Jak przygotować się do rozmowy rekrutacyjnej na Python Developera

Wojciech Lichota - STX Next

Styczeń 2018

Prezentacja pokazywana na PyRa (11.2015), Python Łódź (06.2016) oraz PyGda (01.2018). Więcej o temacie na slide 4.

Wojciech Lichota

STX Next

wojciech@lichota.pl

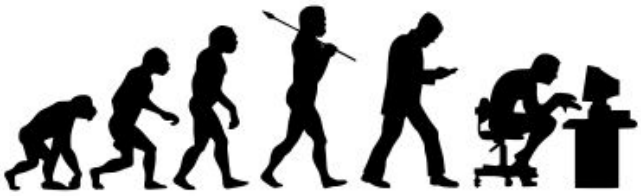
http://lichota.pl

Od ponad 10 lat pracuję w STX Next. Gdy dołączałem byłem 6 osobom a dziś jest nas już ponad 280 osób. Pierwsze 6 lat byłem developerem głównie Pythona. Następnie CTO prze 2 lata, a później na ochotnika wyjechałem do Łodzi aby otworzyć tam nowy oddział STX Next. Po 1,5 roku i zbudowania 40 osobowego zespołu przeprowadziłem się do Gdańska i otwieram kolejne biuro!

Rekrutujemy: https://stxnext.com/job-offers/city/gdansk/

1. Nie chcę reklamować STX Next i naszego, notabene świetnego, procesu rekrutacyjnego. Opowiem o pytaniach używanych przez różne firmy i agencje rekrutacyjne.
2. Chcę ułatwić młodym osobom przygotowanie do pierwszej rozmowy o pracę, pokazując zarówno przykłady jak i źródła wiedzy.
3. Prezentacja zawierać będzie także wskazówki dla bardziej doświadczonych osób, które nie były dawno na rozmowie technicznej.
4. Będę dzielił się także wskazówkami “z drugiej strony barykady”, ponieważ od wielu lat oceniam osoby na rozmowach kwalifikacyjnych.
5. Zachęcam do przerywania mi i uzupełniania moich wypowiedzi o własne doświadczenia.

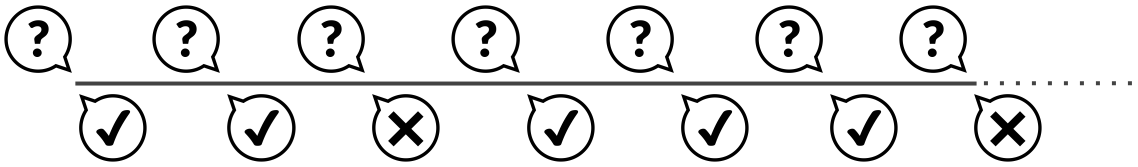
Evolution



http://www.nicolasbize.com/blog/how-i-ended-up-conducting-the-most-successful-technical-interviews-with-a-single-question/

1. Ten artykuł opowiada o tym jak autor zmieniał swoje podejście do rozmów rekrutacyjnych i jak dopracowywał swoje pytania.
2. Ja przeszedłem podobną ewolucję.
3. Są na rynku firmy na różnym etapie, przez co ciężko stwierdzić na jaką rozmowę się trafi.

Quiz



1. Rozmowa składająca się z wielu krótkich pytań.
2. Czasem zadawane przez telefon (screening), ustnie lub w formie papierowej (sic!).
3. Krok pierwszy w ewolucji.
4. Stosowane częściej na stanowiska juniorskie.
5. Osobiście nie lubię osobiście takich pytań, ponieważ w dużej części testują one pamięć kandydata a nie jego wiedzę czy zdolności.
6. Osobiście wolę, gdy pytanie algorytmiczne (patrz slide 13+) były tak skonstruowane aby można było użyć tego idiomów albo natknąć się na anty-pattern.

