarne',
'Logika','Analiza matematyczna',
'Algebra liniowa']
pd.Series(procenty_z_kolosa, index=przedmioty) #pamietaj o przecinku tutaj ;)
```

```
Funckje elementarne     93
Logika                  50
Analiza matematyczna    73
```

```
Algebra liniowa         98
dtype: int64
```

Alternatywny sposób

```
#Sposób #2

data ={'Funckje elementarne':93,
'Logika':50,
'Analiza matematyczna':73,
'Algebra liniowa':98}
#wstawiamy tutaj nawiasy klamrowe
#jesli robimy w ten sposob a nie [] pamietaj !
pd.Series(data)
```

```
Funckje elementarne     93
Logika                  50
Analiza matematyczna    73
Algebra liniowa         98
dtype: int64
```

Zapomnieliśmy o wstępie do programowania i co teraz?

```
przedmioty = ['Funckje elementarne',
'Logika',
'Analiza matematyczna',
'Wstęp do programowania',
'Algebra liniowa']
wyniki =pd.Series(data, index=przedmioty)
wyniki
```

```
Funckje elementarne       93.0
Logika                    50.0
Analiza matematyczna      73.0
Wstęp do programowania     NaN
Algebra liniowa           98.0
dtype: float64
```

Pokazuje nam `NaN` jako że nie zedfiniowalismy tej danej w `data`, skrót `NaN` oznacza `Not a number` i często wskazuje nam na brak danych.

```
#Funkcja isnull()  pozwala nam na sprawdzenie braków danych w ramce danych

wyniki.isnull()
```

```
Funckje elementarne       False
Logika                    False
Analiza matematyczna      False
Wstęp do programowania     True
Algebra liniowa           False
dtype: bool
```

Wasz kolega się pyta was o to z jakis kolosów macie piątkę i chcecie użyć do tego Pythona?

Żaden problem, możemy przeszukiwać potrzebne nam rekordy w tabeli za pomocą operatorów logicznych

```
wyniki[wyniki>90]
```

```
Funckje elementarne    93.0
Algebra liniowa        98.0
dtype: float64
```

Dobra zdaliście już kolokwium z wstępi do programowania i chcecie zastąpić `NaN` waszym wynikiem, oto jak to zrobić :)

```
wyniki['Wstęp do programowania']=78
wyniki
```

```
Funckje elementarne       93.0
Logika                    50.0
Analiza matematyczna      73.0
Wstęp do programowania    78.0
Algebra liniowa           98.0
dtype: float64
```

Kolega się was pyta czy to prawda że z funkcji elementarnych macie 50%

Nic prostszego :D

```
wyniki['Funckje elementarne']==50
```

```
False
```

Sortowanie danych

```
values=pd.Series([-12,534.423,
53,28,np.nan, 35,9,43,np.nan])
values
```

```
0    -12.000
1    534.423
2     53.000
3     28.000
4        NaN
5     35.000
6      9.000
7     43.000
8        NaN
dtype: float64
```

```
#Sortowanie rosnąco

values.sort_values(ascending = True)
```

```
0    -12.000
6      9.000
3     28.000
5     35.000
7     43.000
2     53.000
1    534.423
4        NaN
8        NaN
dtype: float64
```

```
#Sortowanie malejąco

values.sort_values(ascending = False)
```

```
1    534.423
2     53.000
7     43.000
```

```
5     35.000
3     28.000
6      9.000
0    -12.000
4        NaN
8        NaN
dtype: float64
```

Uwaga: *Zauważ że wartości* `NaN` *wędrują na sam koniec, ponieważ nie są w żaden sposób określone do porównywania*

```
#Sortowanie list stringów

string_values=pd.Series(['q2','1','ATLAS',
'BajoJajo','LLM','AULA numer 1',
  'Ja ci dam bajojajo...'])
string_values
```

```
0                      q2
1                       1
2                   ATLAS
3                BajoJajo
4                     LLM
5            AULA numer 1
6    Ja ci dam bajojajo...
dtype: object
```

