Wyższa Szkoła Bankowa we Wrocławiu

O prowadzącym

Kamil Musiał

Tester w Tieto (wcześniej 7 lat tester / integrator / verification specialist w Nokia)

certyfikowany tester ISTQB (full advanced)

od 5 lat wykładowca WSB Wrocław (testowanie, telekomunikacja, sieci, IoT, Python)

doktorant Politechnika Wrocławska

fan morsowania zanim to było modne

kamil.musial@wsb.wroclaw.pl
kamil.musial@chorzow.wsb.pl

O prowadzącej

Natalia Kniat

- CI Lead / testerka / integratorka w Nokia
- certyfikowana testerka ISTQB
- trenerka Nokia Academy (testowanie, telekomunikacja, Python)
- zorganizowana mama z zamiłowaniem do wyrobów własnych

natalia.kniat@wsb.wroclaw.pl

Wprowadzenie do systemu kontroli wersji – GIT(8 h)

01 Git teoria

02 Git praktyka

03 Zdalne repozytorium

04 Github ćwiczenia

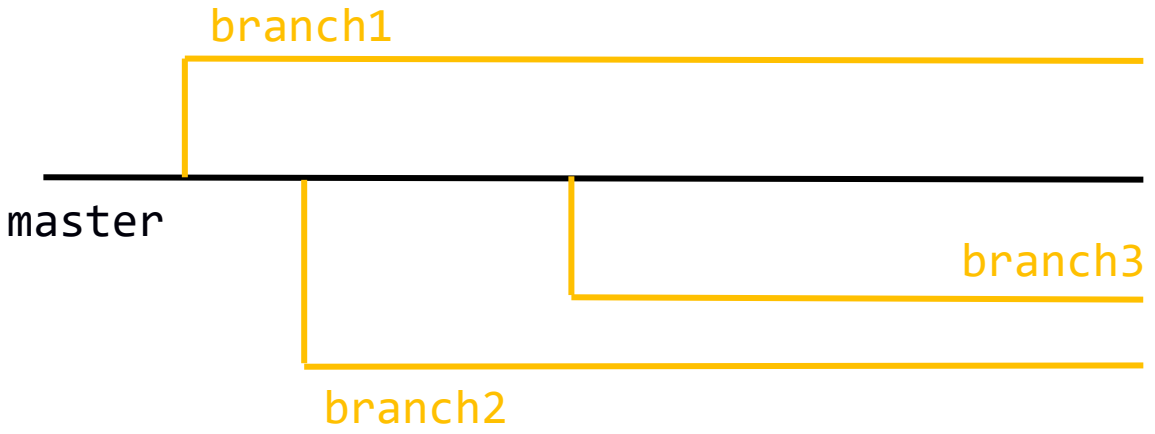
05 Wspólna praca - konflikt

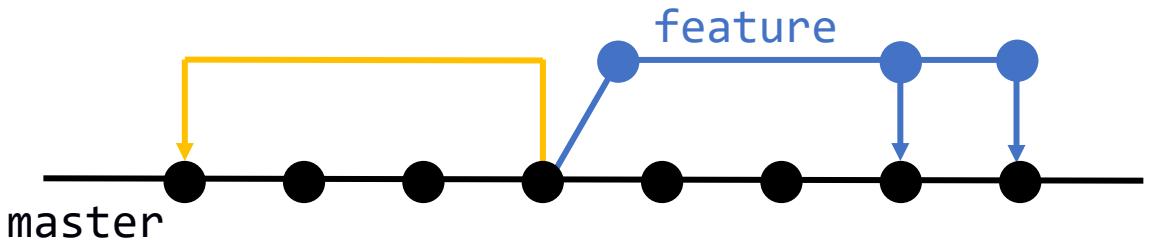
06 Github fork

Teoria – co robi GIT

System kontroli wersji
(SCM – Source Controll Management),
jest podstawą każdego projektu.

Możesz używać gita do
zachowywania aplikacji i skryptów
rozwijanych w czasie zajęć





Teoria – jak przygotować GITa

Rozpoczęcie pracy z GITem

1. Wejdz na https://vlabs.wsb.wroclaw.pl/portal
2. Utwórz folder, gdzie będziemy tworzyli nasze pierwsze repozytorium git:

```
mkdir nauka_git
cd nauka_git
```

3. Zainicjuj repozytorium git-a:

```
git init
```

do tego momentu GIT będzie działał w danym folderze

Teoria – jak przygotować GITa

Rozpoczęcie pracy z GITem

4. Skonfiguruj git-a:
   wpisz nazwę użytkownika oraz hasło:

   # sprawdź konfigurację (mała litera L):

```
git config –l
git config --global user.name „natalia_wsb”
git config --global user.email „natalia@wsb.wroclaw.pl"
```

- ✓ *GIT działa na Windowsie i Linuxie*
- ✓ *Nazwa użytkownika i email są tylko „dla nas” - będą widoczne przy sprawdzaniu dokonanych zmian*

Teoria – jak używać GITa

Rozpoczęcie pracy z GITem

`git status` sprawdzenie statusu
`git add <plik>` dodanie pliku do śledzenia
`git add .` dodanie całego folderu / projektu do śledzenia
`git commit -m „message”` zapisanie stanu / stworzenie kopii
`git log` sprawdzenie historii zmian
`git checkout <commit_hash>` przejście do danego commita

Dobre źródło:
https://www.notion.so/zarkom/Introduction-to-Git-ac396a0697704709a12b6a0e545db049

