



# 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](mailto:kamil.musial@wsb.wroclaw.pl)

[kamil.musial@chorzow.wsb.pl](mailto: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](mailto:natalia.kniat@wsb.wroclaw.pl)

# Wprowadzenie do systemu kontroli wersji – GIT(8 h)



01

Git  
teoria

04

Github  
ćwiczenia

02

Git  
praktyka

05

Wspólna praca  
- konflikt

03

Zdalne  
repozytorium

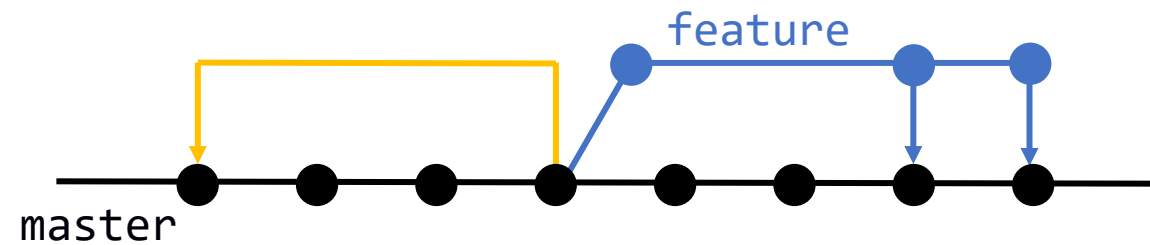
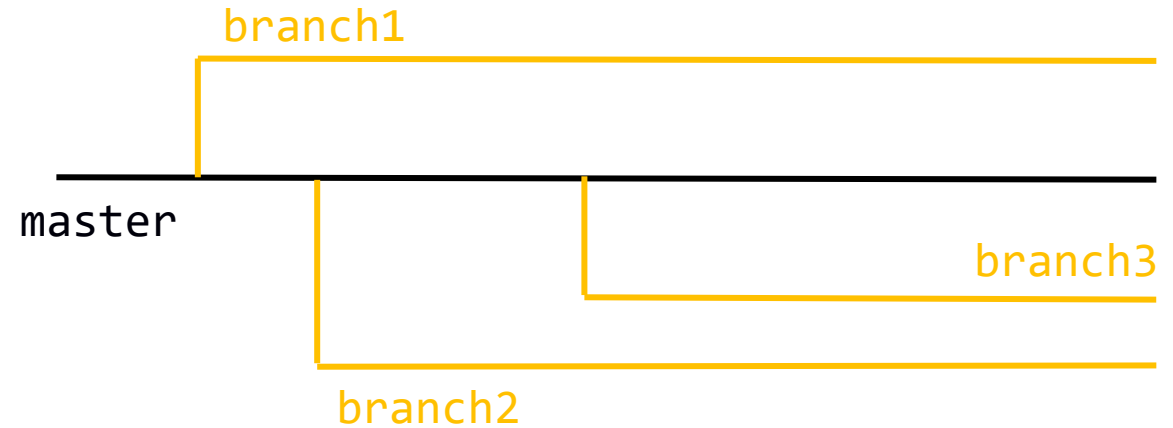
06

Github  
fork

# Teoria – co robi GIT

System kontroli wersji  
(SCM – Source Control 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/porta1>
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



## Podstawowe komendy

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

Dobre źródło:

<https://www.notion.so/zarkom/Introduction-to-Git-ac396a0697704709a12b6a0e545db049>



# Teoria – jak używać GITa

Rozpoczęcie pracy z GITem



## Operacje na branch'ach

`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

The screenshot displays a Windows desktop with a blue background. The taskbar at the bottom contains icons for various applications: Studio, CodeBlocks, NetBeans IDE 8.2, Python 3.8 (64-bit), Visual Studio Code, Mu Editor, NetBeans 11.3, Trados - Translation Memories, Node.js, Qlik Sense Desktop, VMware Horizon Client, Excel 2016, Notepad++, SAP Logon, and VMware Horizon Performance Tracker.

