

PODSTAWY PROGRAMOWANIA W PYTHON

Dzień 1



AGENDA

Część pierwsza – podstawy programowania – spotkania 1-8

- Podstawowe pojęcia, pamięć,
- Instrukcje warunkowe, pętle
- Kolekcje

Część druga – wstęp do obiektowości – spotkania 9-15

- obiektowość
- Klasy, hierarchia klas, dziedziczenie
- Praktyczne wykorzystanie



Przemysław Lalak



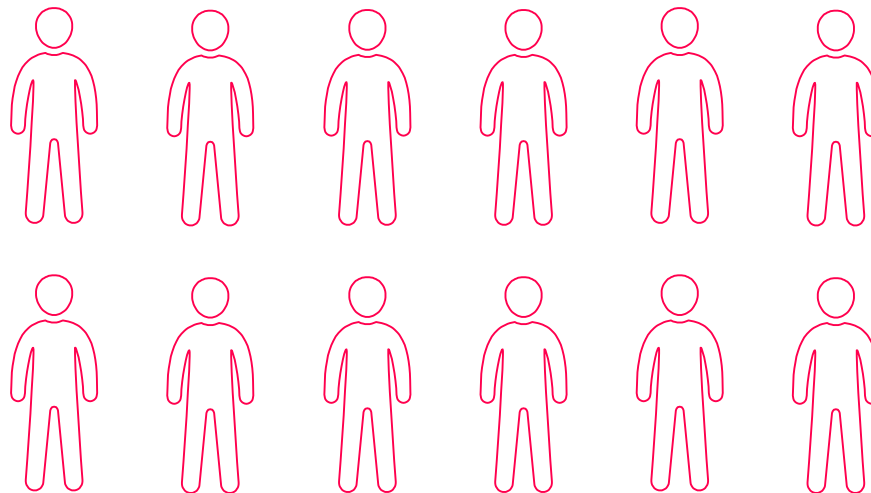
lalak@protonmail.com



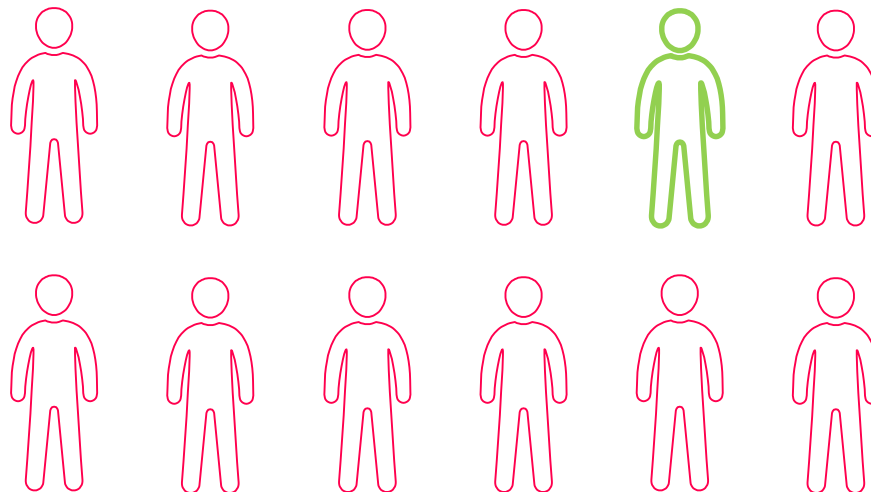
github.com/stemaan

Slajdy: Arek Gutkowski

POZNAJMY SIĘ



POZNAJMY SIĘ



TWOJA DROGA

Nie liczy się to, jak daleko będziesz
w stosunku do innych,
ale to, jak daleko znajdziesz się
za 7 tygodni, w stosunku do
siebie
z dnia dzisiejszego.

CEL KURSU

- Fundamentalne pojęcia
- Analiza problemów
- Dobre praktyki
- Składnia języka



MINDSET

ZASOBY

▪ Google

- Dokumentacja Python: docs.python.org/3/
- StackOverflow
- 4programmers: 4programmers.net/Forum
- Slack: [**isapythonkrk3.slack.com**](https://isapythonkrk3.slack.com)
- GitHub: <https://github.com/stemaan/isapythonkrk3>

AGENDA

DAY 1

- Myślenie algorytmiczne
- Input, algorytm, output
- Pseudokod
- Języki kompilowane vs interpretowane
- Python
- Terminal / wiersz poleceń
- Git / GitHub

1.

