Xebia

WHERE EXPERTS GROW

Xebia

Arkadiusz Gutkowski
Tech Lead, Python Dev

Next Level

kilka słów o tym jak wejść na kolejny poziom umiejętności

1. O prowadzącym
2. Programowanie to przygoda
3. 10 błędów początkujących
4. Tips & Tricks
5. Q&A

O prowadzącym

Arkadiusz Gutkowski

Senior Developer, Tech Lead w **Xebia Poland**

- **Role**: Senior Python Developer, Tech Lead, Team Lead
- **Języki**: Python, C#, C
- **Obszary**: Chmura, Web, Blockchain, DevOps
- **Projekty**: PCI, Blockchain, NFC, IOT, CAD
- **Społeczność**: PyGda, Django Girls, Infoshare Academy

arkadiuszgutkowski

xebia.com

Nasza Historia

- W Polsce działamy od 2005 roku.
  Przez 17 lat jako PGS Software.
- W 2021 roku dołączyliśmy do grupy **Xebia** i od 2023 działamy pod tą marką.
- Pracujemy zdalnie oraz mamy biura we Wrocławiu, Gdańsku, Rzeszowie i Warszawie.
- **Mamy ponad 1000 osób na pokładzie!**

O nas

Ponad 1000 specjalistów

Zespoły Scrumowe zawierające:

- full-stacków,
- front-endowców,
- back-endowców,
- testerów,
- UX/UI designerów,
- analityków biznesowych
- kierowników projektów.

2947 projektów

Posiadamy klientów głównie z Europy i USA.

3 oddziały

Mamy biura w 3 lokalizacjach: Wrocław, Gdańsk i Rzeszów.

Część osób pracuje 100% zdalnie.

Posiadamy też biuro co-workingowe w Warszawie.

Poszerzamy naszą wiedzę

Budżet rozwojowy do 6800 zł:

Szkolenia zewnętrzne

Kursy językowe

Szkolenia wewnętrzne

- Upskill (Szkolenia techniczne)
- Szkolenia biznesowe i liderskie

Certyfikaty

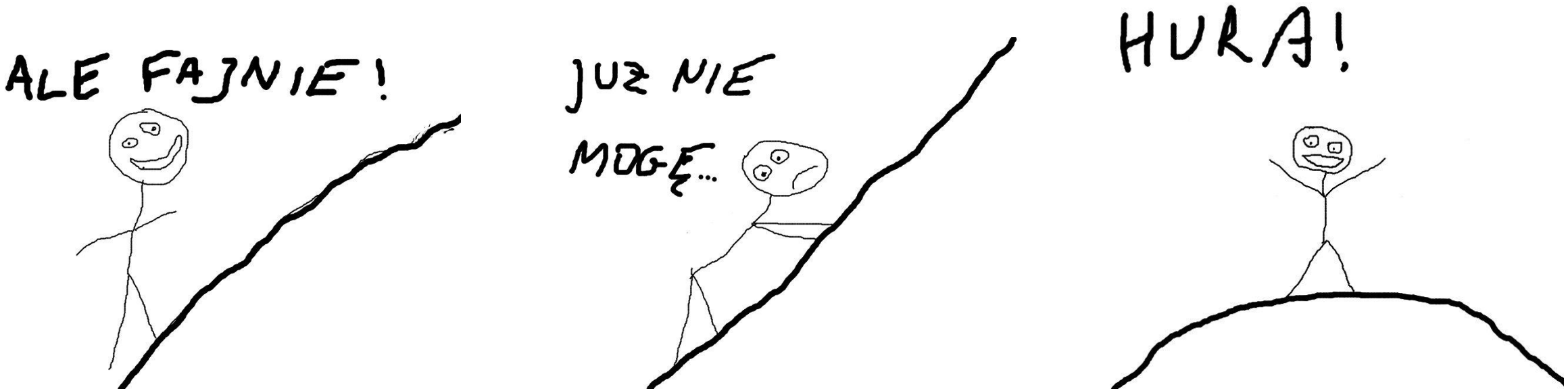
m.in.:

- AWS
- Azure
- ISTQB

Wyjazdy na konferencje

10 najczęstszych błędów początkujących

Programowanie to przygoda!