Teoria – jak używać GITa

Rozpoczęcie pracy z GITem

`git checkout –b <branch_name>` stworzenie nowego branch'a

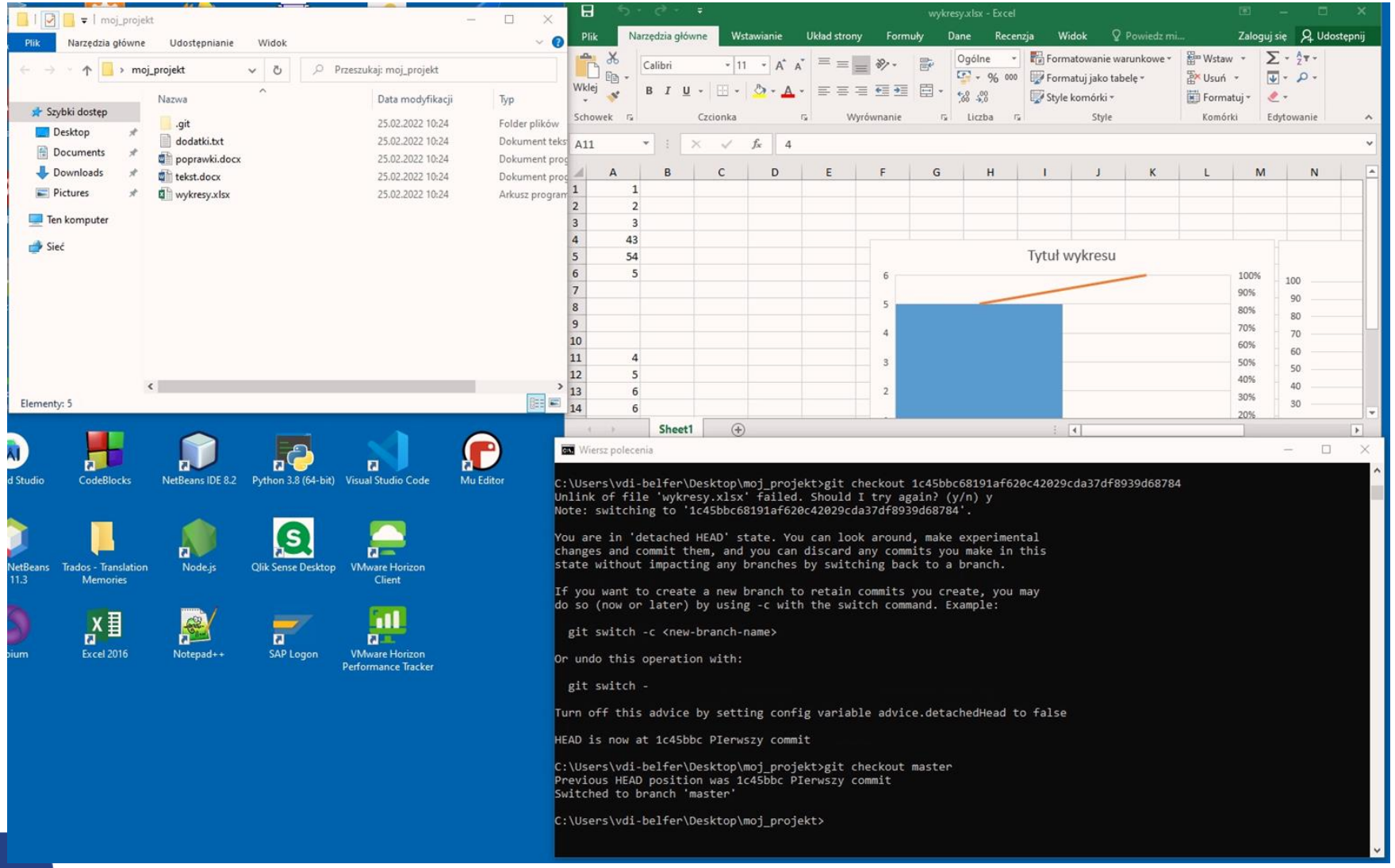
`git checkout <branch_name>` przejście do danego branch'a