QUIZ

★

Logic

- ★ Masz osiem piłek oraz wagę szalkową. Siedem piłek waży tyle samo, jedna jest cięższa. Ile ważeń potrzebujesz aby być pewny, która jest tą najcięższą?
- ★ Jak byś podzielił tort o nieregularnym kształcie na dwie równe części?
- ★ Cegła waży 1kg i pół cegły. Ile waży cegła?

1. Pytania na logiczne myślenie.
2. Często zadawane w rekrutacji na stażystów.
3. W STX Next nie są używane.

QUIZ

Idioms
&
Syntactic Sugar

- Slice (`my_list[2:-1]`)
- comprehensions

  list `[n ** 2 for n in range(10) if n % 2 == 0]`
  dict `{n: n ** 2 for n in range(10)}`
  generator `(n ** 2 for n in range(10))`
- `def my_func(*args, **kwargs):`
- `a, b, c = my_tuple`
- dekoratory (`@`)
- generatory (`yield`)
- context manager (`with`)
- meta-klasy (`__metaclass__`)
- type hints

1. Pytania o idiomy Pythona (konstrukcje typowe dla języka Python, rzadko spotykane w innych językach)
2. Idiomy mają pokazać czy programujesz “pythonic way” czy Twój kod będzie bardziej przypominał C niż Pythona.
3. W pytaniach o Syntactic Sugar może chodzić o wytłumaczenie co kryje się “z tyłu”, np. jak zastąpić kod z “with” na “try-finall”
4. Nie cierpię szczególnie zadań polegających na napisaniu na kartce papieru wyniku np. list comprehension

QUIZ

Quirks

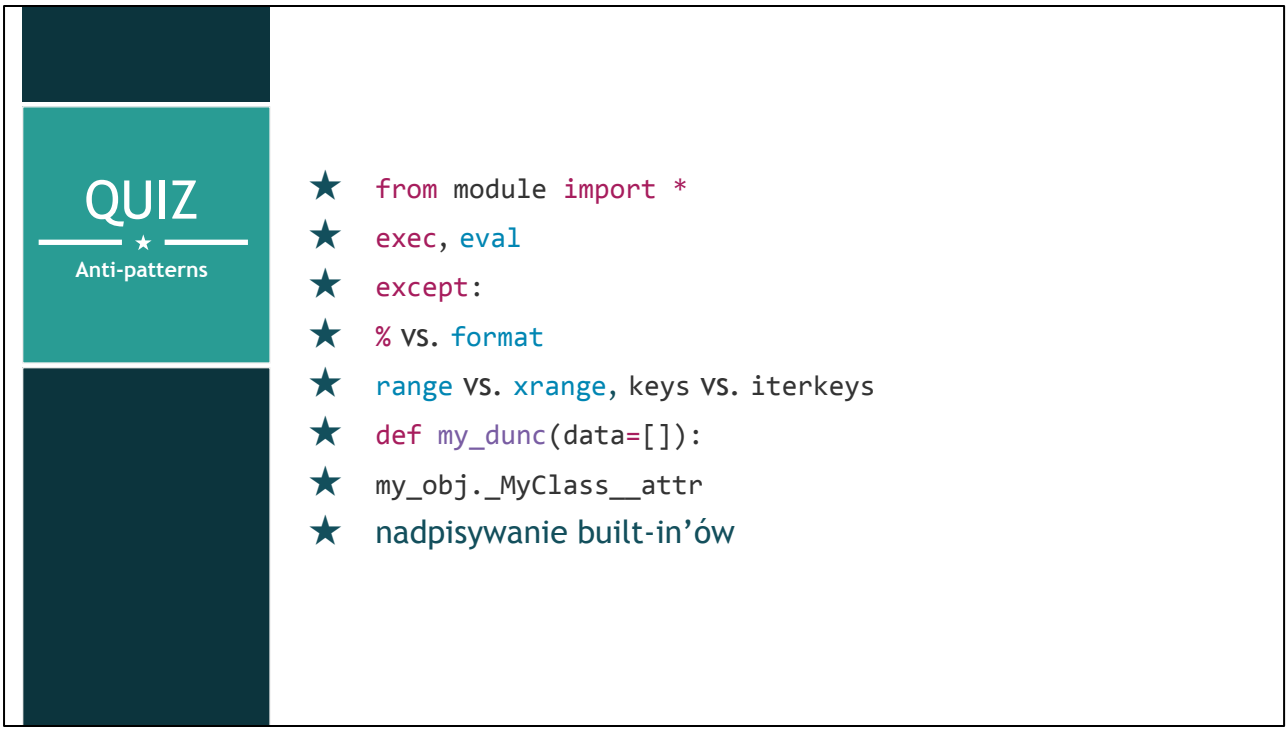
- ★ muttable types

```
a = [1, 2, 3]; b = a; b[1] = 0; print(a)
```