0. Składnia języka

```
if is_delivered and is_notified:
    is_done = True
else:
    is_done = False
```

```
is_done = is_delivered and is_notified
```

1. Argumenty domyślne

```
def foo(bar=[]):
    bar.append('baz')
    return bar
```

```
['baz']
['baz', 'baz']
['baz', 'baz', 'baz']
```

```
def foo(bar=None):
    if bar is None:
        bar = []
    bar.append('baz')
    return bar
```

```
['baz']
['baz']
['baz']
```

2. Pola klas

```
class A(object):
    x = 1

class B(A):
    pass

class C(A):
    pass
```

```
print(A.x, B.x, C.x)
>>> 1 1 1

B.x = 2
print(A.x, B.x, C.x)
>>> 1 2 1

A.x = 3
print(A.x, B.x, C.x)
>>> 3 2 3
```

3. Zakres zmiennych

```
x = 10

def foo():
    print(x)

foo()
```

```
>>> 10
```

```
x = 10

def foo():
    x += 1
    print(x)

foo()
```

```
Traceback (most recent call last):
…
UnboundLocalError: local variable 'x' referenced before assignment
```

4. modyfikacja list podczas iteracji

```
numbers = list(range(10))

for i in range(len(numbers)):
    if numbers[i] % 2 == 0:
        del numbers[i]
```

```
Traceback (most recent call last):
…
    if numbers[i] % 2 == 0:
IndexError: list index out of range
```

```
numbers = list(range(10))

for i, num in enumerate(numbers):
    if num % 2 == 0:
        del numbers[i]
```

```
>>> [1, 3, 5, 7, 9]
```

5. Cykliczne zależności modułów

```python
import b

def f():
    return b.x

print(f())
```

a.py

```python
import a

x = 1

def g():
    print(a.f())
```

b.py

```
>>> import a
1
>>> import b

>>>
```

```
>>> import b

Traceback (most recent call last):
…
  import a
…
AttributeError: module 'b' has no attribute 'x'
```

6. Konflikty nazw

Xebia

email.py

```
import smtplib

me = 'me@example.com'
pwd = 'supersecretpassword1'

mailer = smtplib.SMTP('smtp.gmail.com', 587)
mailer.ehlo()
mailer.starttls()
mailer.login(me, pwd)
mailer.sendmail(me, 'friend@example.com', 'Hello')

mailer.close()
```

```
Traceback (most recent call last):
…
   import smtplib
…
   import email.utils
…
mailer = smtplib.SMTP('smtp.gmail.com', 587)
AttributeError: module 'smtplib' has no attribute
'SMTP'
```

7. Lambda w pętlach

```
squares = []
    for x in range(5):
        squares.append(lambda: x**2)

for a in squares:
    print(a())
```

```
16
16
16
16
16
```

```
squares = []
for x in range(5):
squares.append(lambda n=x: n**2)

for a in squares:
print(a())
```

```
0
1
4
9
16
```

8. MRO

```
class Animal(object):
    def __init__(self, name):
        self.nazwa = name


class Horse(Animal):
    def __init__(self, name, color):
        self.color = color
        Animal.__init__(self, name)


class Donkey(Animal):
    def __init__(self, name):
        super().__init__(name)


class Mule(Donkey, Horse):
    def __init__(self, name):
        super().__init__(name)


emule = Mule('Jonny')
```

TypeError: __init__() missing 1 required positional argument: 'color'

9. Wielowątkowość jest prosta?

```
import threading

counter = 0

def worker():
    'My job is to increment the counter and print the current count'
    global counter

    counter += 1
    print('The count is %d' % counter)
    print('---------------')

print('Starting up')
for i in range(10):
    threading.Thread(target=worker).start()
print('Finishing up')
```