```
#Analogicznie, posługując się kolejnością alfabetyczną

string_values.sort_values(ascending = True)
```

```
1                       1
2                   ATLAS
5            AULA numer 1
3                BajoJajo
6    Ja ci dam bajojajo...
4                     LLM
0                      q2
dtype: object
```

```
string_values.sort_values(ascending = False)
```

```
0                       q2
4                      LLM
6    Ja ci dam bajojajo...
3                 BajoJajo
5             AULA numer 1
2                    ATLAS
1                        1
dtype: object
```

Ramki danych

Najwięcej czasu w swojej karierze studenckiej spędzicie na danych właśnie w formacie ramek danych w postaci np. danych typu csv, xlsx etc.

```
#Wracając do 1 semestru tworzymy ramkę danych z naszymi ocenami ;)

data = {'Przedmioty':
  ['Algebra liniowa',
'Analiza matematyczna',
'Geometria analityczna',
'Wstęp do programowania',
'Funkcje elementarne',
'Logika'],
        'Punkty z kolokwium' :
          [73,78,69,70,93,52],
        'Oceny':
          [4.0,4.0,3.5,4.0,5.0,3.0]}
df=pd.DataFrame(data)
#Funkcja pandas.DataFrame() tworzy nam ramkę danych, powołując się na poprzednie sposoby t
df
```

	Przedmioty	Punkty z kolokwium	Oceny
0	Algebra liniowa	73	4.0
1	Analiza matematyczna	78	4.0
2	Geometria analityczna	69	3.5
3	Wstęp do programowania	70	4.0
4	Funkcje elementarne	93	5.0
5	Logika	52	3.0

Ramki danych są najbardziej przyjazną formą danych na sam początek dlatego na niej skupimy się najbardziej, właśnie stowrzyliśmy własną mini ramkę, i mamy ogólny pogląd jak będą one wyglądać.

Teraz przejdźmy do prawdziwych danych

```
#Za pomocą funkcji pandas.read_csv()
#wczytujemy dane tabelaryczne do ramki danych

df=pd.read_csv("healthcare-dataset-stroke-data.csv")
```

```
#Funckja head() wywołuje pierwszych 5 obserwacji
#analogicznie tail() wywołuje 5 ostatnich
#a sample() wywołuje 5 losowych obserwacji
#funkcja info() powzala na wyswietlenie informacji o zbiorze danych
#Funckja columns() pokazuje kolumny

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 5110 non-null   int64
 1   gender             5110 non-null   object
 2   age                5110 non-null   float64
 3   hypertension       5110 non-null   int64
 4   heart_disease      5110 non-null   int64
 5   ever_married       5110 non-null   object
 6   work_type          5110 non-null   object
 7   Residence_type     5110 non-null   object
 8   avg_glucose_level  5110 non-null   float64
 9   bmi                4909 non-null   float64
 10  smoking_status     5110 non-null   object
 11  stroke             5110 non-null   int64
dtypes: float64(3), int64(4), object(5)
memory usage: 479.2+ KB
```

```
#.T robi nam transpozycje, w cely wizulanym bo renderuje w pdf
```

```
df.head().T
```

	0	1	2	3	4
id	9046	51676	31112	60182	1665
gender	Male	Female	Male	Female	Female
age	67.0	61.0	80.0	49.0	79.0
hypertension	0	0	0	0	1
heart_disease	1	0	1	0	0
ever_married	Yes	Yes	Yes	Yes	Yes
work_type	Private	Self-employed	Private	Private	Self-employed
Residence_type	Urban	Rural	Rural	Urban	Rural
avg_glucose_level	228.69	202.21	105.92	171.23	174.12
bmi	36.6	NaN	32.5	34.4	24.0
smoking_status	formerly smoked	never smoked	never smoked	smokes	never smoked
stroke	1	1	1	1	1