Myślenie algorytmiczne

input -> algorytm -> output

SYSTEM BINARNY

0, 1

SYSTEM DZIESIĘTNY

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

1 2 3

100

10

1

1

2

3

100

10

1

1

2

3

$100 \times 1 + 10 \times 2 + 1 \times 3$

100		10		1
1		2		3
100	+	20	+	3

4

2

1

0

0

0

4

2

1

0

0

1

4

2

1

0 1 0

4

2

1

0

1

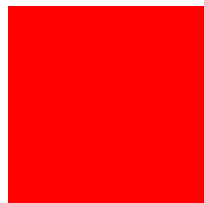
1



ASCII

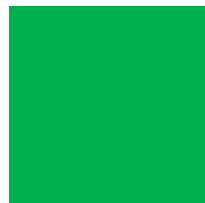
A	B	C	D	E	F	G	H	I	J	K	L	M
65	66	67	68	69	70	71	72	73	74	75	76	77
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
78	79	80	81	82	83	84	85	86	87	88	89	90

i	S	A
105	83	65



105

R



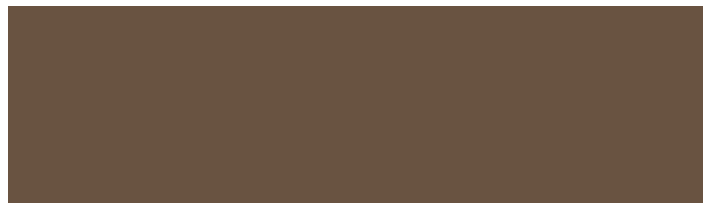
83

G



65

B



105

R

83

G

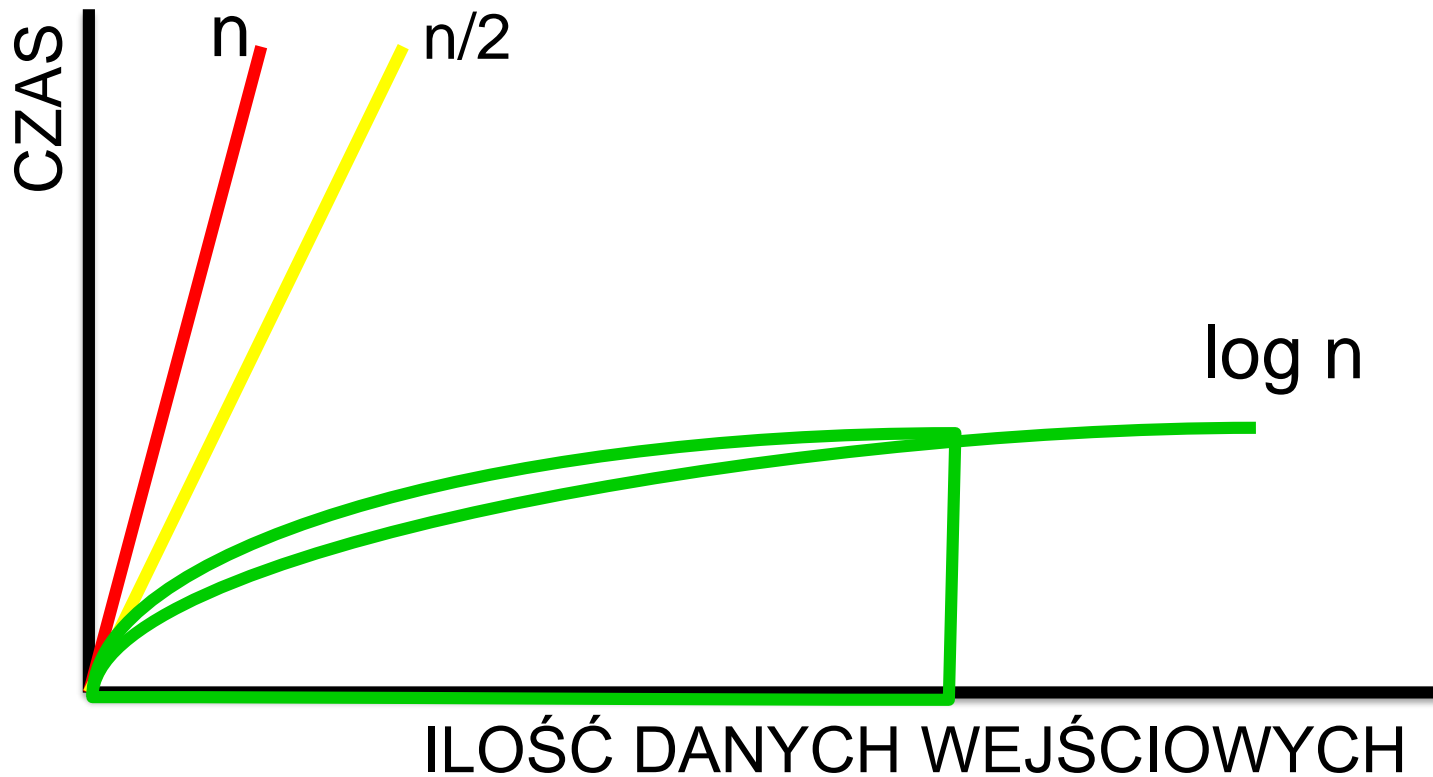
65

B

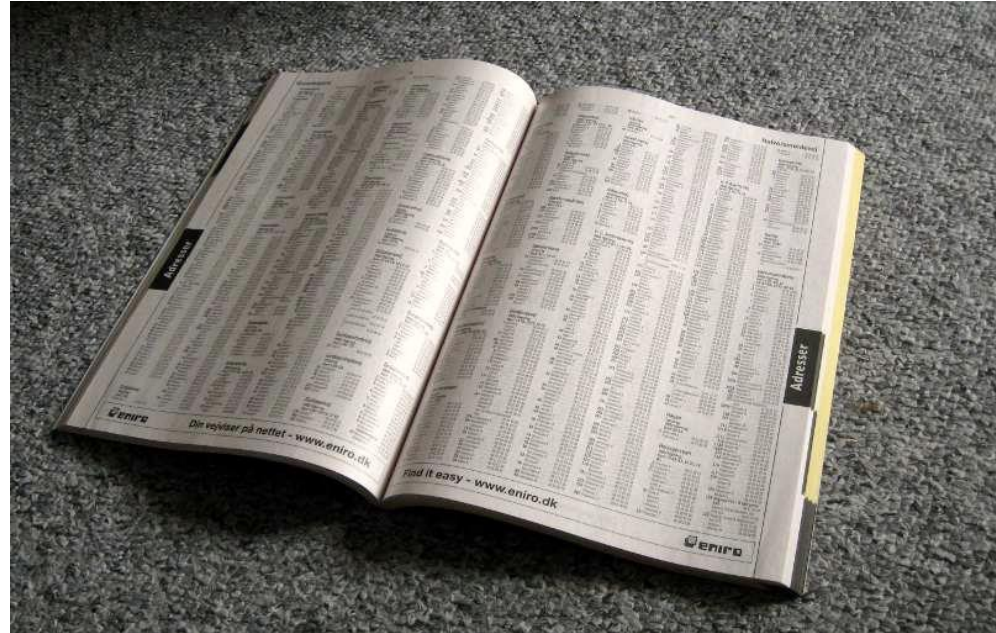
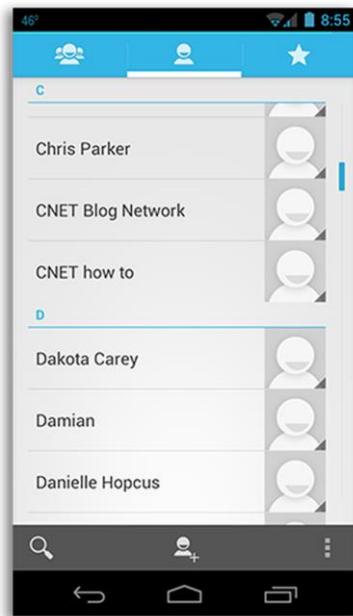
ALGORITHM

