

Język znaczników MARKDOWN

Markdown jest bardzo prostym językiem znaczników. Jest on powszechnie stosowany między innymi w serwisie GitHub do dokumentacji repozytoriów, ale stał się bardzo popularnym narzędziem, które pozwala w znacznie prostszy sposób tworzyć proste dokumenty renderowane do formatu HTML.

Polecane źródła wiedzy o markdown:

- <https://www.markdownguide.org>
- <https://www.dillinger.io>

1. Podstawowe elementy markdown.

Pogrubienie

Tekst **pogrubiony** uzyskujemy poprzez otoczenie tekstu znacznikami podwójnej gwiazdki ****** lub podkreślenia **__**, jednak umieszczenie spacji między znacznikami a tekstem powoduje wyłączenie __ jego __ ****** funkcji ******. Jest to odpowiednik znacznika HTML ``.

Kursywa

Kursywa jest uzyskiwana poprzez dodanie pojedynczego podkreślenia otaczającego tekst, np. *_kursywa_* lub pojedynczej **gwiazdki**, ale ponownie spacja może zapobiec jego ** poprawnego* _ użycia_*. Odpowiada znacznikowi `` w HTML.

Znaczniki możemy zagnieżdżać tak jak w HTML, np. *****Bold and beautiful*****, pamiętajmy jednak o kolejności. Osiągamy efekt obu znaczników, czyli ***pogrubioną kursywę***. Ten sam efekt możemy osiągnąć poprzez wstawienie potrójnego podkreślenia lub potrójnych gwiazdek

```
__Jak wyszło?__  
***A tu podobnie?***
```

Jak wyszło? A tu podobnie?

Dlaczego powyższy tekst został wypisany w jednej linii? O tym poniżej.

Paragraf

Nowy paragraf umieszczamy poprzez wstawienie jednej pustej linii pomiędzy wierszami tekstu, tak jak w przypadku dwóch paragrafów powyżej. Odpowiada znacznikowi `<p>`.

```
Paragraf jeden.  
  
Paragraf dwa.
```

Jeżeli zapiszemy dwie linie nawet dodając znak nowego wiersza w poprzedniej nie spowoduje rozpoczęcia nowego paragrafu.

Powyższy tekst został zapisany jako:

```
Jeżeli zapiszemy dwie linie
nawet dodając znak nowego wiersza w poprzedniej
nie spowoduje rozpoczęcia nowego paragrafu.
```

Miękka spacja (ang. soft break) to rozpoczęcie kolejnej linii od nowego wiersza, ale bez widocznego odstępu w postaci pustej linii. Możemy to osiągnąć poprzez dodanie w poprzedniej linii dwóch spacji na jej końcu. Odpowiada znacznikowi `
`.

Linia pierwsza.

Linia druga.

```
Linia pierwsza.<spacja><spacja>
Linia druga.
```

Nagłówki

Nagłówki, czyli odpowiedniki znaczników `<h1>` do `<h6>` znanych z HTML to znak `#` wstawiany przed tekstem nagłówka w liczbie odpowiadającej poziomowi nagłówka.

```
# nagłówek poziomu 1
## nagłówek poziomu 2
### nagłówek poziomu 3
#### nagłówek poziomu 4
##### nagłówek poziomu 5
##### nagłówek poziomu 6
```

Poniżej efekt

nagłówek poziomu 1

nagłówek poziomu 2

nagłówek poziomu 3

nagłówek poziomu 4

nagłówek poziomu 5

nagłówek poziomu 6

Blok cytowania

Blok cytowania tworzymy poprzez rozpoczęcie linii od znacznika `>`, np.

```
> To jest blok cytowania
>
> W trzech wierszach.
```

To jest blok cytowania

W trzech wierszach.