```
#pokazuje statystyki opisowe
df.describe()
```

	id	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	4909.000000	5110.
mean	36517.829354	43.226614	0.097456	0.054012	106.147677	28.893237	0.048
std	21161.721625	22.612647	0.296607	0.226063	45.283560	7.854067	0.215
min	67.000000	0.080000	0.000000	0.000000	55.120000	10.300000	0.000
25%	17741.250000	25.000000	0.000000	0.000000	77.245000	23.500000	0.000
50%	36932.000000	45.000000	0.000000	0.000000	91.885000	28.100000	0.000
75%	54682.000000	61.000000	0.000000	0.000000	114.090000	33.100000	0.000
max	72940.000000	82.000000	1.000000	1.000000	271.740000	97.600000	1.000

Widzimy jak prezentuję się bardziej okazały zbiór niż oceny z kolokwiów, obserwacji i kolumn jest nieco więcej ;)

```
#Zwraca informację o typach danych

df.dtypes
```

```
id                   int64
gender              object
```

```
age                 float64
hypertension          int64
heart_disease         int64
ever_married         object
work_type            object
Residence_type       object
avg_glucose_level   float64
bmi                 float64
smoking_status       object
stroke                int64
dtype: object
```

```
#Zwraca informacje o kształcie ramki danych

df.shape
```

```
(5110, 12)
```

Manipulacja danymi

Wyobraź sobie taką sytacje dzwoni do Ciebie szef i mówi że zapomniał wysłać Ci dane o jednej dodatkowej kolumny o stosunku `avg_glucose_level` do `bmi` i to do kwadratu!

Jak sobie z tym poradzić?

Zadzwonić do szefa i powiedzieć mu że jest debilem że tego nie wysłał i stracić pracę :)

ALBO

Dodać tą kolumnę samemu dzięki bibliotecw `Pandas`

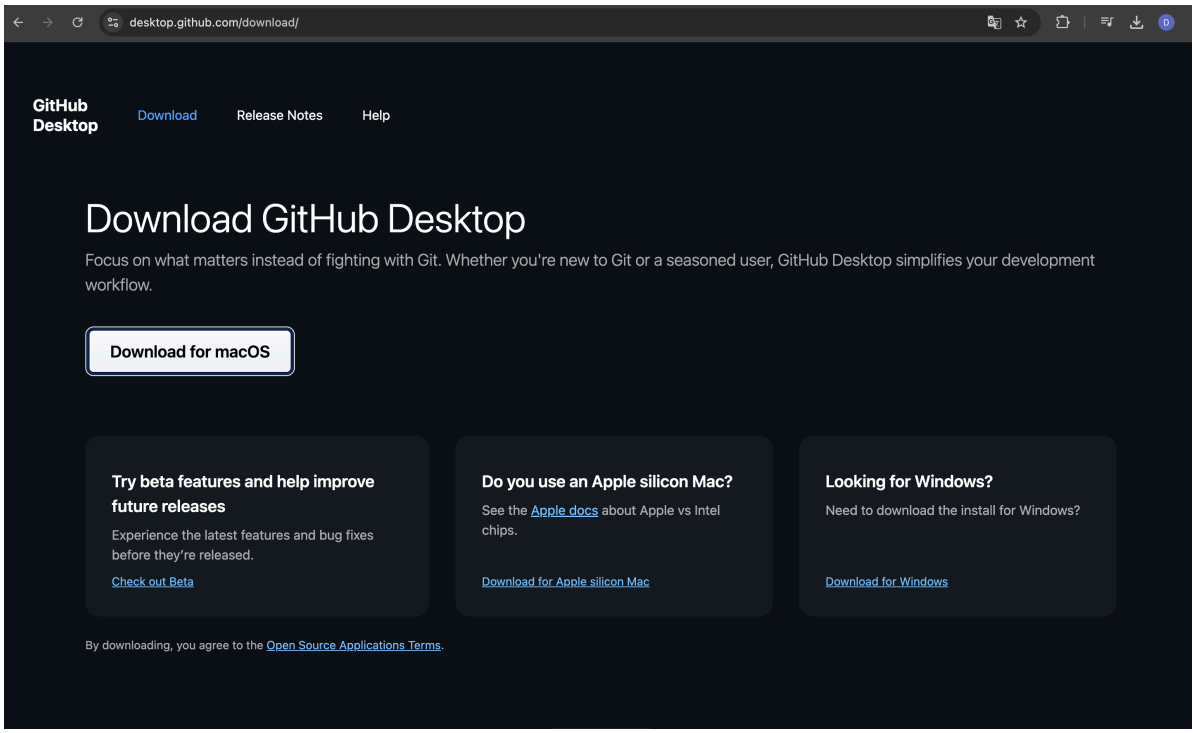
```
df["Nowa_kolumna"]=(df["avg_glucose_level"]/df["bmi"])**2
```