Three windows are open:

- File Explorer:** Shows the contents of the 'moj\_projekt' folder. The files listed are: '.git', 'dodatki.txt', 'poprawki.docx', 'tekst.docx', and 'wykresy.xlsx'. The 'Data modyfikacji' column shows the date 25.02.2022 10:24 for all files.
- Excel (wykresy.xlsx):** Displays a spreadsheet with data in columns A and B. A bar chart is visible on the right side of the sheet, titled 'Tytuł wykresu'. The chart shows a single blue bar representing the value 43 in cell A4. The y-axis ranges from 0 to 100%.
- Terminal (Wiersz polecenia):** Shows the execution of Git commands to checkout a specific commit and switch to the master branch.

```
C:\Users\vdi-belfer\Desktop\moj_projekt>git checkout 1c45bbc68191af620c42029cda37df8939d68784
Unlink of file 'wykresy.xlsx' failed. Should I try again? (y/n) y
Note: switching to '1c45bbc68191af620c42029cda37df8939d68784'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

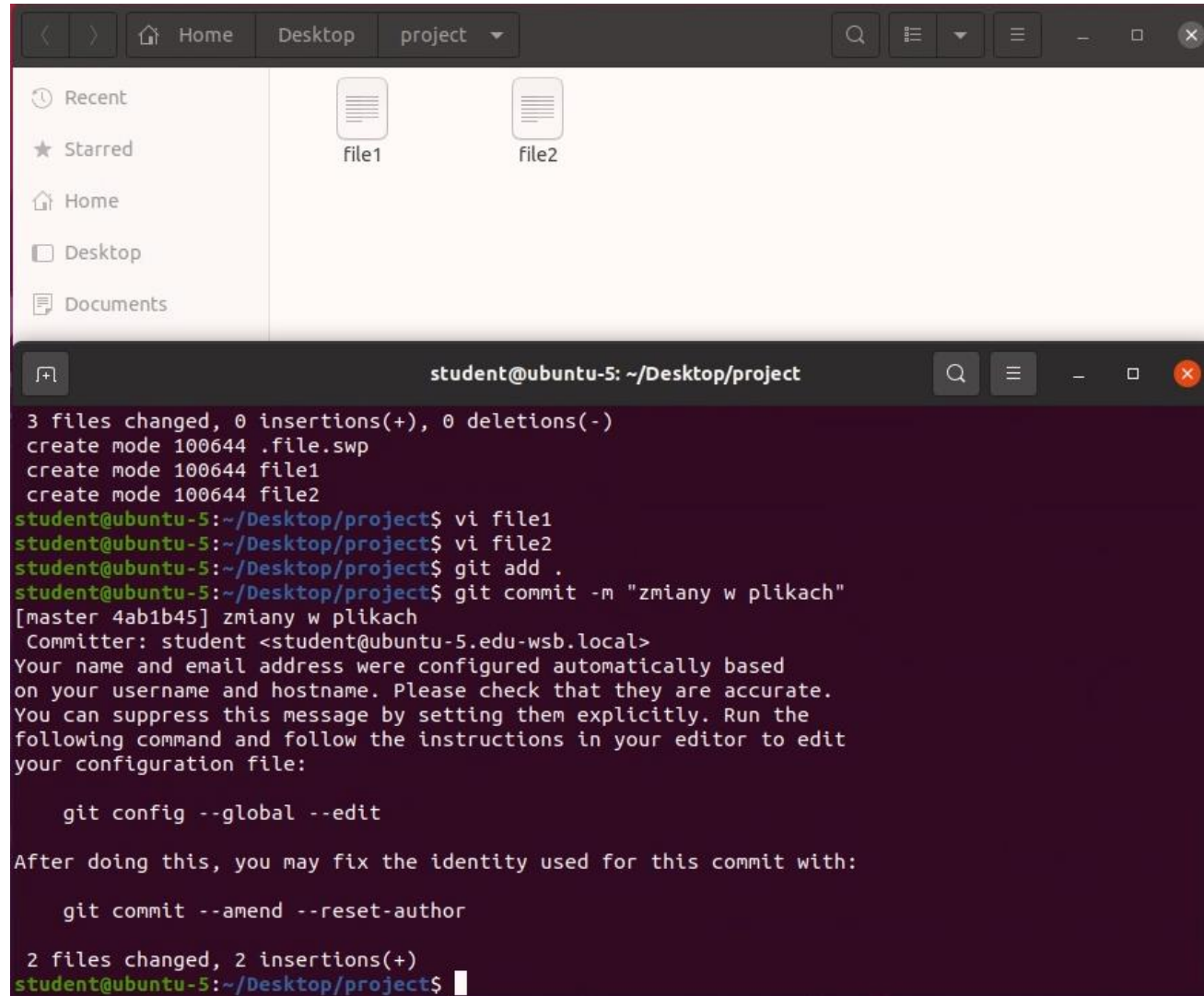
Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 1c45bbc Pierwszy commit

C:\Users\vdi-belfer\Desktop\moj_projekt>git checkout master
Previous HEAD position was 1c45bbc Pierwszy commit
Switched to branch 'master'

C:\Users\vdi-belfer\Desktop\moj_projekt>
```