Te bloki również możemy zagnieżdzać.

```
> Cytowanie
>> zagnieżdżone
```

Cytowanie

zagnieżdżone

Listy numerowane i punktowane

Listy numerowane zapisujemy właściwie tak jak są później wyświetlane, z tym, że ewentualne pomyłki w numeracji w źródle zostaną poprawione w renderowanej wersji.

```
1. Element 1.
2. Element 2.
2. Element 3.
```

- 1. Element 1.
- 2. Element 2.
- 3. Element 3.

Zagnieżdżone listy wymagają wstawiania wcięć w odpowiednim miejscu.

```
1. Element 1.
  1. Element 1.1
    1. Element 1.1.1
2. Element 2
```

- 1. Element
 - 1. Element 1.1
 - 2. Element 1.1.1
- 2. Element 2.

Pojedyncze wcięcie to **4 spacje**.

Listy nienumerowane (punktowane) tworzymy używając znaczników w postaci *, - lub +.

```
Dziś muszę kupić:  
* sztabkę masła ;-)  
+ ogórka  
- sok pomarańczowy  
* bułkę  
  
A potem:  
- marchew  
* buraka  
+ i cebulę
```

Dziś muszę kupić:

- sztabkę masła 🍞
- ogórka
- sok pomarańczowy
- bułkę

A potem:

- marchew
- buraka
- i cebulę

Jak żąglujemy znacznikami to w efekcie wizualnie uzyskujemy większe odstępy pomiędzy elementami listy i oficjalna dokumentacja mówi, że nie powinniśmy tego robić. Zamieniając znaczniki na jeden z nich efekt jest nieco inny.\

```
Dziś muszę kupić:  
* sztabkę masła ;-)  
* ogórka  
* sok pomarańczowy  
* bułkę  
  
A potem jednak zagnieździmy:  
* warzywa:  
  * buraka  
  * i cebulę
```

Dziś muszę kupić:

- sztabkę masła 🧈
- ogórka
- sok pomarańczowy
- bułkę

A potem jednak zagnieźdźmy:

- warzywa:
 - buraka
 - i cebulę:
 - czerwoną,
 - szalotkę.

Listy numerowane i punktowane możemy łączyć ze sobą:

```
1. Dziś muszę kupić:  
  * sztabkę masła ;-)  
  * ogórka  
  * sok pomarańczowy  
  * bułkę  
2. A potem jednak zagnieźdźmy:  
  * warzywa:  
  * buraka  
  * i cebulę:  
    * czerwoną,  
    * szalotkę.
```

- ```
1. Dziś muszę kupić:
 ◦ sztabkę masła 🧈 * ogórka
 ◦ sok pomarańczowy
 ◦ bułkę
2. A potem jednak zagnieźdźmy:
 ◦ warzywa:
 ◦ buraka
 ◦ i cebulę:
 ▪ czerwoną,
 ▪ szalotkę.
```

Tu wcięcia są wyrównane do miejsca rozpoczęcia się tekstu danego punktu w poprzednim elemencie listy.

## Kod i bloki kodu

Wyróżnianie elementów w tekście, które odnoszą się np. do nazw **zmiennych** czy **komend** lub **innych elementów**, które chcemy wyróżnić w ten sposób w trybie liniowym (ang. inline) odbywa się poprzez otoczenie tekstu znakami **backtick** ```. Jeżeli chcemy jednak umieścić większy fragment kodu w postaci np. listingu to używamy potrójnego backtica jak otwierającego i zamykającego znacznika oraz możemy dodać nazwę technologii (języka), w której ten fragment został napisany po to, aby sformatować go

charakterystycznie dla danej technologii, ale ten wygląd może się różnić w zależności od implementacji interpretera markdown.

```
```python
print('Hello Python!')
```
```

W efekcie wyświetli nam:

```
print('Hello Python!')
```

```
```bash
sudo apt install git
```
```

```
sudo apt install git
```

```
```sql
SELECT * FROM person;
```
```

```
SELECT * FROM person;
```

## Linki i obrazy

W zależności od implementacji, możemy po prostu wstawić link z określonym protokołem, np. <https://www.onet.pl>, ale gdy chcemy temu linkowi nadać inny alias w tekście, który będzie przekierowywał na dany adres to stosujemy składnię [alias](adres URL).