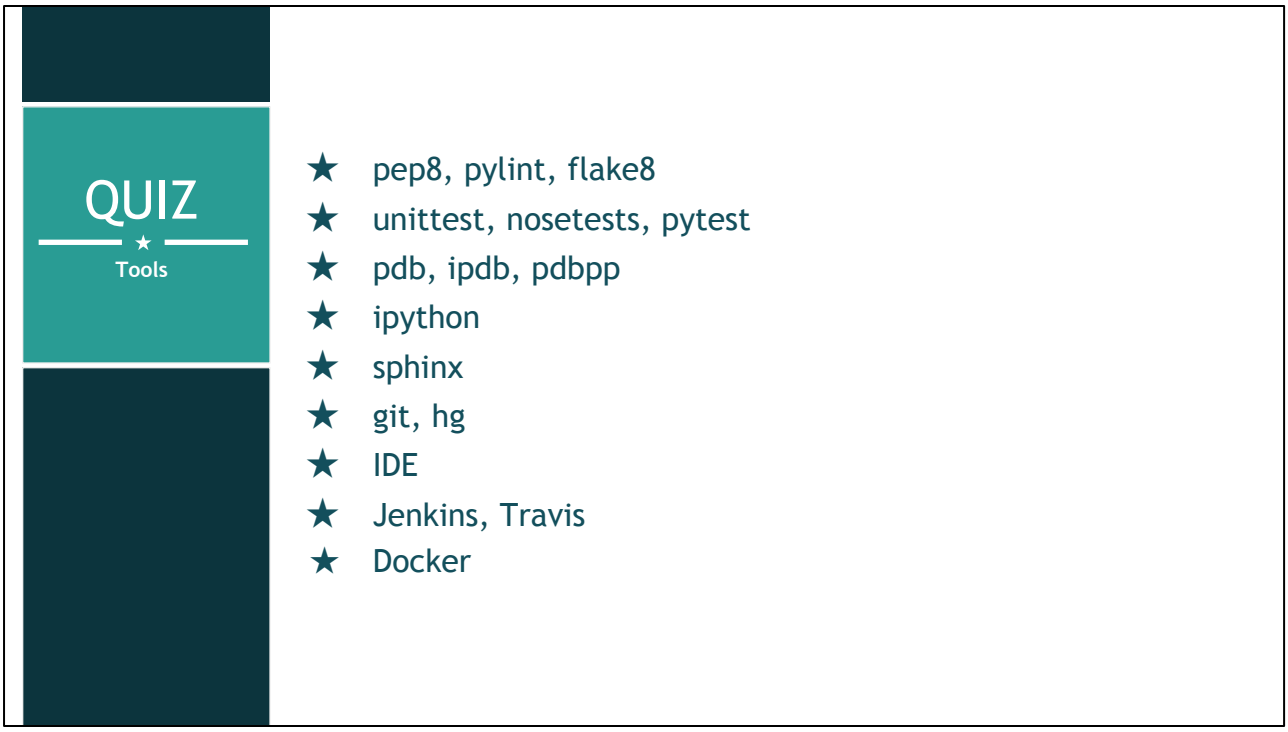
- ★ immutable types

  Co może być kluczem słownika?
- ★ `sort` vs. `sorted`
- ★ `''.join(my_list)`
- ★ `__getattr__` vs. `__getattribute__`
- ★ `_, _, _ = [1, 2, 3]`
- ★ `a, *b, c = [1, 2, 3, 4, 5]`
- ★ `a, b = b, a`
- ★ `True, False = False, True`
- ★ `..., Ellipsis`

1. “kruczki” i “dziwnostki” Pythona
2. Osobiście nie lubię, bo testują wiedzę nieprzydatną (ew. mało przydatną) w pracy



1. Anty-patterny które należy unikać w kodzie.
2. Pytanie może dotyczyć opowiedzenie dlaczego tak się nie pisze, jakie może to rodzić konsekwencje.



1. Quiz może zawierać pytania dotyczące typowych narzędzi programisty, np.:
   a. podaj komendę do tworzenia nowego branch w git
   b. RUN vs. CMD w Docker
   c. ulubiony IDE i dlaczego

- ★ dekoratory - http://stackoverflow.com/a/1594484/2342911
- ★ yield - http://stackoverflow.com/a/231855/2342911
- ★ meta-klasy - http://stackoverflow.com/a/6581949/2342911
- ★ http://stackoverflow.com/questions/101268/hidden-features-of-python
- ★ https://github.com/faif/python-patterns
- ★ książka “Writing Idiomatic Python”, Jeff Knupp
- ★ http://pythonfasterway.org/

Polecane pomoce do bloku “Quiz”

1. Zadania polegające na “rozgryzieniu” problemu i napisania kodu rozwiązującego problem.
2. Zadania najczęściej rozwiązuje się przed komputerem, rzadziej na tablicy czy kartce papieru.
3. Osobiście nie lubię pytań dotyczących problemów matematycznych, bo takie bardzo rzadko pojawiają się w codziennej pracy.
4. Wolę gdy w zadaniu chodzi i przetworzenie danych, najlepiej w którym należy użyć różnych struktur (np. list, słowników, itp.)

Przykładowe pytania algorytmiczne jakie miałem na rozmowach rekrutacyjnych - głównie w Poznaniu.