# Ćwiczenie Linux



The screenshot displays a Linux desktop environment. At the top, a file manager window is open, showing the 'project' directory on the Desktop. It contains two files, 'file1' and 'file2'. Below the file manager, a terminal window is open, showing the execution of several commands to create files, edit them with 'vi', and commit them to a Git repository. The terminal output shows the successful creation of three files, the editing of two, and the commit of two files with a message 'zmiany w plikach'.

```
student@ubuntu-5: ~/Desktop/project
3 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 .file.swp
create mode 100644 file1
create mode 100644 file2
student@ubuntu-5:~/Desktop/project$ vi file1
student@ubuntu-5:~/Desktop/project$ vi file2
student@ubuntu-5:~/Desktop/project$ git add .
student@ubuntu-5:~/Desktop/project$ git commit -m "zmiany w plikach"
[master 4ab1b45] zmiany w plikach
Committer: student <student@ubuntu-5.edu-wsb.local>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

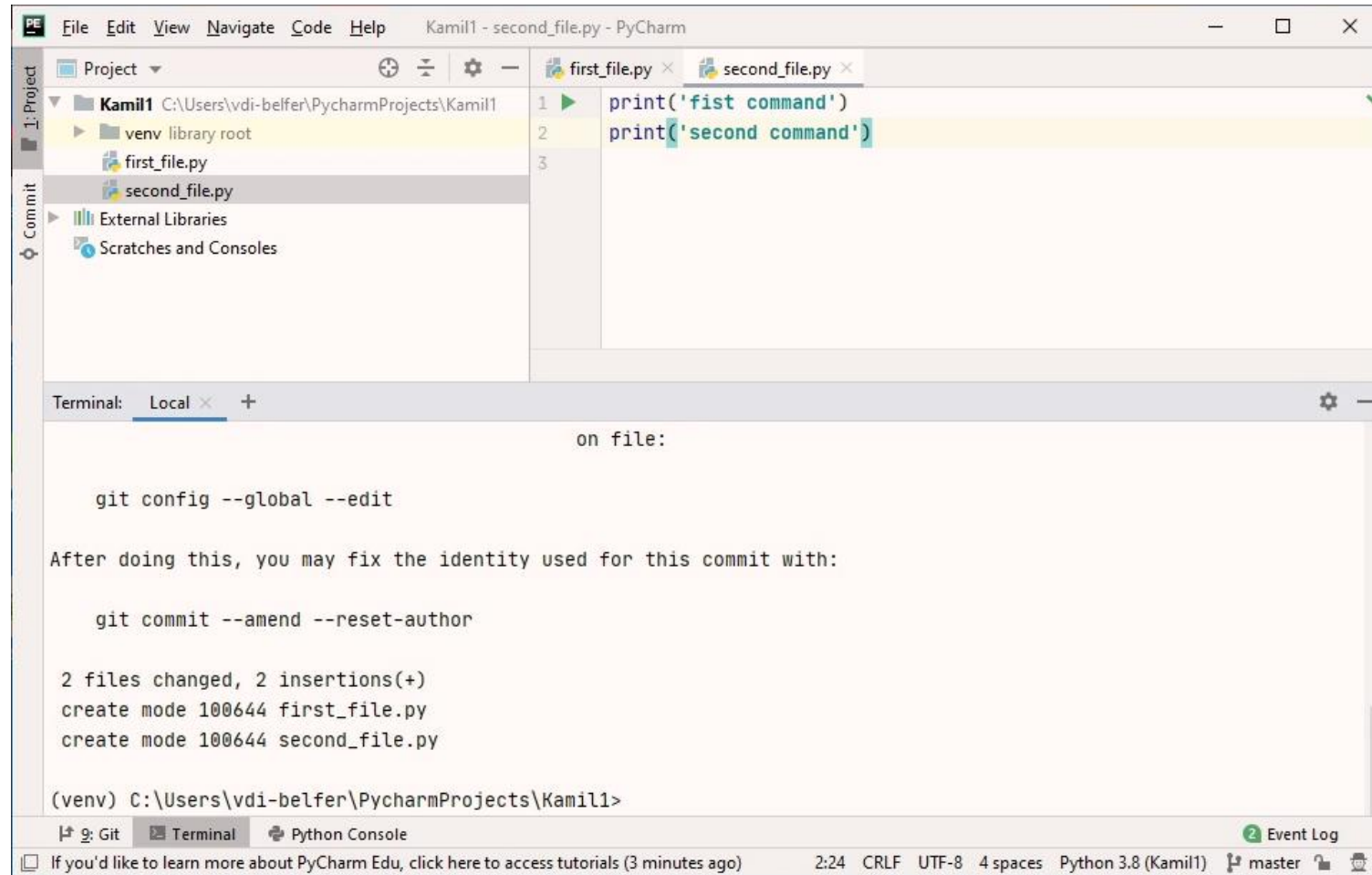
    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

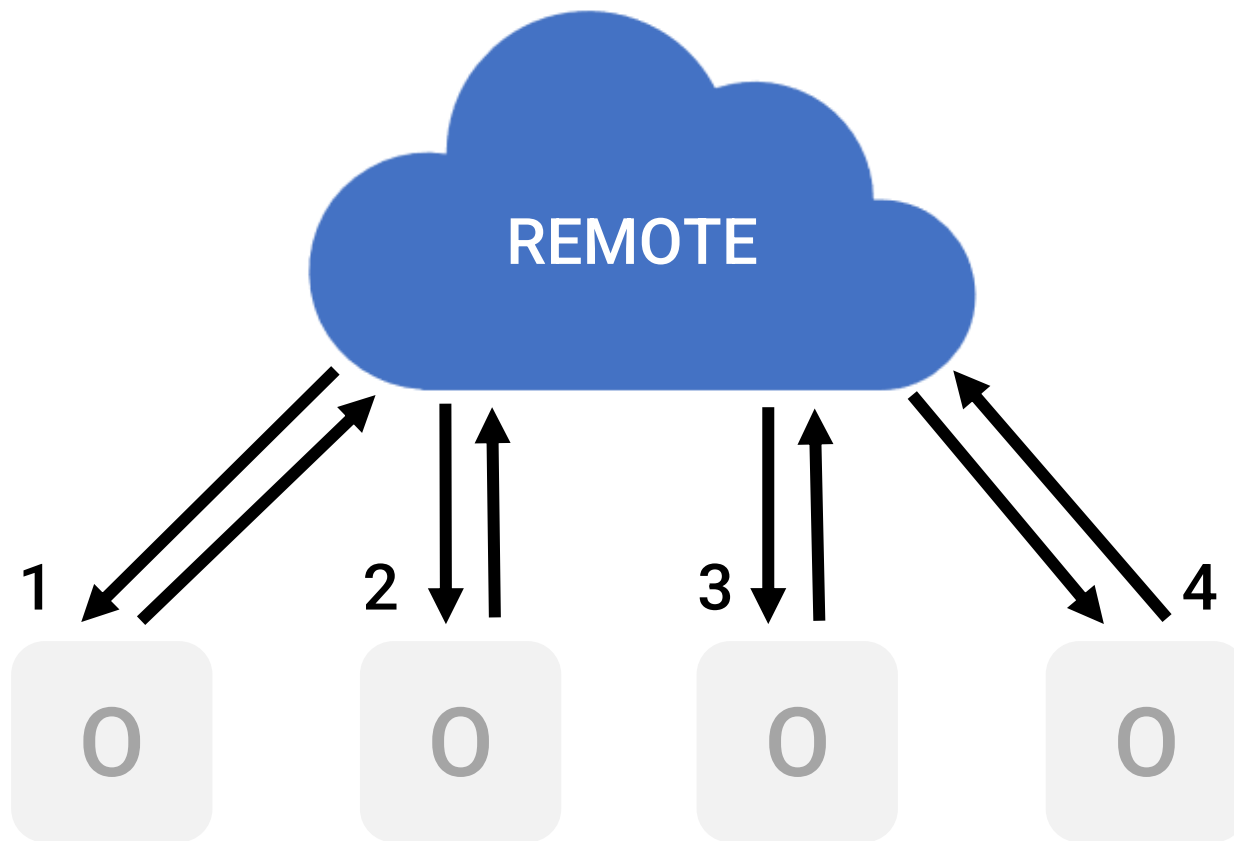
2 files changed, 2 insertions(+)
student@ubuntu-5:~/Desktop/project$
```