“ *Skończony zestaw instrukcji
potrzebnych do wykonania
zadania.*

ZŁOŻONOŚĆ ALGORYTMICZNA



ALGORYTM



PSEUDOKOD

PSEUDOKOD

1. weź książkę telefoniczną
2. otwórz książkę na środku
3. zobacz nazwiska
4. jeśli "Wojtkowiak" jest wśród osób
5. zadzwoń do niego
6. w przeciwnym razie jeśli "Wojtkowiak" jest wcześniej w książce
7. otwórz lewą połowę po środku
8. Idź do kroku 3
9. w przeciwnym razie jeśli "Wojtkowiak" jest później w książce
10. otwórz prawą połowę po środku
11. idź do kroku 3
12. w przeciwnym razie
13. poddaj się

Polecenia

1. **weź książkę telefoniczną**
2. **otwórz książkę na środku**
3. **zobacz nazwiska**
4. jeśli "Wojtkowiak" jest wśród osób
5. **zadzwoń do niego**
6. w przeciwnym razie jeśli "Wojtkowiak" jest wcześniej w książce
7. **otwórz lewą połowę po środku**
8. Idź do kroku 3
9. w przeciwnym razie jeśli "Wojtkowiak" jest później w książce
10. **otwórz prawą połowę po środku**
11. idź do kroku 3
12. w przeciwnym razie
13. **podдай się**

Instrukcje warunkowe

1. weź książkę telefoniczną
2. otwórz książkę na środku
3. zobacz nazwiska
4. **jeśli "Wojtkowiak" jest wśród osób**
5. zadzwoń do niego
6. **w przeciwnym razie jeśli " Wojtkowiak" jest wcześniej w książce**
7. otwórz lewą połowę po środku
8. Idź do kroku 3
9. **w przeciwnym razie jeśli " Wojtkowiak" jest później w książce**
10. otwórz prawą połowę po środku
11. idź do kroku 3
12. **w przeciwnym razie**
13. poddaj się

Pętle

1. weź książkę telefoniczną
2. otwórz książkę na środku
3. zobacz nazwiska
4. jeśli "Wojtkowiak" jest wśród osób
5. zadzwoń do niego
6. w przeciwnym razie jeśli "Wojtkowiak" jest wcześniej w książce
7. otwórz lewą połowę po środku
8. **Idź do kroku 3**
9. w przeciwnym razie jeśli "Wojtkowiak" jest później w książce
10. otwórz prawą połowę po środku
11. **idź do kroku 3**
12. w przeciwnym razie
13. poddaj się

PYTHON



DLACZEGO PYTHON

- Prosta składnia (syntax)
- Kompaktowy kod
- Kod niezależny od systemu
- Wszechstronny
- Popularność

Języki kompilowane vs interpretowane

KOMPILOWANE

- Cały program jest kompilowany
- Z reguły szybszy
- Poprawka błędu wymaga ponownej kompilacji
- Niektóre błędy możliwe do wykrycia już na etapie kompilacji

INTERPRETOWANE

- Interpretowana jest linijka po linijce
- Z reguły wolniejszy
- Łatwiejszy w obsłudze
- Prostsza składnia
- Błędy dopiero podczas uruchomienia kodu

INTERPRETER PYTHON

```
C:\Users\ArkadioG>python
Python 3.6.2 (v3.6.2:5fd33b5, Jul 8 2017, 04:57:36) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>> _
```