```
df["Nowa_kolumna"].sample(5)
```

```
2110    24.268587
2530    19.242271
4242     9.724594
248     16.167785
3788    64.320400
Name: Nowa_kolumna, dtype: float64
```

Wstęp do GitHuba

Aby pobrać GitHub Desktop należy wejść na stronę https://desktop.github.com/download/ i wybrać wersję odpowiednią dla Twojego systemu operacyjnego.

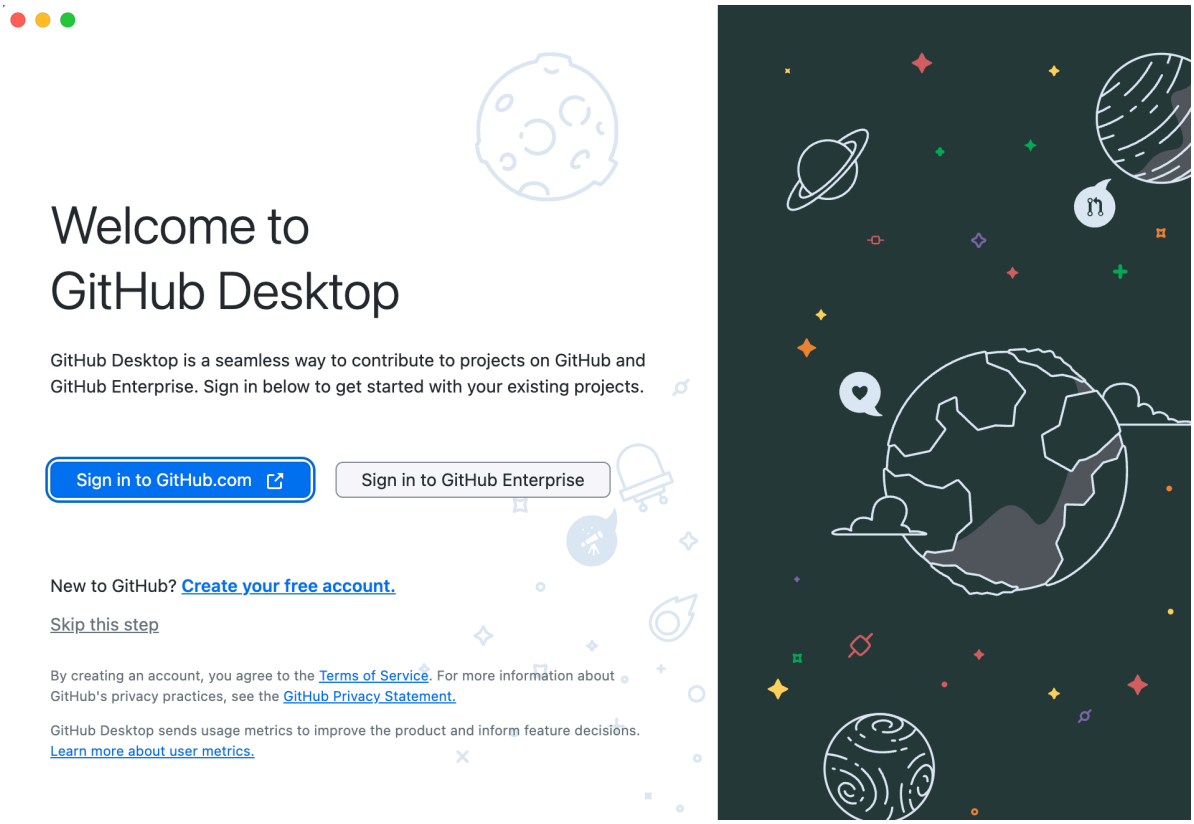


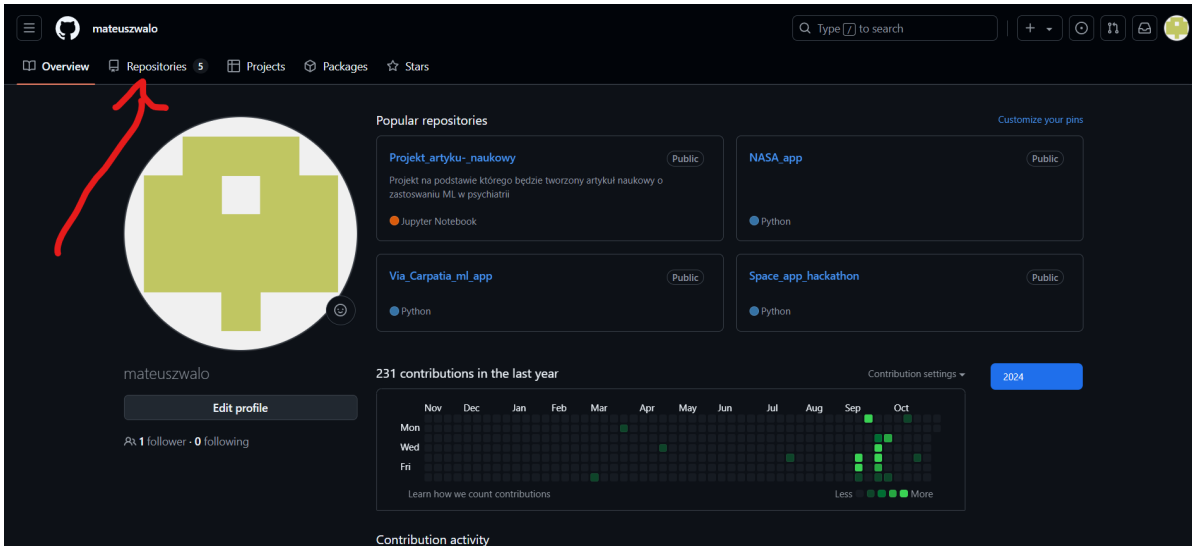
Po pobraniu, instalujesz GitHub Desktop i otwierasz aplikację. Jeśli masz konto na GitHub, logujesz się na nie, jeśli nie, tworzysz nowe konto.

Po połączeniu kluczy SSH z aplikacją możemy publikować na nim w repozytoriach własne projekty :D

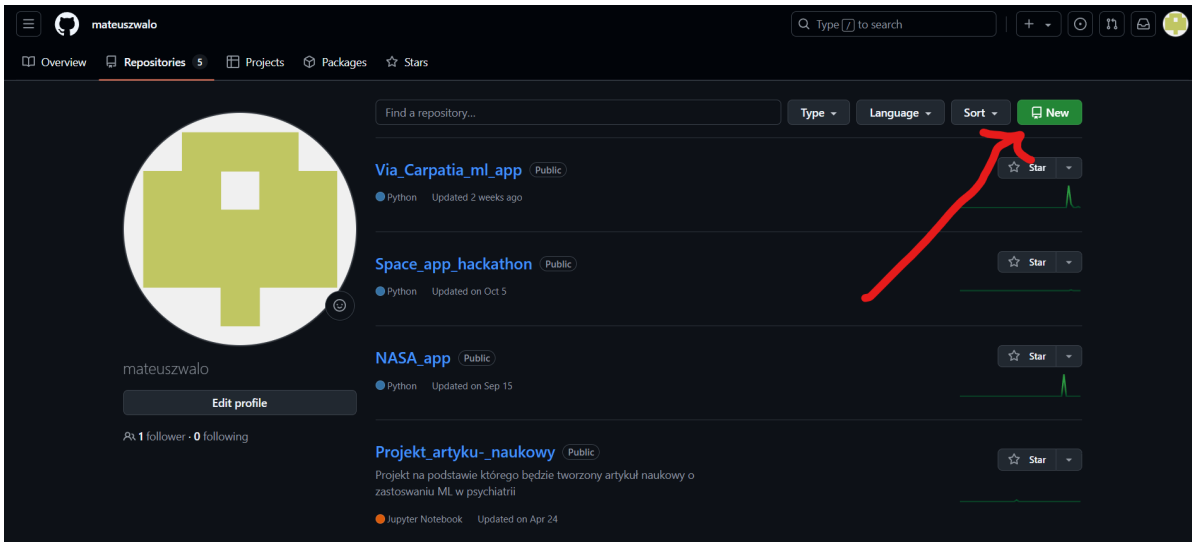
Instrukcja do połączenia kluczy: https://docs.github.com/en/enterprise-cloud@latest/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account

Jak stworzyć nowe repozytorium?


Welcome to
GitHub Desktop