# Ćwiczenie PyCharm



# Remote repository

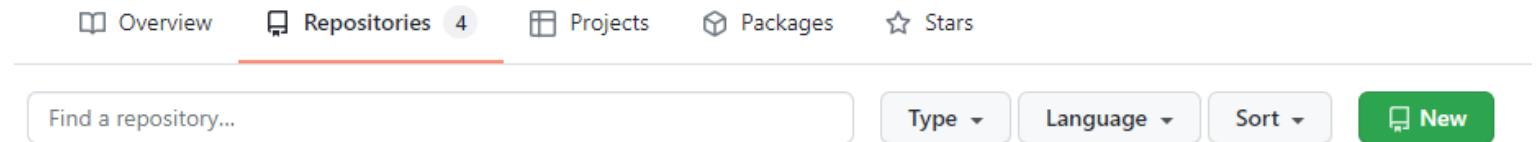
Filozofia pracy



# Przygotowanie środowiska



1. Załóż konto na <https://github.com/join>
2. Stwórz repozytorium





# Przygotowanie środowiska



## Create a new repository



A repository contains all project files, including the revision history. Already have a project repository? [Import a repository.](#)

Owner \* Repository name \*

 kamusial / first\_repo 

Great repository names are short and memorable. Need inspiration? How about [congenial-robot?](#)

Description (optional)

- ☒  **Public**  
Anyone on the internet can see this repository. You choose who can commit.
- ☐  **Private**  
You choose who can see and commit to this repository.


### Initialize this repository with:

Skip this step if you're importing an existing repository.

☒ **Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

☐ **Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)

This will set  **main** as the default branch. Change the default name in your [settings](#).