INTERPRETER PYTHON

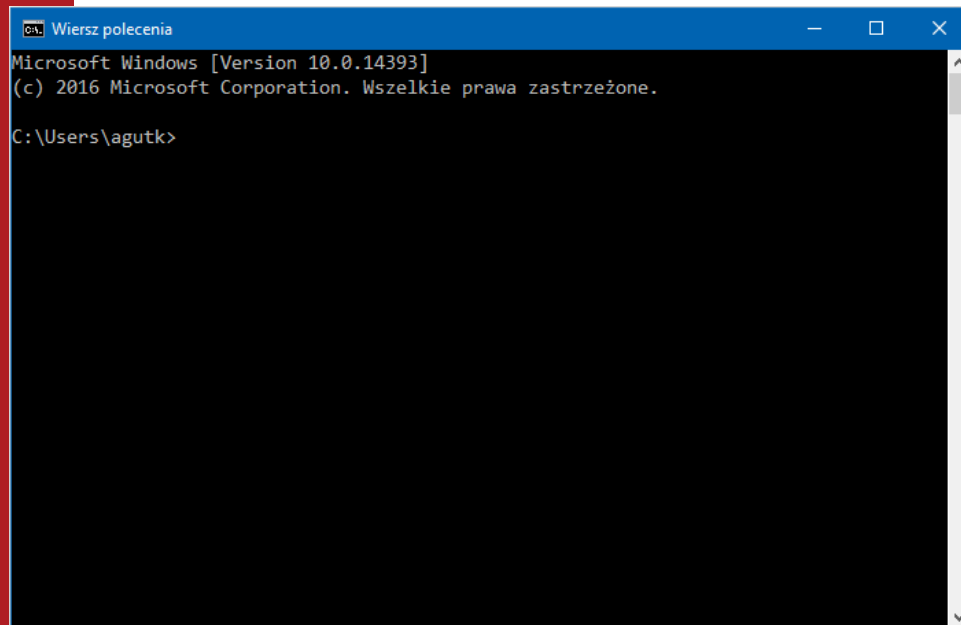
- uruchamianie

w wierszu poleceń wpisujemy **python** lub **python3.6**
i naciskamy enter

- wychodzenie

wpisujemy **exit()** i naciskamy enter

Wiersz poleceń / terminal



A screenshot of a Windows Command Prompt window. The title bar is blue and contains the text 'Wiersz polecenia' followed by standard window control buttons (minimize, maximize, close). The main area has a black background with white text. The text displayed is: 'Microsoft Windows [Version 10.0.14393]', '(c) 2016 Microsoft Corporation. Wszelkie prawa zastrzeżone.', and 'C:\Users\agutk>'. A vertical scrollbar is visible on the right side of the window.

```
Wiersz polecenia
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. Wszelkie prawa zastrzeżone.
C:\Users\agutk>
```

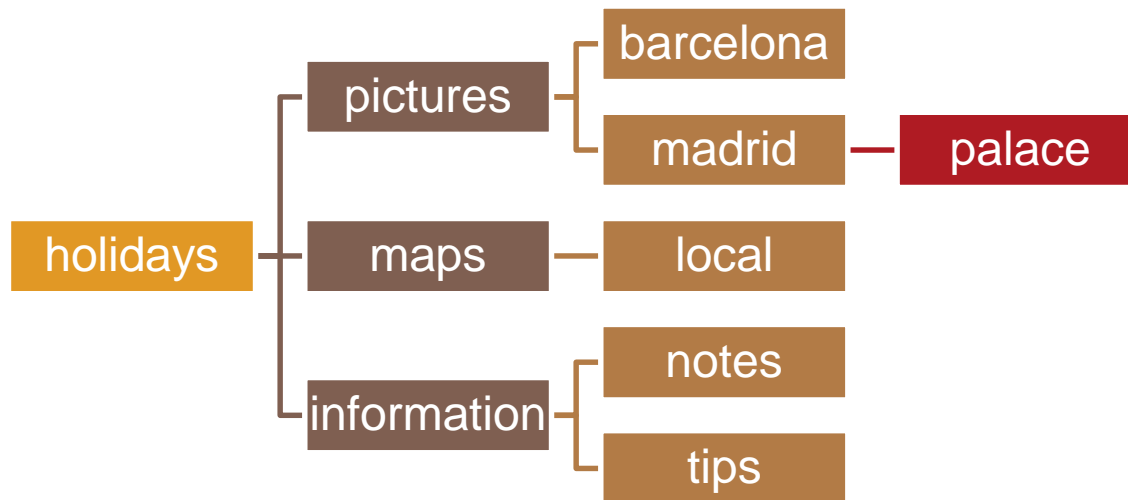

Cheatsheets

- <http://bit.ly/mac-ter>
- <http://bit.ly/wincommands>

Wiersz poleceń

/? –help -- help --h - pomoc
mkdir / md – tworzenie folderu
rmdir / rd – usuwanie folderu
move – przenoszenie zmiana nazwy
dir / ls – wyświetlenie zawartości folderu
cd – przejście do innej lokalizacji
pwd – obecny folder (linux, osx, powershell)
type / cat – wyświetlenie zawartości pliku
touch – utworzenie pliku
echo Lalalalla > plik.txt