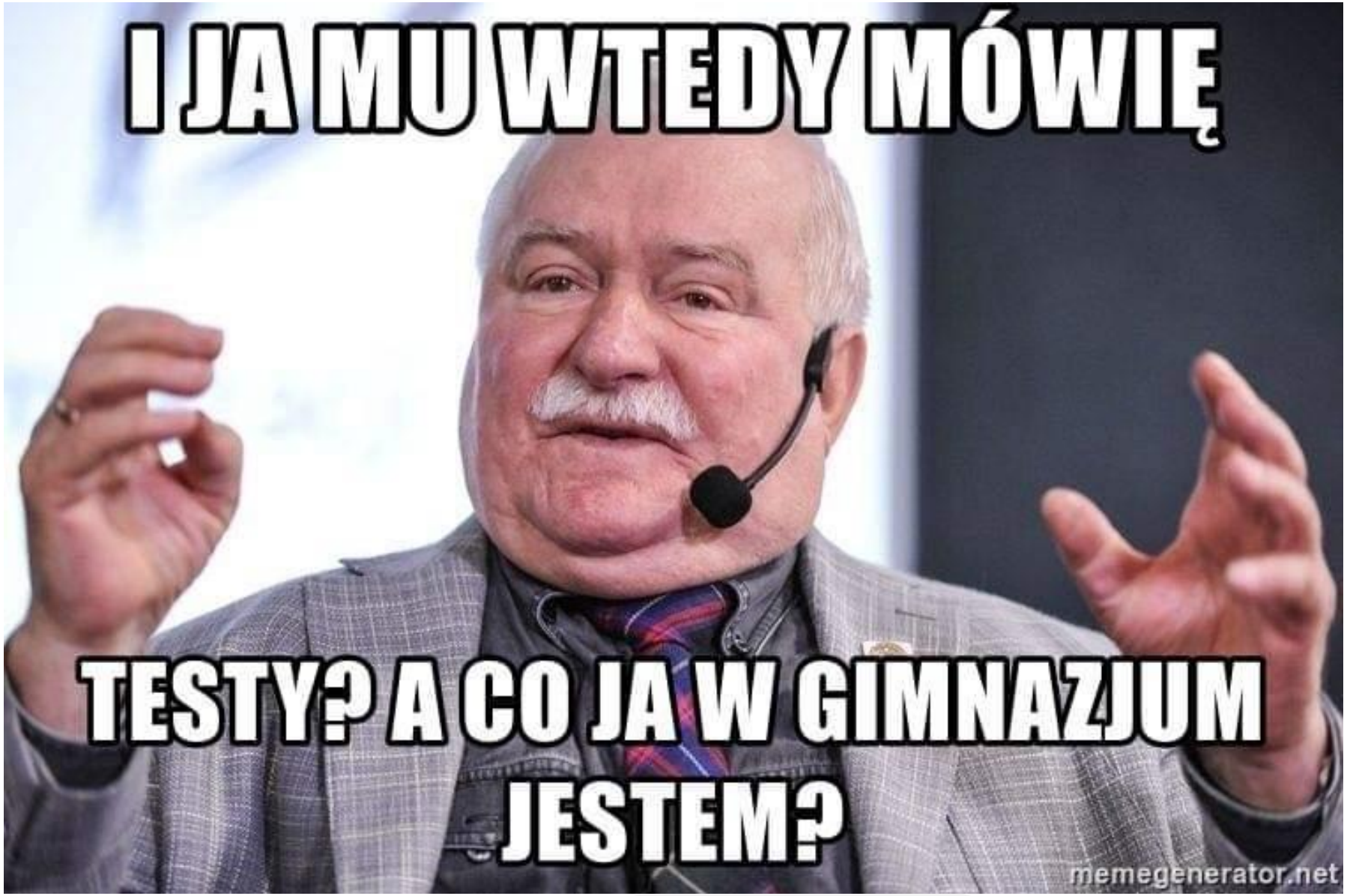
Tips & Tricks

Poznaj język

- czy w pełni wykorzystuję składnię?
- czy znam możliwości biblioteki standardowej?
- czy znam narzędzia wspomagające?
- czy znam niskopoziomowe elementy?

Przewiduj problemy

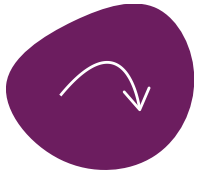
- http request - response
- kolizje losowo generowanych wartości
- zmiany w kodzie
- skala aplikacji
- nazwy wprowadzające w błąd
- **testy na pomoc…**


I JA MU WTEDY MÓWIĘ
TESTY? A CO JA W GIMNAZJUM JESTEM?
memegenerator.net

Xebia

Dobry kod to kod, który łatwo się testuje.

