Zmienne i typy podstawowe

Deklaracja zmiennych

W Pythonie bezpiecznie jest myśleć o zmiennych jak o etykietach wskazujących nam na jakąś zawartość (formalnie przyjmujemy, że oba te elementy są częścią zmiennej - jednak wg. mnie zaproponowane podejście ułatwia na początku uniknięcie wielu błędów). Ta sama zawartość może mieć wiele etykiet (czyli kilka zmiennych może wskazywać na te same dane - warto jednak takich sytuacji unikać).

Zmienne deklarujemy wg. poniższego schematu:

nazwa_zmiennej = zawartość

Znak równości służy więc do przypisania nazwy (którą umieszczamy po lewej stronie znaku) do jakiegoś obiektu (który umieszczamy po prawej stronie znaku) np.

```
moj_zwierzak="pies"
```

W powyższym przykladzie `moj_zwierzak` to nazwa zmiennej (którą będziemy wykorzystywać w programie) a `"pies"` to zawartość.

Tworząc nazwę MUSIMY przestrzegać kilku zasad:

- nazwa nie może zawierać większości znaków specjalnych (w tym spacji - można zastąpić j ą `_`)
- nie może składać się z samych liczb, ani zaczynać od nich

Dobrą praktyką jest:

- używanie jedynie małych liter
- jeśli chcemy aby nazwa składała się z kilku słów oddzielamy je "podłogą" (czyli `_`)
- nadawanie nazw, które dobrze oddają zawartość (nie powinny jednak być przesadnie długie)
- unikanie polskich znaków

Dynamiczne typy zmiennych

Zwróć uwagę, że raz zadeklarowana zmienna MOŻE zmienić swój typ - np. coś co wskazywało na liczbę chwilę później może wskazywać na tekst. Czyli w Pythonie taka operacja wykona się poprawnie:

```
moja_zmienna = 1
moja_zmienna = "pies"
```

W wielu innych językach (tzw. statycznie typowanych) taki zapis byłby niepoprawny - jeśli raz przypisalibyśmy do zmiennej np. liczbę to przez cały cykl życia aplikacji trzeba by było się tego trzymać.

Mimo tej możliwości którą daje Python zazwyczaj dobrą praktyką jest trzymanie pod utworzoną zmienną jednego typu danych.

Print

- służy do wyświetlania na ekranie reprezentacji obiektu (np. do wyświetlania tekstu)
- składnia wygląda następująco:

print(zawartość_do_wyświetlenia)

np.

```
zdanie = "To jakiś tekst"
```

spowoduje wyświetlenie następującej zawartości w terminalu:

```
To jakiś tekst
```

Należy pamiętać o pisaniu polecenia z małej litery - czyli `print` jest ok, podczas gdy np. `Print` nie zostanie rozpoznane (w większości języków programowania wielkość liter ma znaczenie).

Input

- służy do umożliwienia użytkownikowi wprowadzenia jakiejś zawartości
- wszystko co zostanie wprowadzone w ten sposób będzie traktowane jako tekst(!), jeśli chcemy wprowadzić z wykorzystaniem `input` liczbę musimy dokonać konwersji z tekstu na np. liczbę całkowitą, lub zmiennoprzecinkową
- składnia wygląda następująco:

nazwa_zmiennej = input ("opcjonalny_komunikat dla użytkownika")

np.

```
imie_gracza = input("Wprowadz imię: ")
```

Typy podstawowe

Typy określają wspólne cechy obiektów, metody z nimi związane. Dzięki nim programiści wiedzą jakie operacje na nich są dozwolone (a jakie nie). Są więc bardzo przydatnym środkiem komunikacji przy pisaniu aplikacji.

Python obsługuje następujące typy podstawowe:

Liczby ca?kowite (integer)

Liczy naturalne (1, 2, 3, 4...) i przeciwne do nich (-1, -2, -3) oraz zero.

Przypisując do zmiennej liczbę całkowitą po prostu podajemy ją po znaku równości np.

```
moja_liczba = 12
```

Liczby zmiennoprzecinkowe (float)

Liczby zapisane z "częścią ułamkową". Ważne - część ułamkowa zapisana jest po KROPCE, nie po przecinku.

```
moja_liczba = 22.15
```

Ci?gi znaków (string)

Seria znaków zapisana między cudzysłowami pojedynczymi, lub podwójnymi (nie można ich jednak mieszać!) np.

```
moj_tekst = "To tylko tekst."
```

Warto?ci logiczne (bool)

Wartości prawda / fałsz zapisane jako True / False (istotna jest wielkość znaków!).

np.

```
gra_gotowa =  True
```

Podstawowe operacje

Podstawowe operatory to:

dodawanie `+`

odejmowanie `-`

mnożenie `*`

dzielenie `/`

modulo `%` (reszta z dzielenia)

Można ich używać na:

	integer	float	string	bool
`+`	tak	tak	tak	tak
`-`	tak	tak	nie	tak
`*`	tak	tak	tak, ale...	tak
`/`	tak	tak	nie	-
`%`	tak	tak	nie	-

W przypadku ciągu znaków (string) możemy wykonać operację mnożenia przez liczbę - powtórzy nam to dany tekst wskazaną ilość razy. Nie można jednak mnożyć ciągu znaków przez ciąg znaków.

Wartości logiczne True / False są równoważne 1 i 0 - stąd też da się wykonać operacje dzielenia, czy modulo... z założeniem, że nie dzielimy przez 0 ;) .

Odno?niki

„3.12.5 Documentation”. Dostęp 18 sierpień 2024. https://docs.python.org/3/.

Viafore, Patrick. *Robust Python: write clean and maintainable code*. First edition. Beijing [China]; Boston [MA]: O’Reilly, 2021.

„Zmienna (informatyka)”. W *Wikipedia, wolna encyklopedia*, 15 kwiecień 2024. https://pl.wikipedia.org/w/index.php?title=Zmienna_(informatyka)&oldid=73515366.

„Magic Python: Mutable vs Immutable and How To Copy Objects”. Dostęp 18 sierpień 2024 https://alexkataev.medium.com/magic-python-mutable-vs-immutable-and-how-to-copy-objects-908bffb811fa

---

Wersja #13
Utworzono 2024-08-25 16:10:59 UTC przez Przemek Jeske
Zaktualizowano 2026-04-28 20:31:10 UTC przez Przemek Jeske