GitHub Desktop is a seamless way to contribute to projects on GitHub and GitHub Enterprise. Sign in below to get started with your existing projects.
Sign in to GitHub.com
Sign in to GitHub Enterprise
New to GitHub? Create your free account.
Skip this step
By creating an account, you agree to the Terms of Service. For more information about GitHub's privacy practices, see the GitHub Privacy Statement.
GitHub Desktop sends usage metrics to improve the product and inform feature decisions.
Learn more about user metrics.



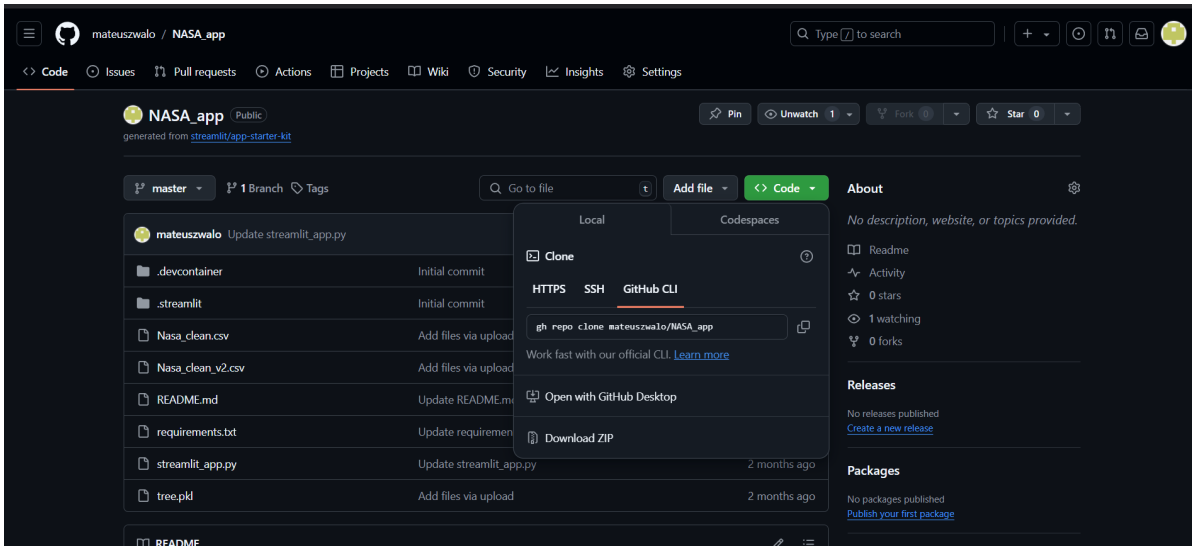
Wchodzimy na naszym profilu w zakładkę`Repositories` gdzie możemy po kliknieciu `New` stworzyć nowe repozytorium.



Możemy, skorzystać z aplikacji Desktopowej gdzie możemy zrobić to poprzez analogię. Możemy skonfigurować folder w którym tworzymy nasz projekt z aplikacją i wysyłać commity bezpośrednio z niego za pomocą komend bashowych git `push` i `pull`, lub aplikacji Desktopowej.