WIERSZ POLECEŃ



1. Utwórz strukturę folderów
1. Zmień nazwę folderu barcelona na valencia
2. Usuń folder information
3. Będąc w folderze pictures dodaj folder barcelona do folderu maps
4. Będąc w folderze maps wyświetl zawartość folderu madrid

Git & GitHub

Git



Autor: Linus Torvalds

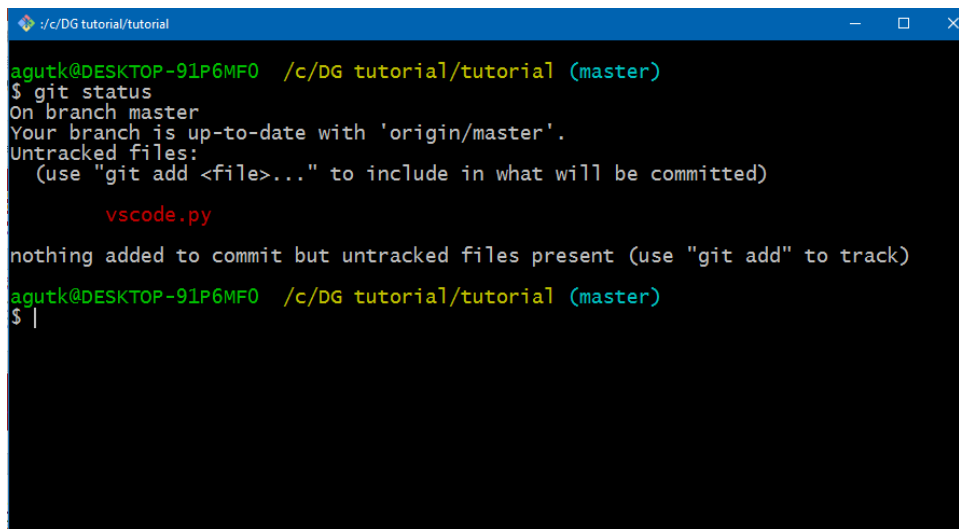
Rozproszony system wersjonowania plików

- każdy developer może pracować nad częścią kodu
- dev może mieć kilka wersji kodu
- umożliwia cofanie zmian, łączenie gałęzi (branch)

<https://git-scm.com/>

Git - konsola

wydajemy polecenia konsolą (wymagana instalacja git na Windows)



```
agutk@DESKTOP-91P6MF0 /c/DG tutorial/tutorial (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        vscode.py

nothing added to commit but untracked files present (use "git add" to track)
agutk@DESKTOP-91P6MF0 /c/DG tutorial/tutorial (master)
$ |
```

ściąga: <https://services.github.com/on-demand/downloads/github-git-cheat-sheet.pdf>

PODSTAWOWE KOMENDY GIT

- `git clone https://...` - skopiowanie(klonowanie) zdalnego repo na komputer
- **git pull** - pobranie zmian ze zdalnego repo
- `git init` – zainicjowanie nowego lokalnego repozytorium
- `git add` – dodanie plików do wersjonowania
- `git commit` – zapamiętanie lokalnego stanu
- `git push` – wysłanie commita na zdalne repo
- `git status`

Git

Dokumentacja, video, książka Git Pro

<https://git-scm.com/doc>

Książka w wersji polskiej (1 edycja):

<https://git-scm.com/book/pl/v1>

Online tutorial z komend:

<https://try.github.io/levels/1/challenges/1>



GitHub

<https://github.com>

- repozytoria kodu w chmurze
- bezpłatne publiczne repozytoria
- najpopularniejsze miejsce z projektami opensource
- must-have dla programisty

Konkurencja – gitlab, bitbucket – dają bezpłatne prywatne repozytoria



GitHub

- zakładamy konto
- tworzymy repozytorium
- klonujemy na swój komputer
- zmieniamy kod
- commitujemy zmiany (zapisujemy do lokalnego repo)
- synchronizujemy z github

