Faites un bon `match` avec Python

David Blanchet - @dblanchet@piaille.fr
Python Rennes - 2024

`Espace` pour le slide suivant, `S` pour afficher les notes, `Échap` pour la navigation rapide

Speaker notes

Hello, welcome, thanks for coming.

I'm...

This talk should help you to `match` better!

Pas dans ce talk

Speaker notes

Not in this talk:

- Find/anticipate sport match results, with Python AI, ML or whatever is currently trending.
- How to improve you "swipe right" rate with Python code.
- Not even how to deal with regular expressions in Python. Even if RE code has an impact on it, we'll talk about that in a few slides.

Pas dans ce talk

🎾⚽🏈...

Pas dans ce talk

🎾⚽🏉...

Pas dans ce talk

🎾⚽🏉...

```python
pattern = r'reg(ular )?ex(p|pression)?'
if result := re.match(pattern, haystack):
    #           ^^^^^ <-- Not this one either
    print(result.group(0))
```

Le mot-clé `match`

Speaker notes

This is about a new Python new `match` keyword, and the new language construction it allows.

`match` comme dans...

Speaker notes

Why this name? Where does it come from?

`match` comme dans...

pattern `match`ing

pattern `match`ing?

Speaker notes

`pattern matching`, that you may already know from several functional programming languages.

Show Haskell example

```
a, b = 0, 1
```

Speaker notes

Python already has something that looks a little like pattern matching.

```
first, snd, third =
range(3)
```

Speaker notes

Maybe you don't know this one?

Idea: **same number** of items each side of the = sign.

Is there a way to deal with situations where it is not the same counts?

```python
first, snd, *tail =
      range(5)
```

Speaker notes

Yes.

What does this do?

```
first, *mid, last = range(5)
```

Speaker notes

What about this one?

```
*head, prev, last =
range(5)
```

Speaker notes

Or this one?

pattern matching

Speaker notes

So, what is the point of pattern matching?

Let the code know the "form", the "pattern" data should get to choose how to deal with them.

https://en.wikipedia.org/wiki/Pattern_matching:

> *act of checking a given sequence of tokens for the presence of the constituents of some pattern.*

Let's dive into some hopefully highlightening examples.

Le mot-clé `match`

Speaker notes

This is about a new Python new `match` keyword, and the new language construction it allows.

It came with Python 3.10, launched in oct. 2021.

Premier contact

```python
if status_code == 200:
    msg = "OK"
elif status_code == 404:
    msg = "Not found"
elif status_code in (401, 403):
    msg = "Access control trouble"
else:
    msg = "Other error"
```

Speaker notes

First example, same code without and with `match`.

Premier contact

```python
if status_code == 200:
    msg = "OK"
elif status_code == 404:
    msg = "Not found"
elif status_code in (401, 403):
    msg = "Access control trouble"
else:
    msg = "Other error"
```

Avec `match`

```python
match status_code:
    case 200:
        msg = "OK"
    case 404:
        msg = "Not found"
    case 401 | 403:
        msg = "Access control trouble"
    case _:
        msg = "Other error"
```

À première vue

```
match ...:
    case ...:
        ...
    case ...:
        ...
```

Speaker notes

If you extract only the structure.

I can hear you...

"C'est juste un `switch/case`..."

Speaker notes

... it is just a `switch`/`case`.

"Ah ! non ! c’est un peu court, jeune homme !"

C. de Bergerac, (presque) à propos de Python

Speaker notes

`switch/case` does not exist in Python.

It would be an error to stop at this similarity.

Let's explore some of the interesting capabilities of this new language instruction.

Livecoding

Speaker notes

The best way to do this is to show code.

Do you know "logo"? *Show builtin example, tell it is also Ok with dicts*

- Key concept here is "destructuring"
- Tell what you want, not how to get it, e.g. list comprehensions

Show cplx example

- Your own classes too

Show more_cplx example

- Tree-like/recursive matching

- `__builtins__`
- `@dataclasses`
- `enums.Enum`
- Vos classes !

Speaker notes

Works where you would expect.

Speaker notes

confusing:

- bindings, for beginners
- soft keywords, for intermediate users

- Souple... mais complexe

- Souple... mais complexe
- Puissant... mais déroutant

- Souple... mais complexe
- Puissant... mais déroutant
- Sucre syntaxique... mais **expressivité**

Speaker notes

Want to know more? Specs!

And history, too:

- 2020-06: First attempt was PEP 622.
- 2020-07: Creation of pep-0634 to pep-0636
- 2021-02: PEPs accepted
- 2021-10: Python3.10, first release with `match` support!

PEP 634 - spécifications techniques → *Quoi*

PEP 634 - spécifications techniques → *Quoi*

PEP 635 - motivations → *Pourquoi*

PEP 634 - spécifications techniques → *Quoi*

PEP 635 - motivations → *Pourquoi*

PEP 636 - tutoriel → *Comment*

C'était...

Faites un bon `match` avec Python

`Merci !`

Questions ?

David Blanchet - @dblanchet@piaille.fr
Python Rennes - 2024