`git merge <branch_name>`

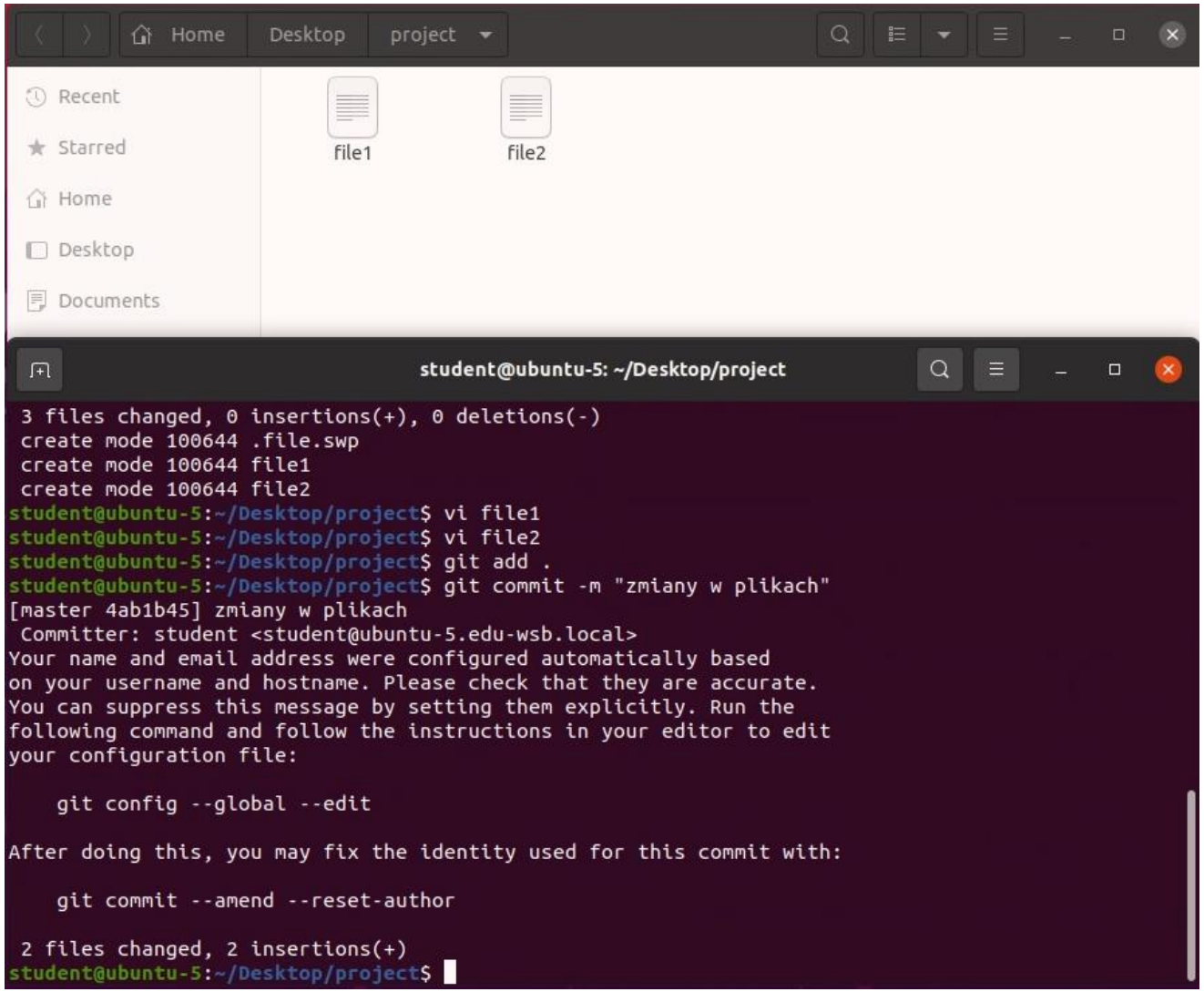
`gitk` sprawdzenie historii

Dobre źródło:
https://www.notion.so/zarkom/Introduction-to-Git-ac396a0697704709a12b6a0e545db049