Jeśli chcemy stworzyć projekt bazujący na czyimś repozytroium starterowym, lub po prostu uczymy się (ewentualnie kradniemy czyjś kod, ale ciiiii nikomu nie mówcie) to możemy skopiować czyjes repozytroium za pomocą polecenia `git clone` + nazwa repozytorium. Pełną komendę do skopiowania repozytorium jakie nas interesuje znajdziemy bezpośrednio w repozytrium. Załóżmy że bardzo wam się spodobała moja apka dla NASA i chcecie ukraść do

niej kod, całę środowisko etc. To wchodzicie na mojego githuba, szukacie apki NASA i robicie dokładnie jak pokazałem poniżej:



i po wpisaniu komendy która się wyświetla w bashu, skopiujecie moją pracę, w celach oczywiście naukowych ta kradzież ;).

Ewentulanie wersja mniej informatyczna można pobrać po prostu ZIP'a i stworzyć środowisko w Anacondzie i wyjdzie na to samo ;)

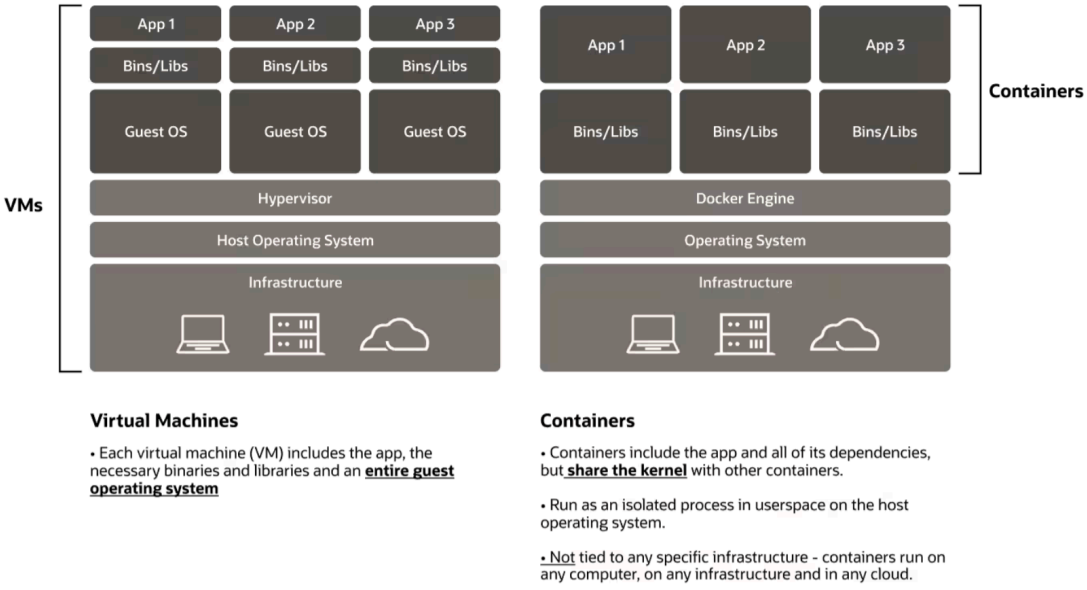
Wstęp do Dockera

Docker to platforma do konteneryzacji, która pozwala na tworzenie, zarządzanie i uruchamianie aplikacji w izolowanych środowiskach, zwanych kontenerami. Dzięki Dockerowi można