Create repository

# Przygotowanie środowiska

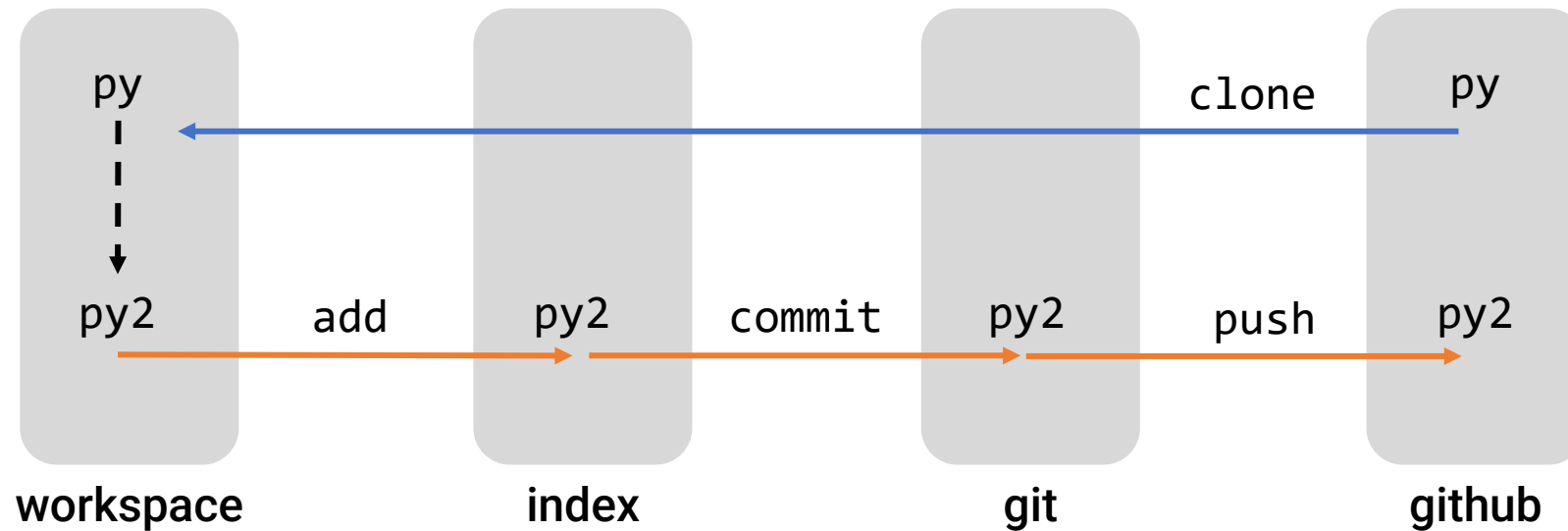


3. Skopiuj / zapisz link do swojego repo:

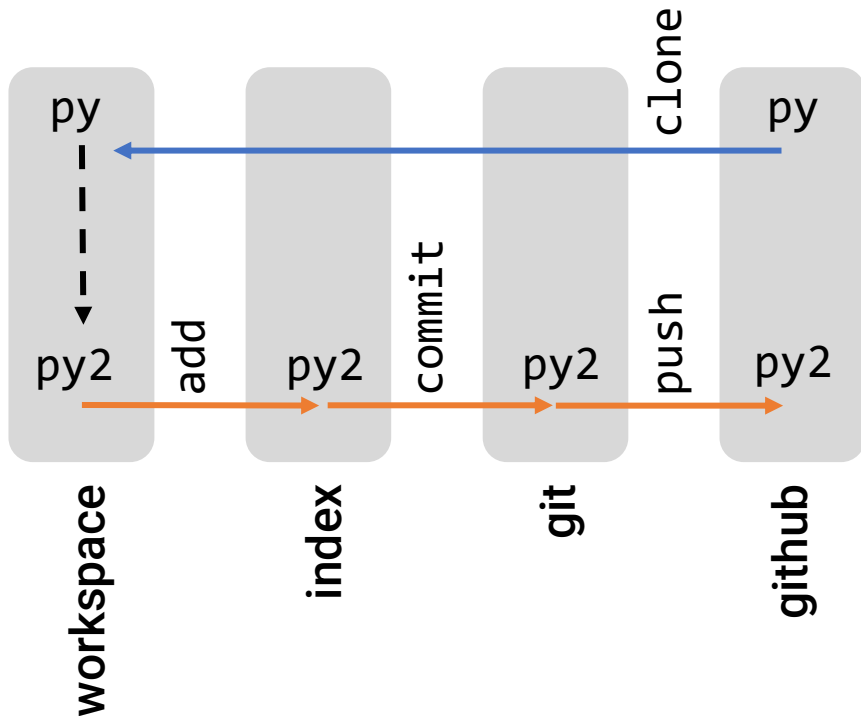
[https://github.com/kamusial/first\\_repo](https://github.com/kamusial/first_repo)



# Podstawowe komendy



# Podstawowe komendy



Pobranie kopii repo:

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

<code>git status</code>	sprawdzenie statusu
<code>git add &lt;plik&gt;</code>	dodanie pliku do śledzenia
<code>git commit -m „message”</code>	zapisanie stanu / stworzenie kopii
<code>git push</code>	upload zmian na github
<code>git log</code>	historii sprawdzenie commitów
<code>git chechout &lt;commit_hash&gt;</code>	
<code>git checkout -b &lt;nazwa_branch'a&gt;</code>	stworzenie nowego brancha i przejście do niego
<code>git checkout &lt;branch_name&gt;</code>	przejście do danego branch'a
<code>git merge &lt;branch_name&gt;</code>	mergowanie zmian