Ja, codziennie



Xebia

How the UML diagram describes the software

How the code is actually written

Organizacja i projekt systemu

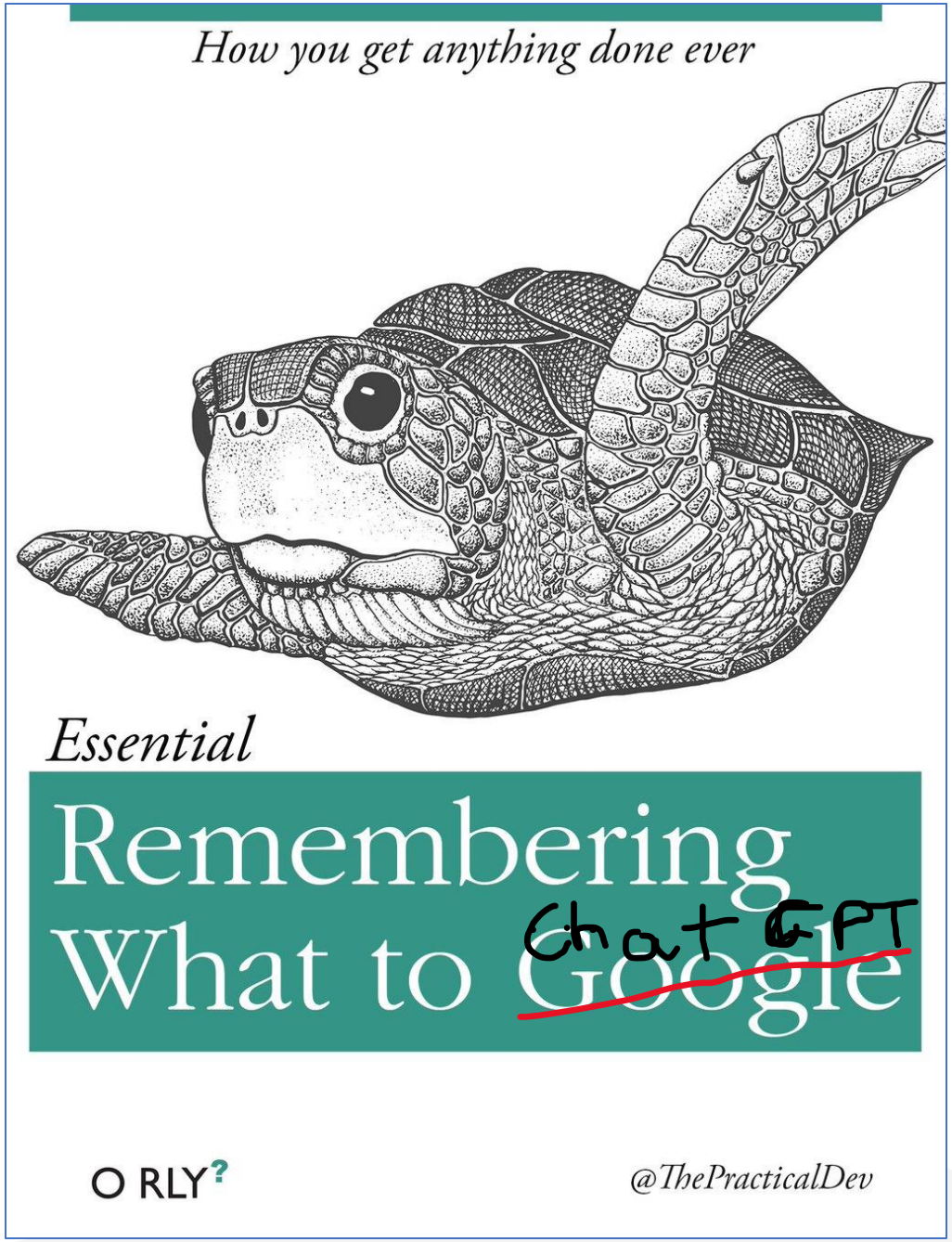
- w jakim celu istnieje ta funkcja?
- jaka jest odpowiedzialność tego kodu?
- czy warto kod podzielić na moduły?
- jakich wzorców użyć?
- nie wynajduj na nowo koła, ale nie używaj wywrotki do przewiezienia wykałaczki
- myśl w przód