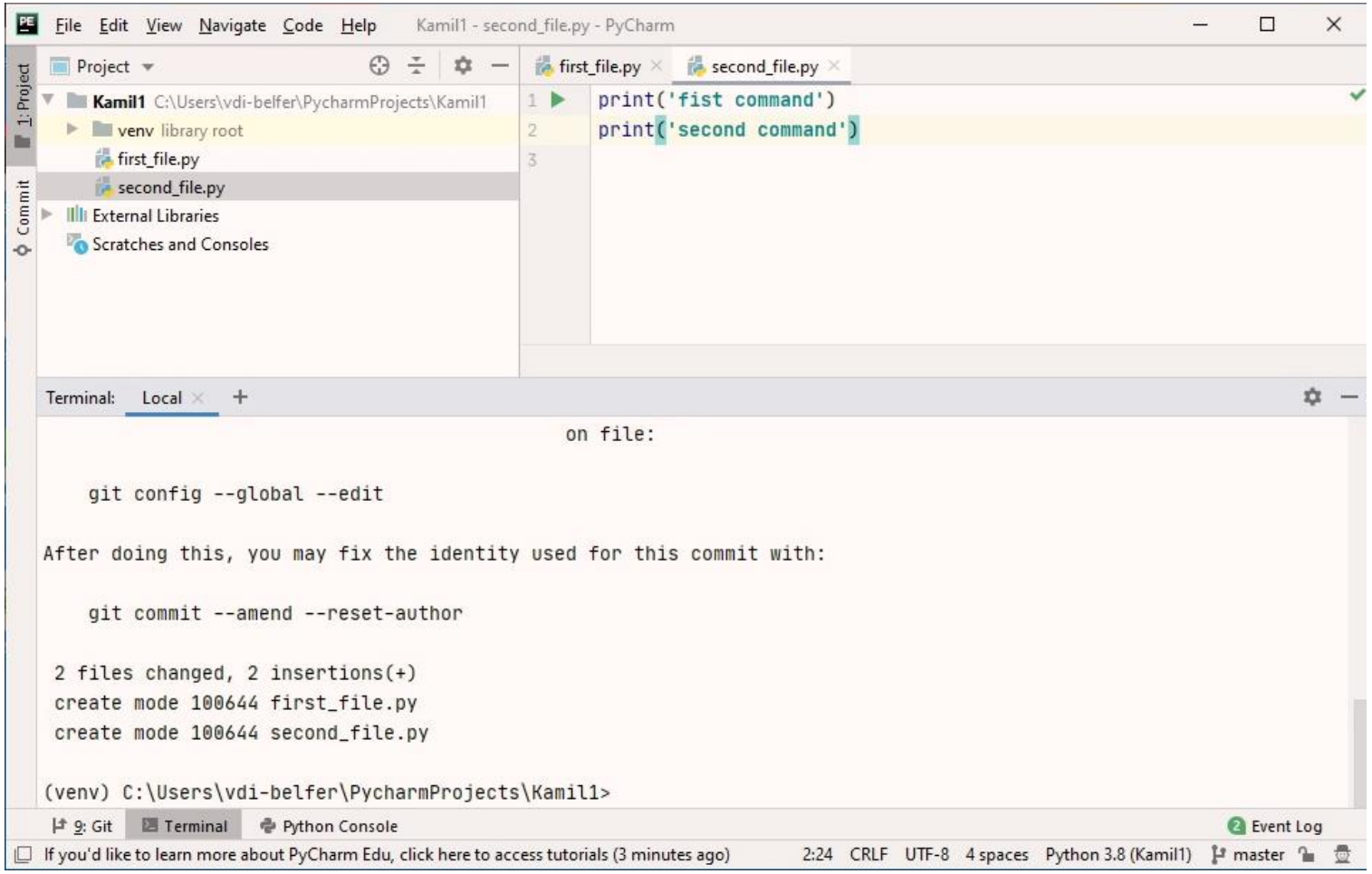
Ćwiczenie Windows



Ćwiczenie Linux

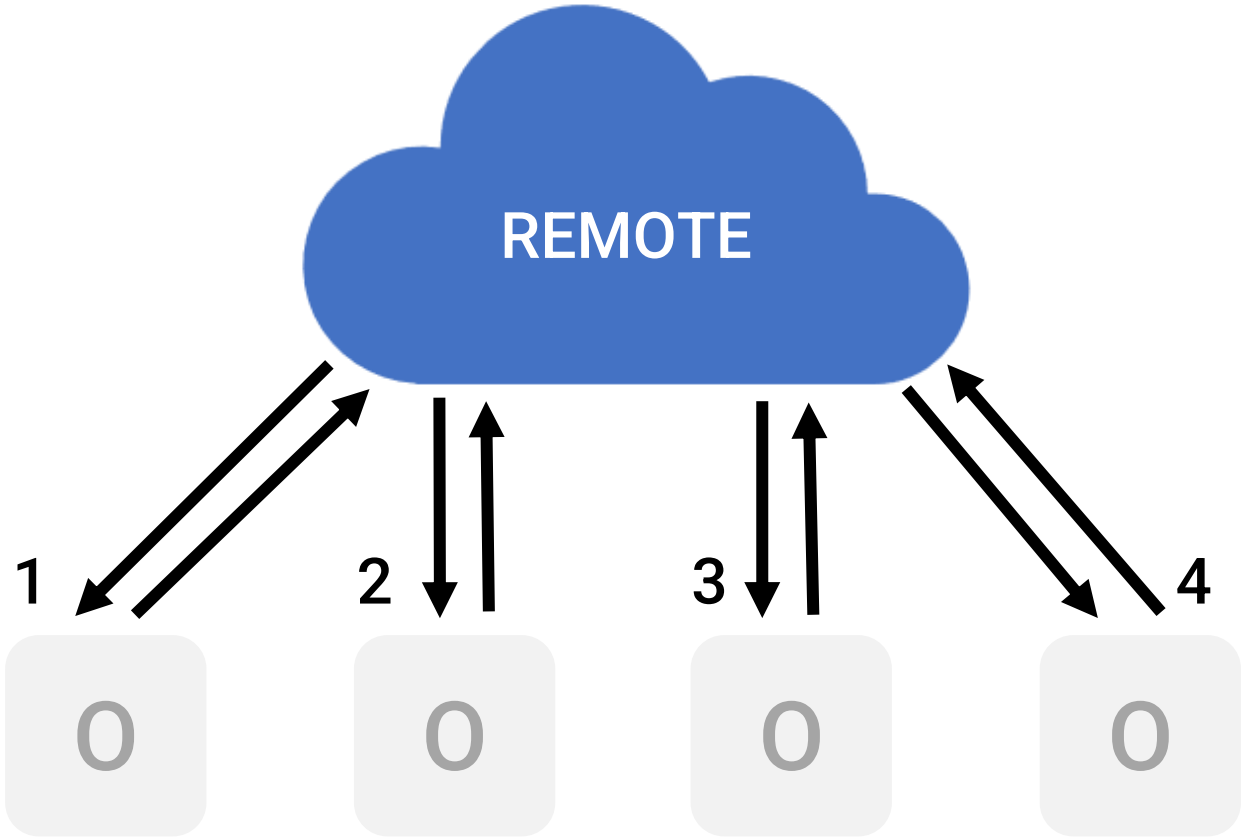


Ćwiczenie PyCharm



Remote repository

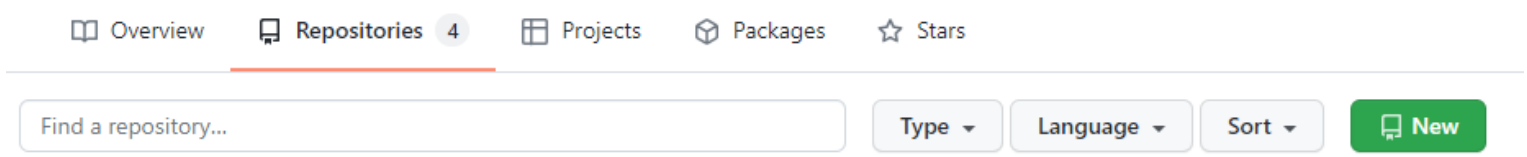
Filozofia pracy



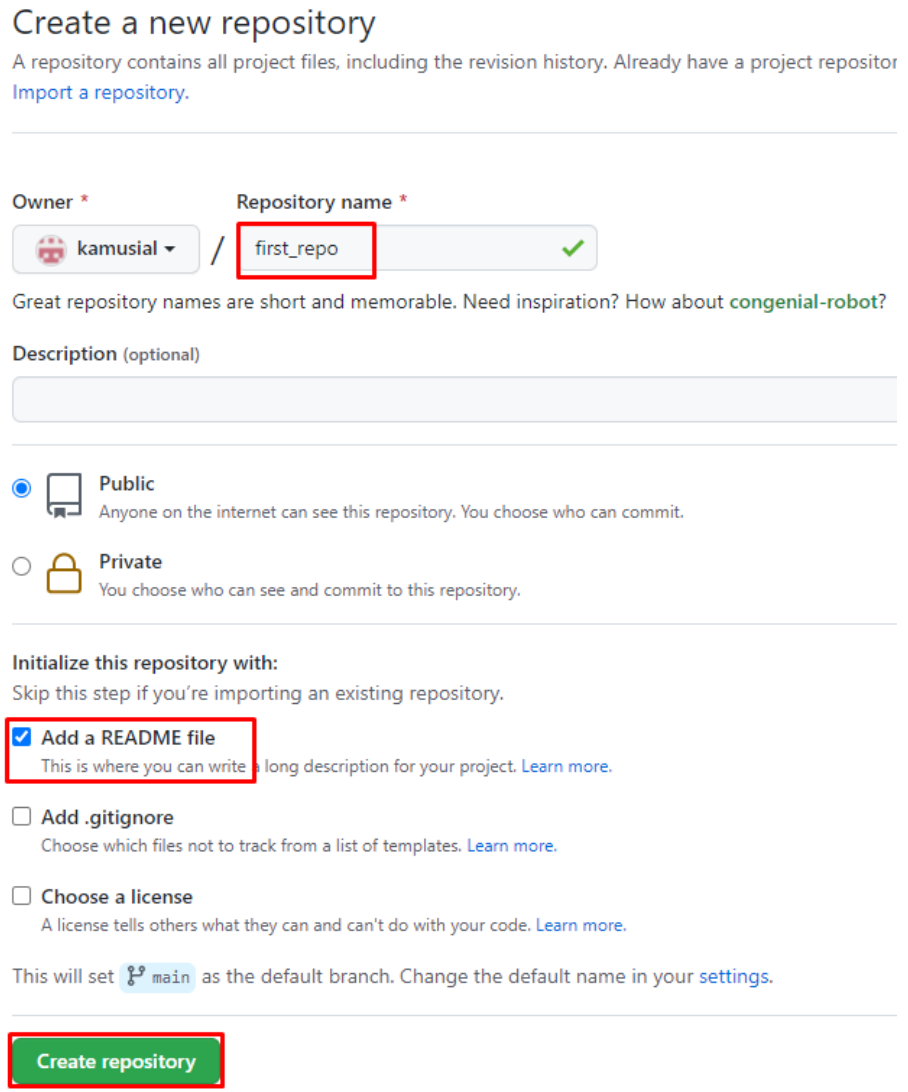