CODE

Algorithms

Pytania używane w STX Next **w przeszłości:**

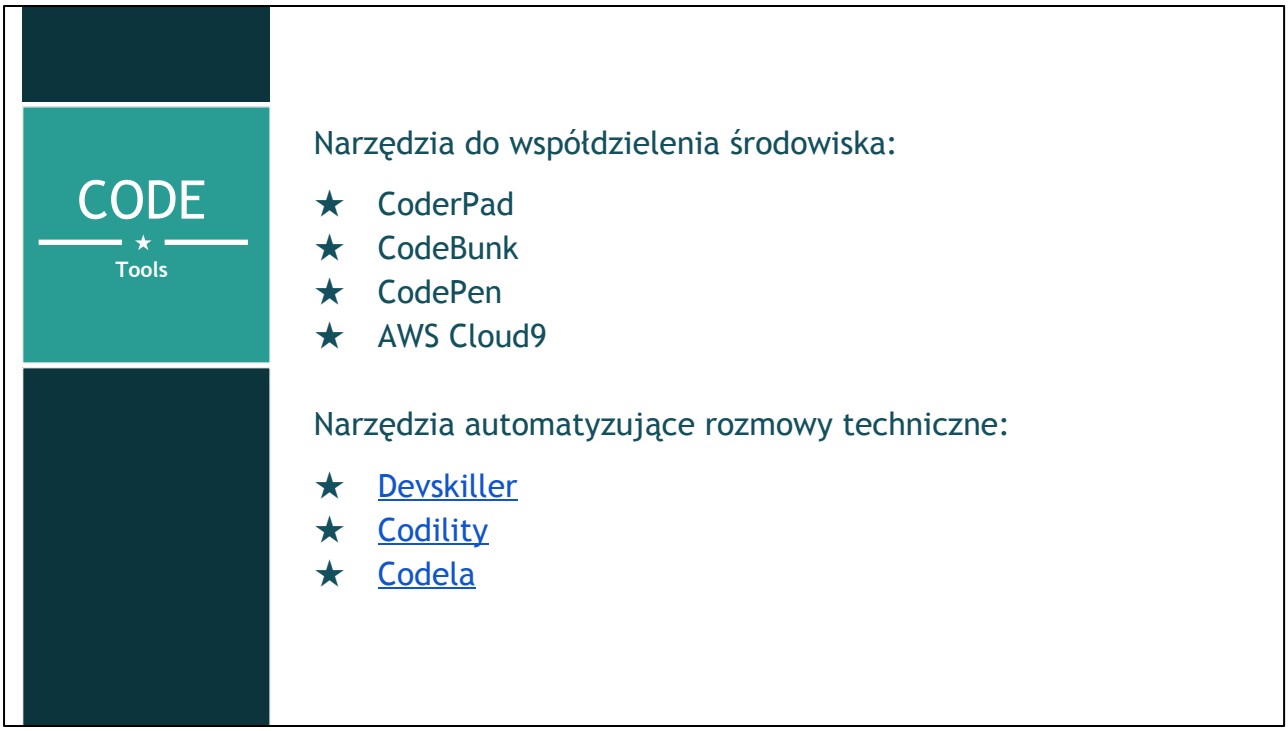
- ★ podział na wiersze

```
>>> print(get_rows([1, 2, 3, 4, 5], 3))
[[1, 2, 3], [4, 5]]
```

- ★ GROT
- ★ WeirdText

Przykładowe pytania używane kiedyś w STX Next.

1. Inne zadania na jakie trafiałem podczas rozmowów.
2. Zadawane szczególnie na stanowiska seniorskie.
3. Osobiście lubię 3 pierwsze (unittesty, code-review, debugowanie) po imitują codzienną pracę i sprawdzają umiejętności które nabywa się z doświadczeniem (nie da się ich nauczyć z książki).
4. Nie cierpię pytań o Big O notation, ponieważ jest to “czysta” teoria, nie przydatna w pracy programisty webowego.
5. Pytanie natomiast jest na miejscu gdy w ogloszeniu o pracę jest Machine Learning, Data Science, cryptography itd.