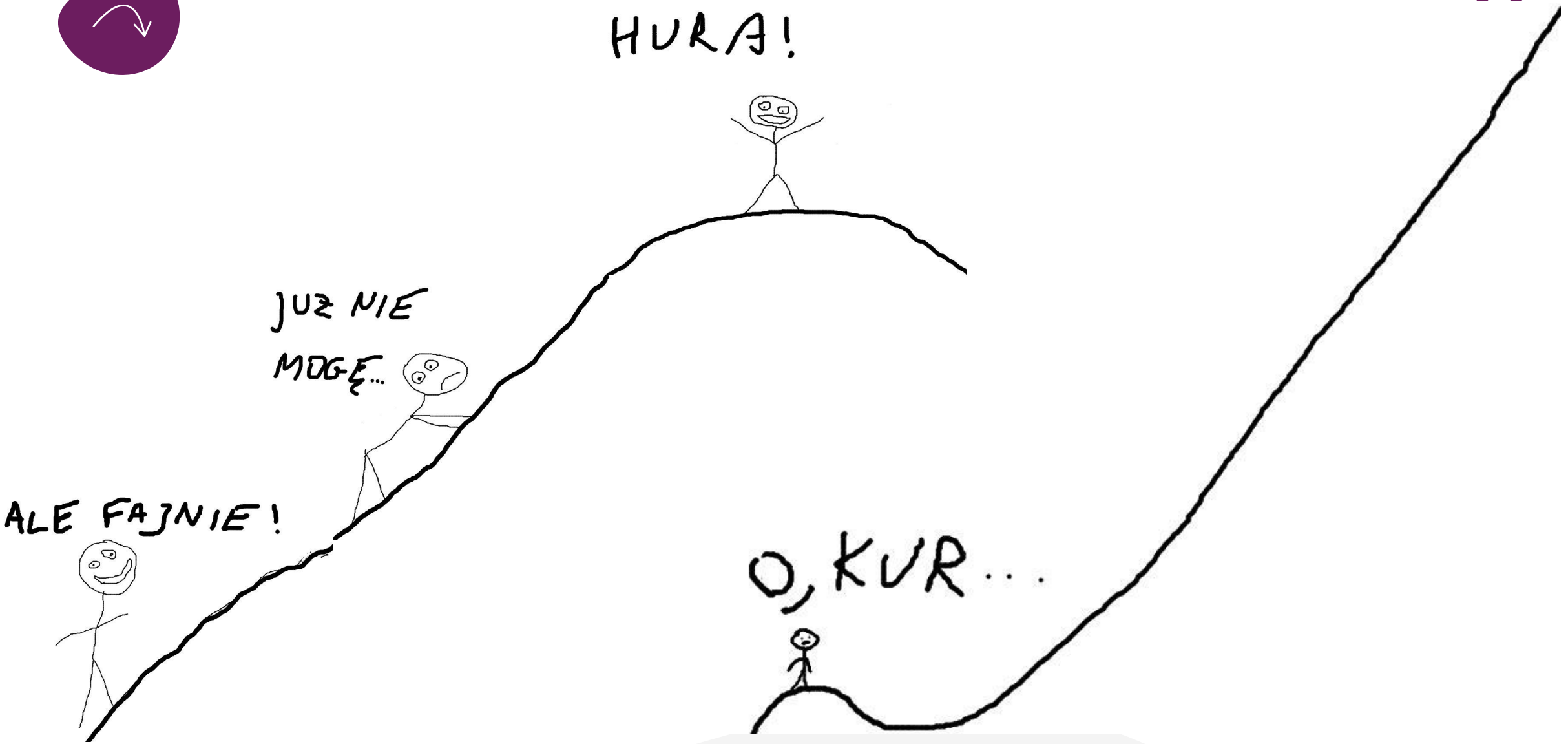
Twoja dotychczasowa praca jest Twoim najlepszym zasobem

RandomPantsAppear@Reddit

"Dobry" programista

- umiej przyjmować (i dawać) krytykę
- dyskutuj, zadawaj pytania
- udzielaj się, szukaj nowych źródeł
- "mądry kod" powinien być czytelny
- nie, nie będziesz pamiętać, //todo i komentarze Cię uratują
- nie ufaj inputom użytkownika


How you get anything done ever
Essential
Remembering
What to Google
ChatGPT
O RLY?
@ThePracticalDev


HURA!
JUŻ NIE MOGĘ...
ALE FAJNIE!
O, KUR...

Referencje

- https://www.reddit.com/r/learnpython/comments/aqa74u/some_lessons_from_16_years_of_development/
- https://gist.github.com/justinmeiners/be4540f515986d93ee12ac2f1980631a
- https://www.toptal.com/python/top-10-mistakes-that-python-programmers-make
- https://youtu.be/9zinZmE3Ogk

Thank you!

arkadiusz.Gutkowski@xebia.com

github.com/Xebia-Poland/pjatk-python-next-level