Przygotowanie środowiska

1. Załóż konto na https://github.com/join

2. Stwórz repozytorium



Przygotowanie środowiska

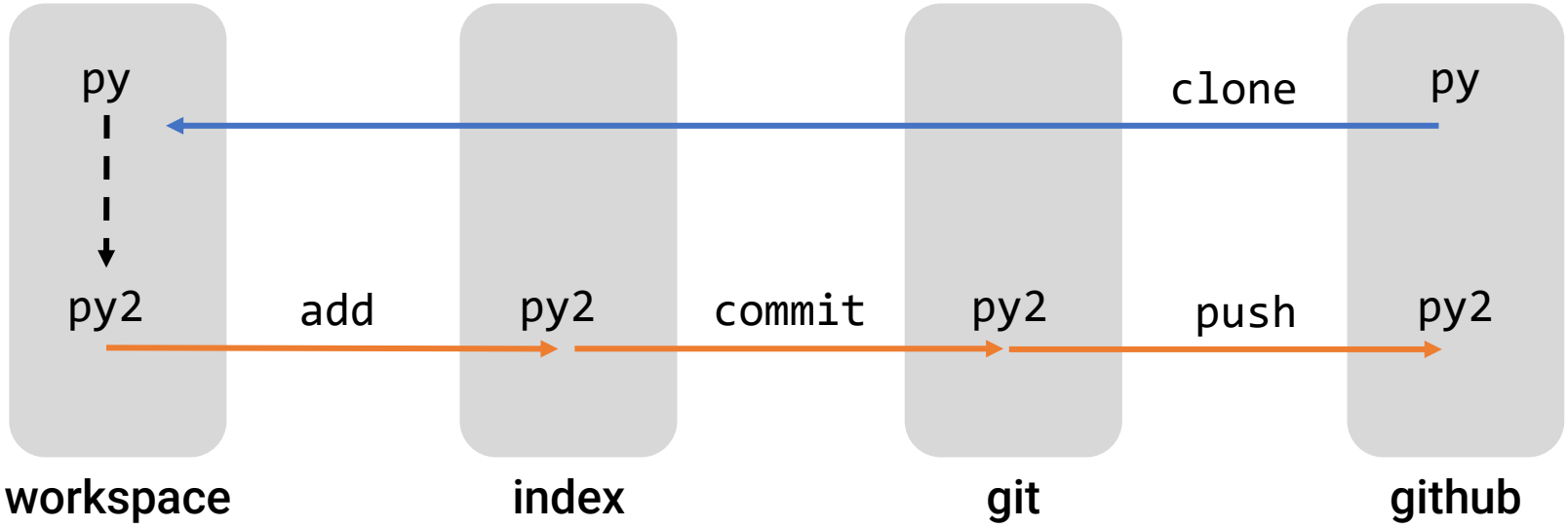


Przygotowanie środowiska

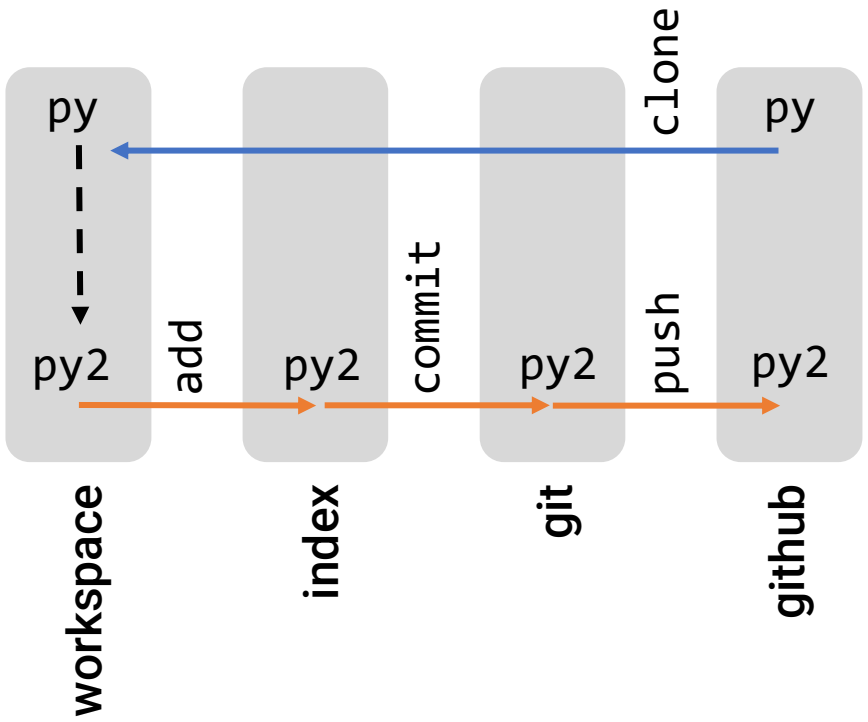
3. Skopiuj / zapisz link do swojego repo:

   https://github.com/kamusial/first_repo

Podstawowe komendy



Podstawowe komendy



Pobranie kopii repo:

```
git clone https://github.com/kamusial/first_repo
```

`git status` sprawdzenie statusu

`git add <plik>` dodanie pliku do śledzenia

`git commit -m „message”` zapisanie stanu / stworzenie kopii

`git push` upload zmian na github

`git log` historii sprawdzenie commitów

`git chechout <commit_hash>`

`git checkout –b <nazwa_branch’a>`
stworzenie nowego brancha
i przejście do niego

`git checkout <branch_name>` przejście do danego branch’a

`git merge <branch_name>` mergowanie zmian

Ćwiczenie

Projekty własne

Otwórz swoje repo w PyCharmie

01

03

Zupdatuj repo na github

Nanieś zmiany

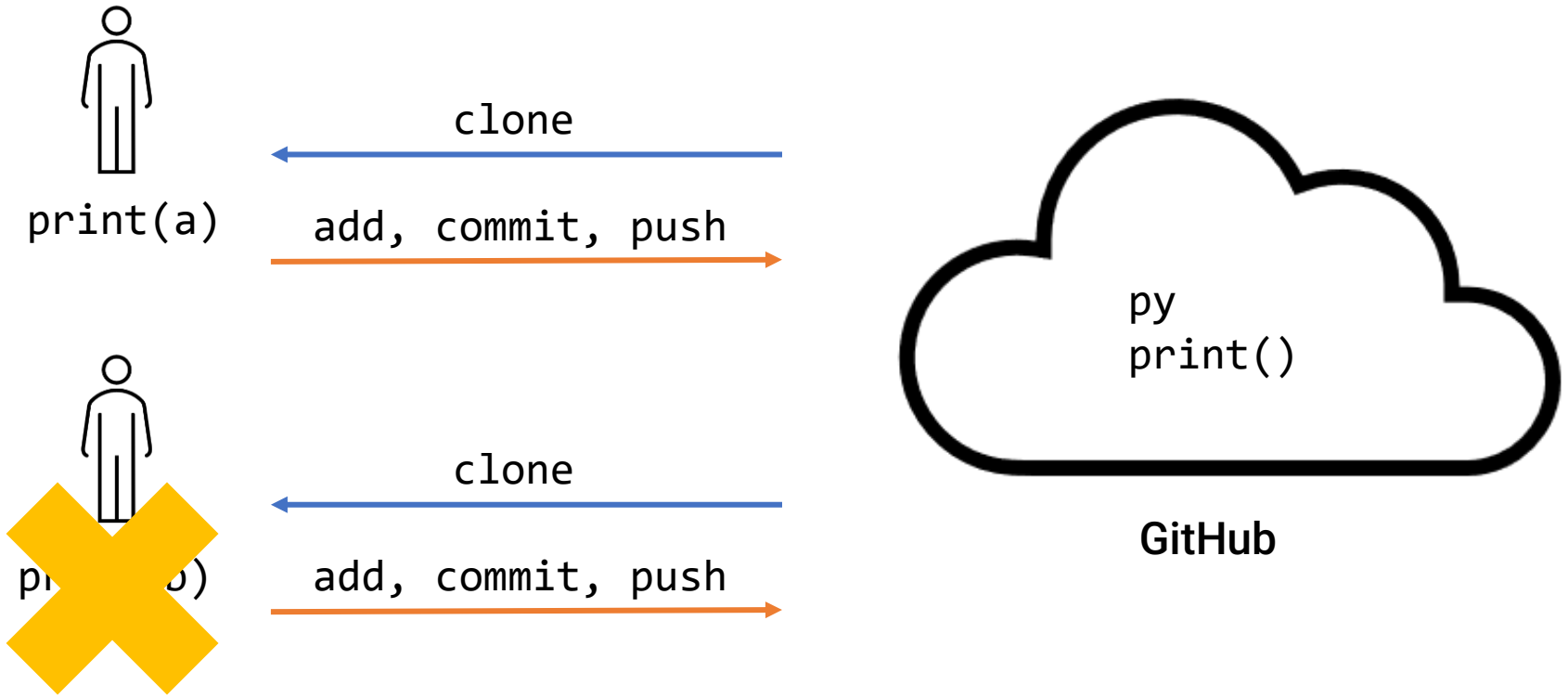
02

04

Stwórz 3 branche – zobacz, jak to wygląda w pycharmie

Wspólna praca

Konflikt



Wspólna praca

Podział na pokoje (MS Teams) lub podział samodzielny

01

Jedna osoba tworzy repo i dodaje pozostałych, jako kolaboratorów

02

Zadanie

Napisać program, który:

- pyta, ile użytkownik ma PLN i oblicza ile to USD
- pyta o niezbędne rzeczy i informuje, czy użytkownik może inwestować
- proponuje konkretne inwestycje bądź odradza

03

Należy:

- Stworzyć moduł z funkcją przyjmującą ilość pieniędzy w PLN oraz kurs i zwracającą ilość USD
- Stworzyć moduł z funkcją przyjmującą płeć, wiek, poziom zdolności finansowej i zwracającą decyzję, czy dana osoba może inwestować (logika dowolna)
- Stworzyć moduł z funkcją proponującą inwestycję w dane aktywa w zależności od ilości posiadanych USD

04

Każdy z uczestników działa na jednym pliku (omijamy konflikty)

Github

Umieszczenie lokalnego projektu

1. tworzymy repo na github

1. kopiujemy repo do siebie

```
git clone <repo>
```

3. dodajemy do niego nasze pliki

```
git add
git commit
```

4. wprowadzamy zmiany

```
git push
```

Github

Co nam podpowiada?