[Onet](http://www.onet.pl)

Efekt:

[Onet](http://www.onet.pl)

Możemy również definiować odnośniki, które da się wielokrotnie wykorzystywać w dokumencie bez konieczności ich każdorazowego przepisywania. Definiujemy link w sposób:

[Onet]: <https://www.onet.pl>

a następnie w tekście tylko odnośnik [Onet](#)

Obrazy definiujemy bardzo podobnie, ale poprzedzamy tę definicję wykrzyknikiem.

![BGG](https://cf.geekdo-static.com/images/logos/navbar-logo-bgg-b2.svg)



Odnosniki mogą również odnosić się do elementów znajdujących się w systemie plików, w którym znajduje się również niniejszy dokument. Możemy podawać ścieżki relatywne (co jest oczywiście zalecane) lub globalne. Podlinkowanie do dokumentu z wprowadzeniem do [narzędzia git](#) w ramach tych zajęć nie jest trudne.

Definicja w tekście: [narzędzia git](../\_\_git/readme.md)

Inne elementy

Do podstawowych elementów możemy również zaliczyć poziome linie, które możemy wstawić poprzez potrójną gwiazdkę \*\*\* lub myślnik ---.

---

I druga linia.

---

Tutaj należy zaznaczyć, że wstawienie takiej linii poprzez dowolną liczbę myślników >= 2 tuż pod tekstem spowoduje... że stanie się on nagłówkiem poziomu 2! A w przypadku znaku == będzie to nagłówek poziomu 1.

2. Elementy rozszerzonego markdowna.

Elementy, które nie wpisują się w podstawowe elementy języka znaczników markdown bywają prędkiej czy później potrzebne i bardzo przydatne. Poniżej opis kilku wybranych z nich.

Tabele

Dane tabelaryczne to dość częsty element stron internetowych. W markdownie definiujemy je tak:

```
Kolumna 1	Kolumna 2	Kolumna ...
dane	dane	dane
dane	dane	dane
```

| Kolumna 1 | Kolumna 2 | Kolumna ... |
|-----------|-----------|-------------|
| dane      | dane      | dane        |
| dane      | dane      | dane        |

Ich wygląd ponownie może różnić się w zależności od implementacji formatowania.

Poniżej dodano wyrównanie w kolumnach.

```
Kolumna 1	Kolumna 2	Kolumna ...
dane	dane	dane
dane	dane	dane
```

| Kolumna 1 | Kolumna 2 | Kolumna ... |
|-----------|-----------|-------------|
|-----------|-----------|-------------|

| Kolumna 1 | Kolumna 2 | Kolumna ... |
|-----------|-----------|-------------|
| dane      | dane      | dane        |
| dane      | dane      | dane        |

**Przypisy dolne**

Przypisy dolne możemy definiować poprzez znacznik [<sup>etykieta</sup>], która to musi zostać wcześniej zdefiniowana.

[<sup>etykieta</sup>]: To jest odnośnik, który może zawierać np. **kod** lub inne elementy. Ale nie w każdym interpreterze markdown musi działać.

**Lista zadań - checkboxy**

Jest to dość wygodny sposób na budowanie np. listy TODO.

```
- [x] Wstęp do Pythona
- [] Narzędzie git
- [] Markdown
```

- ☒ Wstęp do Pythona
- ☐ Narzędzie git
- ☐ Markdown

**Indeks górny i dolny**

Indeks górny np. X<sup>2</sup> może nie działać w każdej wersji, jak również i indeks dolny: H<sub>2</sub>O.

**Podświetlenie tekstu**

Podświetlenie ==ważnego elementu== tekstu poprzez podwójne znaczniki ==, jednak tutaj również musimy zwrócić uwagę czy w danym interpreterze będziemy widzieć odpowiedni efekt.

**Emoji 😊**

Emotki definiujemy umieszczając ich nazwę pomiędzy znakami :nazwa emoji:.