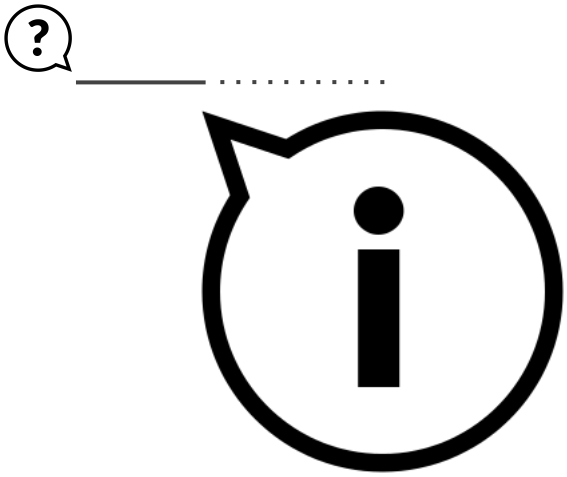
1. Narzędzia z pierwszej grupy pozwalają na wygodne pisanie kodu, a ekspert techniczny może zobaczyć także proces powstawania kodu.
2. Narzędzia z drugiej grupy nie są lubiane przez developerów, ponieważ są odhumanizowane.
    a. Osobiście też ich nie lubię i nie stosuję.
    b. Pytania w raczej dotyczą algorytmów matematycznych.
    c. Nie można zapauzować testu, więc podejdź do niego gdy będziesz miał sporo czasu.
    d. Narzędzia mają detektory ctrl+c/ctrl+v oraz sprawdzają stopień podobieństwa do innych rozwiązań.
    e. Linki pod nazwami narzędzi prowadzą do pytań rekrutacyjnych używanych wcześniej na tych platformach. Dużo przykładów do ćwiczenia.