tworzyć pakiety aplikacji wraz z wszystkimi niezbędnymi zależnościami (takimi jak biblioteki, pliki konfiguracyjne czy zależności systemowe), co umożliwia uruchamianie ich w różnych środowiskach bez ryzyka konfliktów.

Obraz Docker zawiera wszystko, co jest potrzebne do uruchamienia oprogramowania: kod, środowisko uruchomieniowe, sterowniki, narzędzia, skrypty, biblioteki, wdrożenia i inne.

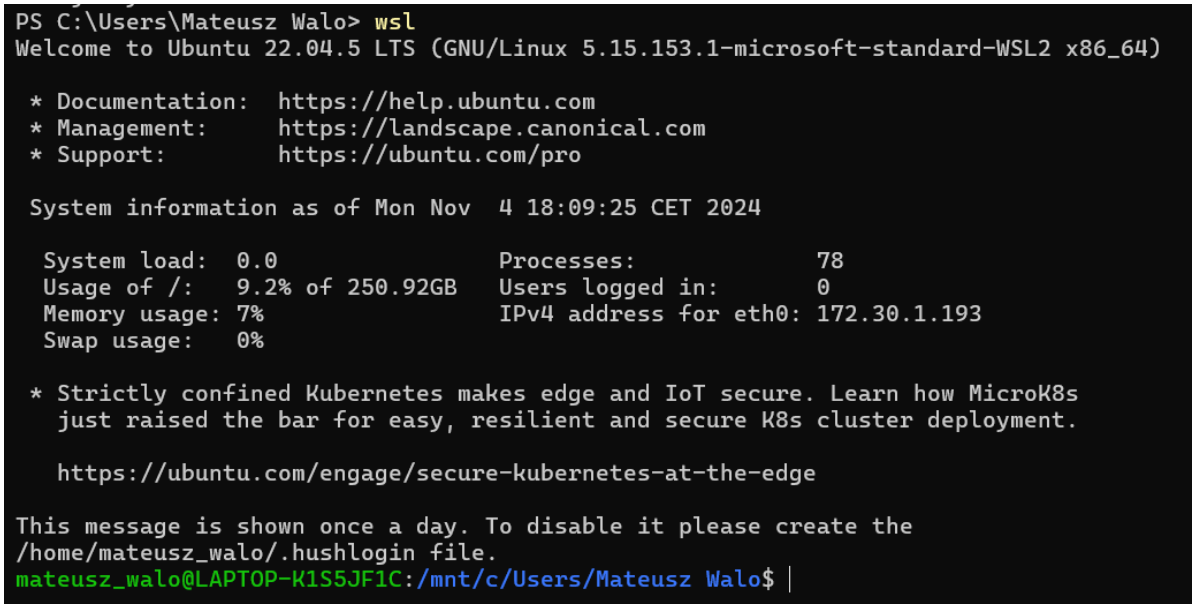
Kontener Docker jest działającą instancją obrazu Docker. W przeciwieństwie do tradycyjnej wirtualizacji z hipernadzorcą typu 1 lub 2, kontener Docker działa na jądrze hostującego systemu operacyjnego. W obrazie Docker nie ma osobnego systemu operacyjnego.

I to na tyle z teorii Dockerowej. Teraz ogarniemy sobie Dockera na własnym kompie krok po kroku.



Uwaga: Jeśli korzystacie z Windowsa to musicie zainstalować sobie WSL (Windows Subsystem for Linux) jeśli tego nie posiadacie jeszcze, ale na systemach operacyjnych u Dr. Zaburko już to powinno być i przydałaby się jakaś basic licencja Ubuntu na przyszłośc ;).

Jeśli korzystacie z Linuxa to wręcz cudownie w tym przypadku, podejrzewam że większość z was ma Windowas więc zaczniemy od sprawdzenia wersji `WSL`. Jeśli widzicie na ekranie to co ja to jesteśmy w domu :D



Następnie pobieramy `Docker Desktop` odpowiedni do naszego systemu operacyjnego

LINK: https://www.docker.com/products/docker-desktop/

Przechodzimy instalator krok po kroku i mam zainstalowanego Dockera.

Możemy przeglądać gotowe obrazy na `Docker Hub`

Podstawowe komendy w Dockerze:

- `docker pull <nazwa_obrazu>` – pobiera obraz z Docker Hub.
- `docker build -t <nazwa_obrazu> .` – buduje obraz z Dockerfile.
- `docker run <nazwa_obrazu>` – uruchamia kontener na podstawie obrazu.
- `docker ps` – wyświetla uruchomione kontenery.
- `docker stop <id_kontenera>` – zatrzymuje działający kontener.

W tym miejscu nastąpi żywa demonstracja działania jak się pracuje z Dockerem :D