# Ćwiczenie

Projekty własne

Otwórz swoje repo w  
PyCharmie

01

Nanieś zmiany

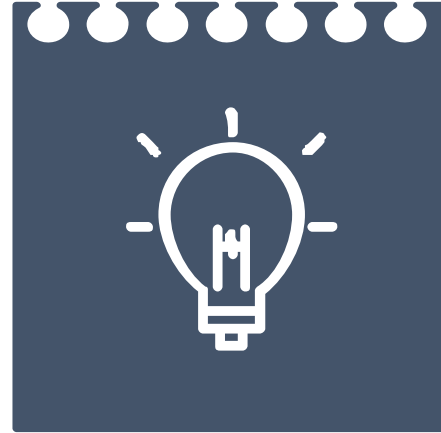
02

Zupdatuj repo na github

03

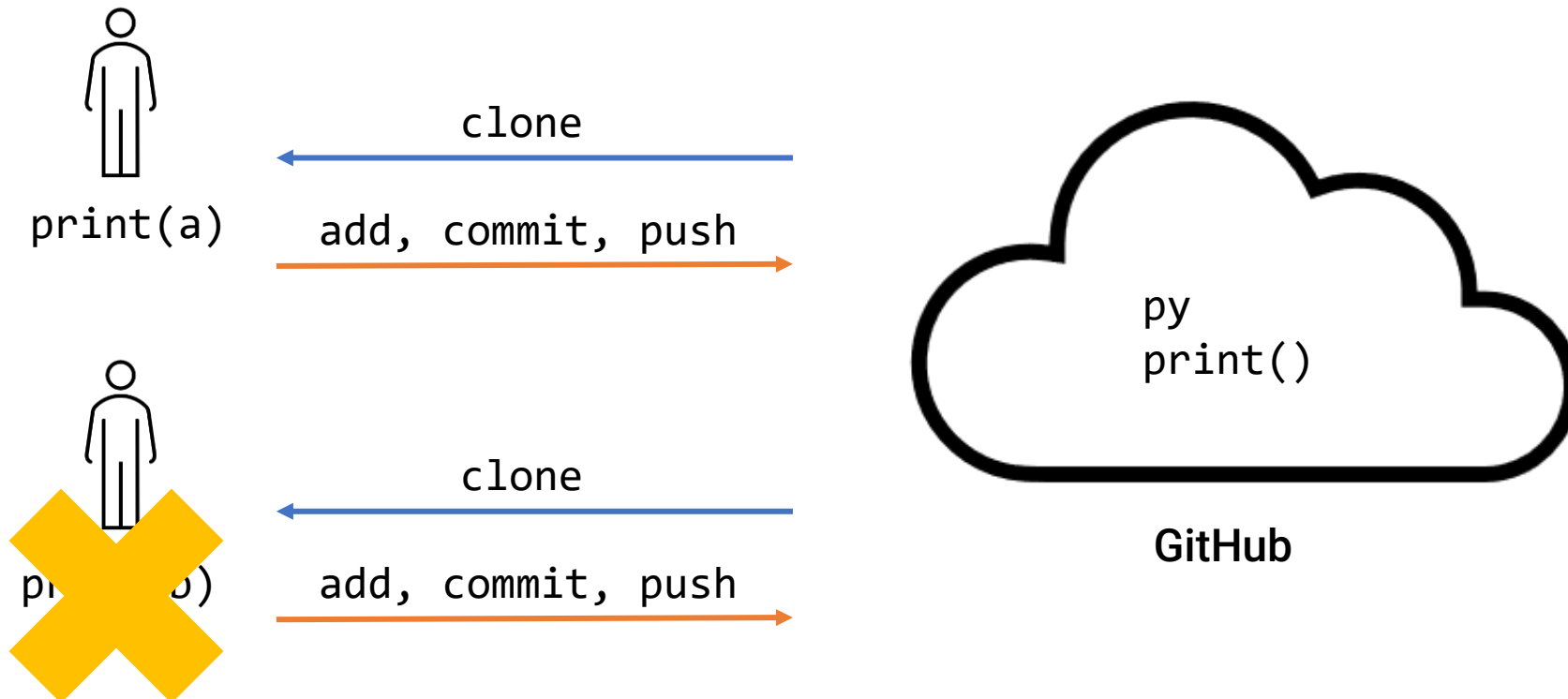
Stwórz 3 branche –  
zobacz, jak to wygląda w  
pycharmie

04



# 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?



GitHub


...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



## Dodatkowe komendy

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

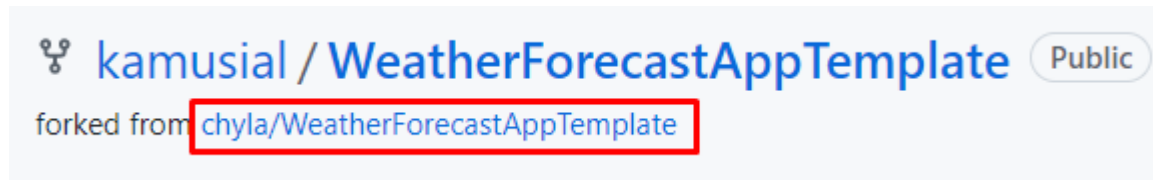
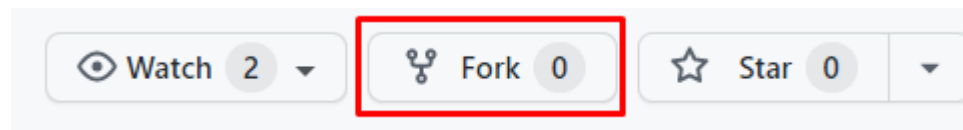


# Pull request

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”

Create pull request

 base repository: kamusial/first\_repo ▼ base: main ▼ ← head repository: kamiljakisurzytkownik/first\_repo ▼ compare: main ▼

✓ **Able to merge.** These branches can be automatically merged.