- ★ Project Euler
- ★ CheckIO
- ★ CodinGame
- ★ CodeCombat
- ★ CodeWars
- ★ exercism.io
- ★ GROT[2]

- ★ on-call duty - nauka debugowania złożonych problemów

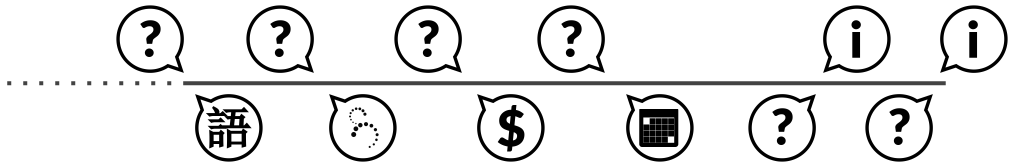
1. Strony z grami i wyzwaniami programistycznymi.
2. Dobry sposób na poćwiczenie zadań algorytmicznych.

Best project



1. Pytanie o projekt w którym pracowałeś lub inne osiągnięcie z którego jesteś dumny.
2. Zadawane zarówno przez rekruterów nietechnicznych jak i technicznych.
3. Nie lekceważ tego pytania.
4. Różne osoby szukają różnych aspektów:
    a. Pasja do języka
    b. Zaangażowanie w pracę
    c. Zespołowość (np. zrobiłem vs. zrobiliśmy)
    d. Rola w zespole (np. proaktywność)

End



1. Pytania, które zawsze się pojawiają na rozmowie:
   a. języki obce - często rozmowa kilkuminutowa, rzadziej cała rozmowa
   b. znajomość Scrum/Agile
   c. oczekiwania finansowe - sprawdź czy są widełki i gdzie się plasujesz w widełkach
   d. dostępność - okres wypowiedzenia i ew. skrócenie
2. Czas na Twoje pytania:
   a. swoimi pytaniami możesz pokazać, że przygotowałeś się do rozmowy (np. poczytałeś o firmie), pokazujesz zaangażowanie
   b. pytania pokazują co Cię interesuje u pracodawcy (np. dobre praktyki programistyczne czy benefity)

- ★ Przeanalizuj ogłoszenie o pracę
- ★ Zaktualizuj CV (szczególnie “agencyjne”)
- ★ Ustal dostępność (np. sprawdź obecną umowę)
- ★ Zastanów się nad wynagrodzeniem i rodzajem umowy
- ★ Poproś o spotkanie “rano”
- ★ Nie przesadź z ubiorem

- ★ Spytaj czy możesz “googlować”
- ★ Nie rób ctrl+c / ctrl+v

- ★ Poproś o feedback
- ★ Podziękuj

- ★ książka “Cracking the Coding Interview”, Gayle L. McDowell

Dodatkowe wskazówki

Podsumowanie

- ★ Ćwicz!
- ★ Trenuj!
- ★ Dokształcaj się!

Powodzenia! :)


ANY
QUESTIONS?
STXNEXT
ZAWINĘLIŚMY
DO
GDAŃSKA

[Monty Python Job Interview]

1. Jeśli jeszcze tego nie wiedziałeś to polecam!
2. Jak kiedyś przyjdzie ktoś z wpisanym “Monty Python” jako hobby w CV to mu zrobię taką rozmowę i będę patrzył kiedy się zorientuje :P