Polecenia w wierszu komend

Pliki ignorowane:

<https://help.github.com/articles/ignoring-files/>



GitHub

Kurs: <https://services.github.com/on-demand/>

Cheatsheet: <https://services.github.com/on-demand/downloads/github-git-cheat-sheet.pdf>

Pomoc: <https://help.github.com/>

Video: <https://www.youtube.com/githubguides> ,
<https://youtu.be/HVsySz-h9r4>

<https://services.github.com/resources/>



GitHub Desktop

<https://desktop.github.com/>

alternatywne programy okienkowe:

<https://git-scm.com/downloads/guis>

The screenshot displays the GitHub Desktop application window. The top bar includes a search icon, the repository name 'muan/sort-search-results', and buttons for 'Pull request', 'No uncommitted changes', and settings. Below this, there are buttons for 'Update from master' and 'View branch'. A 'Sync' button is located on the right. The main area shows a commit history graph with a blue dot indicating the current commit. The left sidebar lists repositories: 'atom', 'electron', 'find-and-replace' (selected), 'libgit2', 'libgit2sharp', 'mojibar', and 'octokit.net'. The right sidebar shows a list of commits, including 'Use a loop' by Ben Ogle and 'Prevent rendering elements indefinitely' by Mu-An Chiou.

Filter repositories

atom

electron

find-and-replace

libgit2

libgit2sharp

mojibar

octokit.net

muan/sort-search-results

Pull request

No uncommitted changes

Update from master

View branch

Sync

master

muan/sort-...ch-results

Use a loop
9 hours ago by Ben Ogle

Use .localeCompare instead of >...
9 hours ago by Mu-An Chiou

Default out of range so results won't...
9 hours ago by Mu-An Chiou

Prevent rendering elements indefinitely

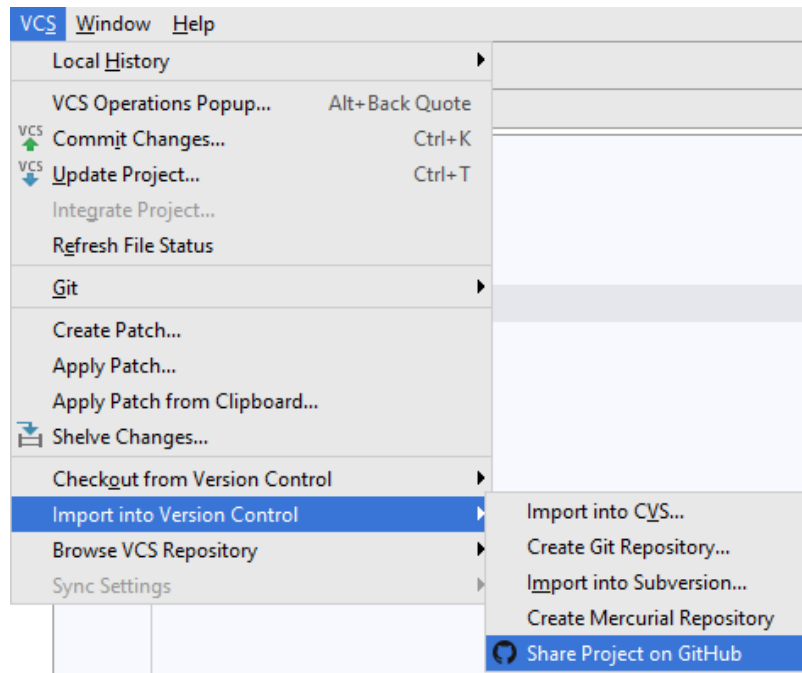
Mu-An Chiou f9848ba

GitHub Revert Collapse all

By making a mark when screen is filled with results and only add more results to above the fold if the insertPoint is above the fold.

GitHub & PyCharm

- tworzymy projekt
- w ustawieniach łączymy się z github (login, hasło)
- menu VCS:



PODSUMOWANIE

- Myślenie algorytmiczne
- Input, algorytm, output
- Pseudokod
- Języki kompilowane vs interpretowane
- Python
- Terminal / wiersz poleceń
- Git/GitHub



Thanks!!