# Wspólna praca

Podział na pokoje (MS Teams) lub podział samodzielny

01

Słuchacze „forkują” repozytorium prowadzącego:  
[https://github.com/nkniat/git\\_python\\_cwiczenia](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



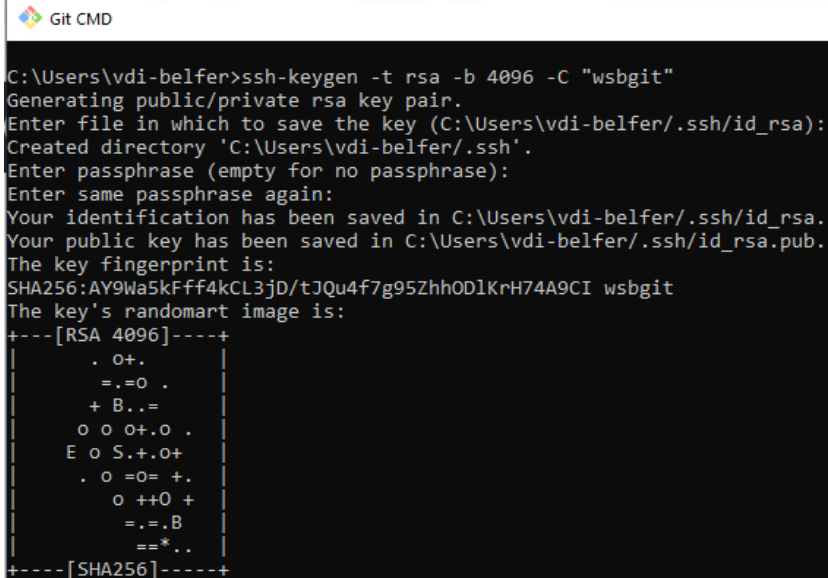
Klucz SSH

1. Uruchom konsolę Git Bash

1. Wpisz komendę:

```
ssh-keygen -t rsa -b 4096 -C
```

```
"tuoj_email@demona.com"
```



```
Git CMD
C:\Users\vdi-belfer>ssh-keygen -t rsa -b 4096 -C "wsbgit"
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\vdi-belfer/.ssh/id_rsa):
Created directory 'C:\Users\vdi-belfer/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\vdi-belfer/.ssh/id_rsa.
Your public key has been saved in C:\Users\vdi-belfer/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:AY9Wa5kFff4kCL3jD/tJQu4f7g95ZhHODlKrH74A9CI wsbgit
The key's randomart image is:
+---[RSA 4096]-----+
|  . o + .                |
|  = . = 0 .              |
|  + B . =                |
|  o o o + . o .          |
| E o S . + . o +         |
|  . o = 0 = + .          |
|  o ++ 0 +               |
|  = . . B                |
|  = * . .                |
+---[SHA256]-----+
```

# Git

## Generowanie klucza



Klucz SSH

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



Klucz SSH

1. Dodaj klucz do agenta SSH
  1. Uruchom agenta (jeśli nie jest włączony):  
**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 /I "\"\" >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



Klucz SSH

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.

### SSH keys

New SSH key

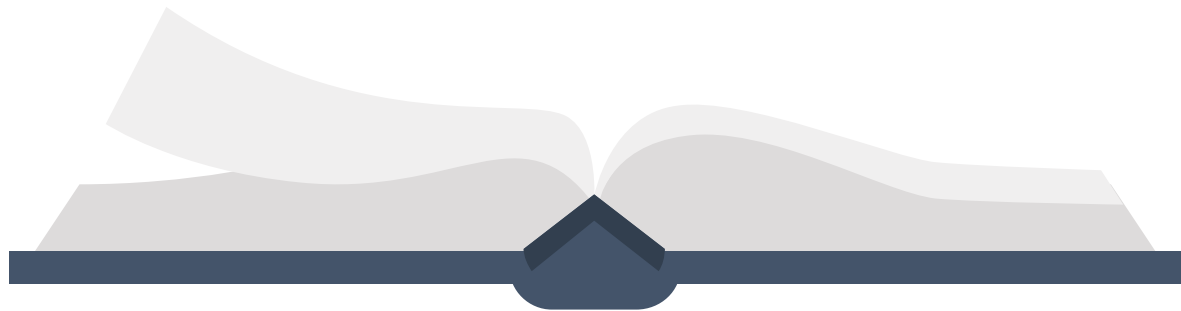
There are no SSH keys associated with your account.

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH problems](#).

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