...or create a new repository on the command line

```
echo "# repo_to_push" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/kamusial/repo_to_push.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/kamusial/repo_to_push.git
git branch -M main
git push -u origin main
```

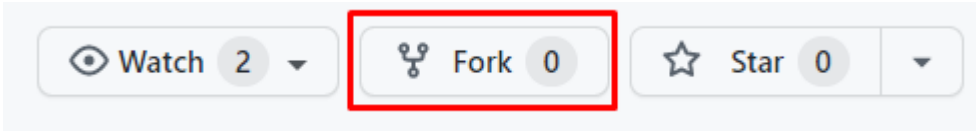
Github

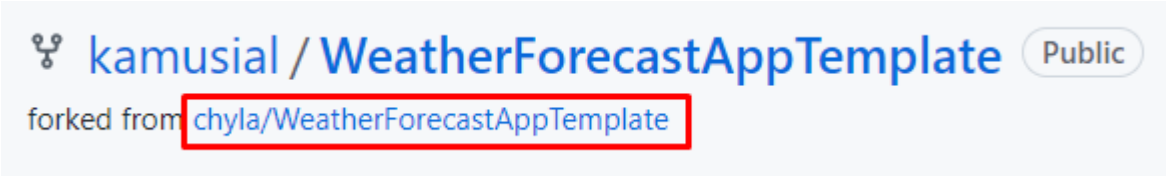
`git branch` wyświetlenie branchy
`git log <nazwa brancha>` historia danego brancha
`git diff` wyświetlenie wprowadzonych zmian

Pull request

Gdy chcemy wprowadzić zmiany w czyimś projekcie

1. Kopiujemy dane repo do swojego repo na GitHub -> **fork**



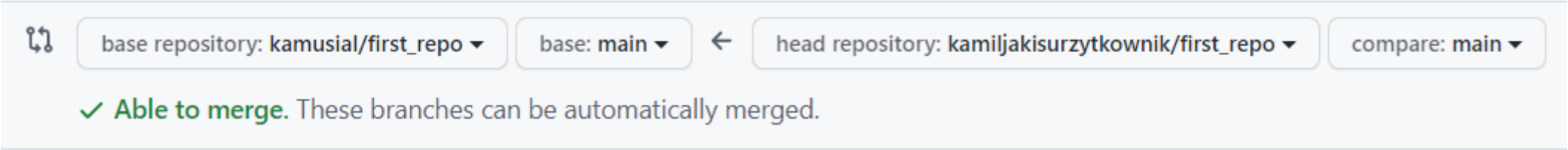


2. Clonujemy repo do siebie (git clone .....)
3. Wprowadzamy zmiany (git add, git commit, git push)

Pull request

Gdy chcemy wprowadzić zmiany w czyimś projekcie

1. Z pozycji GitHuba tworzymy „pull request”



Wspólna praca

Podział na pokoje (MS Teams) lub podział samodzielny

01

Słuchacze „forkują” repozytorium prowadzącego: https://github.com/nkniat/git_python_cwiczenia do siebie

02

Jedna osoba w zespole tworzy z forkowanego repo od prowadzącego swoje własne. Pozostali „forkują” repo od kolegi/kolezanki do siebie i dzielimy się w zespole.
Jedna osoba pisze kod główny,
Pozostałe osoby – każda po jednej funkcji.

03

Napisać program, który oblicza prawdopodobieństwo, że pacjent będzie miał problemy z sercem na podstawie poziomy stresu, ruchu i diety pacjenta.
Napisać brakujący kod, aby program zadziałał. Działamy na 2 plikach:

- główny plik
- plik z funkcjami

04

Programiści piszący funkcje poboczne wykonują pull request do kolegi z głównym kodem programu.

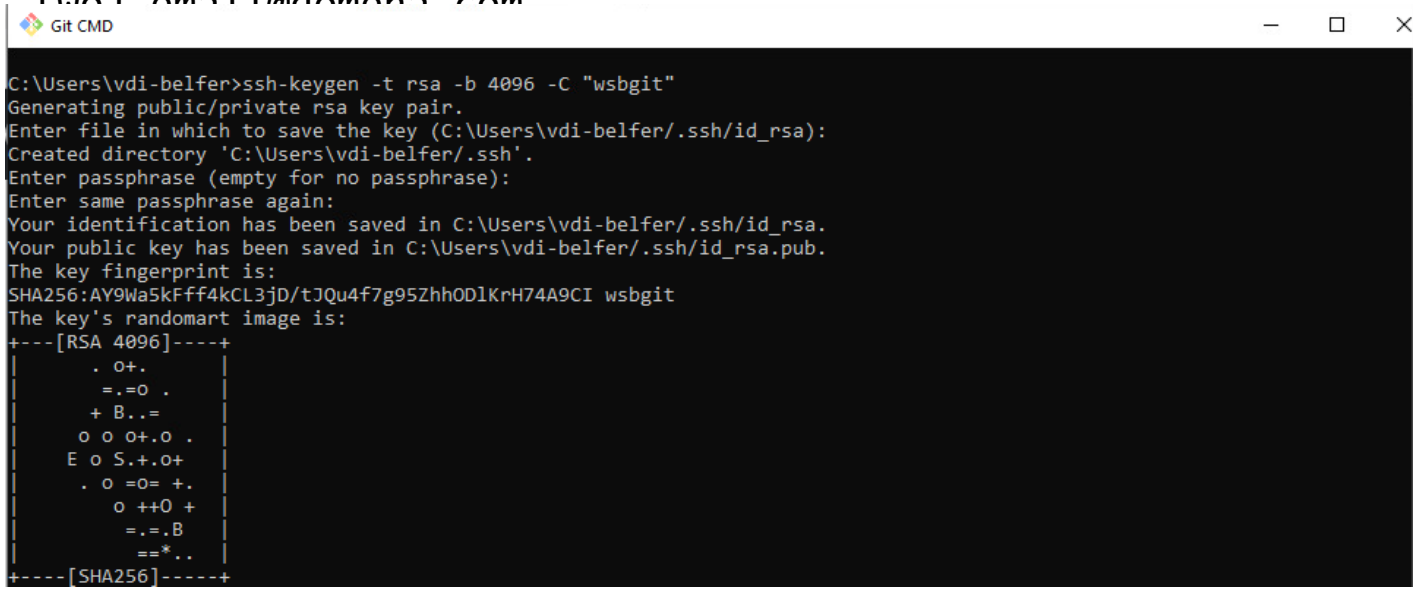
Git

Generowanie klucza

1. Uruchom konsolę Git Bash

1. Wpisz komendę:

```
ssh-keygen -t rsa -b 4096 –C
"twoj_email@domena.com"
```



Git

Generowanie klucza

1. Zostaniesz poproszony o wskazanie lokalizacji, w której zapisać klucz.
   `Enter a file in which to save the key (/c/Users/Twoj_login/.ssh/id_rsa):`

1. Wciśnij Enter

1. Zostaniesz zapytany o hasło (passphrase).
   `Enter passphrase (empty for no passphrase):`
   `Enter same passphrase again:`

1. Wciśnij Enter

Git

Dodawanie do agenta SSH

1. Dodaj klucz do agenta SSH
    1. Uruchom agenta (jeśli nie jest właczony):
       `eval $(ssh-agent -s)`

    1. Dodaj klucz:
       `ssh-add ~/.ssh/id_rsa`

    1. Alternatywnie użyj polecenia:
       `start-ssh-agent.cmd`

```
C:\Users\vdi-belfer>start-ssh-agent.cmd
Removing old ssh-agent sockets
Starting ssh-agent:  done
Identity added: /c/Users/vdi-belfer/.ssh/id_rsa (wsbgit)

C:\WINDOWS\system32\cmd.exe /K "doskey git=^"C:\Program Files\Git\cmd\git.exe" $*" | @FINDSTR /l "\"\"" >NUL
Microsoft Windows [Version 10.0.19044.1526]
(c) Microsoft Corporation. Wszelkie prawa zastrzeżone.

C:\Users\vdi-belfer>_
```

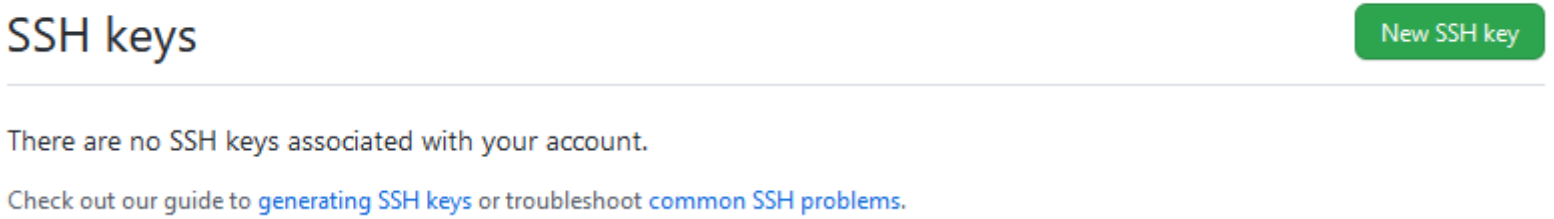
Git

Dodanie klucza SSH do konta GitHub

1. Otwórz plik id_rsa.pub zwykłym edytorem tekstu (notatnik) i skopiuj jego zawartość.

1. Przejdź na stronę https://github.com/settings/keys lub Settings > Access > SSH and GPG keys

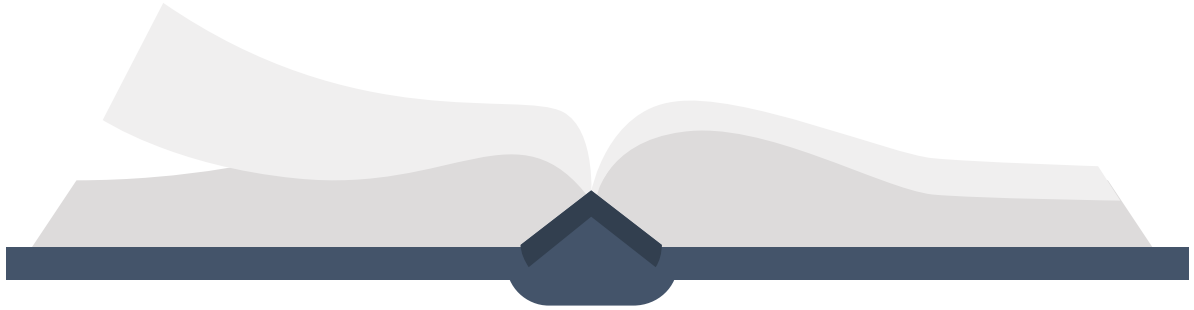
1. Dodaj nowy klucz używając przycisku New SSH key. Nadaj tytuł i wklej klucz ze schowka.



1. Zakończ przyciskiem Add SSH key

Materiały

- Introducing GitHub: A Non-Technical Guide, Peter Bell
- Github for Dummies, Sarah Guthals
- „Python 3. Projekty dla początkujących i pasjonatów”, Adam Jurkiewicz
- „PYTHON dla testera”, Piotr Wróblewski



Koniec

Dziękuję za uwagę