PODSTAWY PROGRAMOWANIA W JĘZYKU PYTHON

Dzień 4

Ankieta

https://ankiety.is-academy.pl/poll/-LDjsR30ylyU72y0gH8E/-LfKF1DMNUYJ6ERRpO1r

Agenda

- Inne kolekcje: słownik
- Funkcja:
  - Definicja
  - Wejście i wyjście
  - Zakres zmiennych
  - Docstring

Kolekcje c.d.

Słownik

```
dict(), {klucz1: wartosc1, ...}
```

- Zbiór par klucz - wartość, mogących różnić się typem wartości (zarówno klucz - wartość jaki i klucze między sobą).
- Klucz musi być typem niezmiennym (np. string, int, tuple) i być unikatowy (tylko jeden wewnątrz słownika).
- Klucze nie zachowują kolejności (alfabetycznej, podania).

Kolekcje

	Lista (list)	Krotka (tuple)	Zbiór (set)	Słownik (dict)
Czy edytowalne (mutable)?	Tak	Nie	Tak	Tak
Indeksowanie [ ]	Tak	Tak	Nie	Tak
Indeks	Liczba od 0	Liczba od 0	-	Obiekt niezmiennego typu.
Wymaga unikatowych wartości	Nie	Nie	Tak	Tak (tylko klucze)
Zachowuje kolejność wprowadzania	Tak	Tak	Nie	Nie

Funkcje

Funkcja

- Blok wydzielonego, powtarzalnego kodu, używany do wykonania określonego, *możliwie prostej* działania.
- Funkcje zapewniają modułowość kodu oraz ułatwiają jego ponowne użycie (*zasada DRY*).
- Python posiada funkcje wbudowane oraz pozwala na ich definiowanie.
- Wywołanie funkcji:

```
nazwa_funkcji()
nazwa_funkcji(argumenty)
type(“Ala ma kota”)
```

Definicja funkcji

```
def nazwa_funkcji(parametr1, ...):
    Instrukcja
    Instrukcja
    Instrukcja/wyrazenie:
        Instrukcja
    Instrukcja
    return wartosc
```

Nagłówek funkcji
(nazwa + parametry)

Ciało
funkcji

Wartość zwracana

Definicja funkcji - słowa kluczowe

- **Definicja funkcji** musi zaczynać się od wyrażenia *def*.
- **Nazwa funkcji** pojawia się w przestrzeni nazw.
- Funkcja posiada od zera, do nieskończenie* wielu **parametrów**.
- **Ciało funkcji** tworzy swoją przestrzeń nazw, przez rozszerzenie przestrzeni nazw, w której jest zdefiniowana o parametry z **nagłówka funkcji** (mogą nadpisać wcześniejsze zmienne).
- **Funkcja “zawsze coś zwraca”**, choć nie zawsze musi posiadać wyrażenie *return*. Może zawierać ich wiele, lecz instrukcje w bloku po wyrażeniu *return* nigdy nie będą wykonane.

Argumenty
i wartości zwracane

Argumenty funkcji

- Funkcja może nie posiadać żadnych argumentów.

```
>>> def funkcja_1():
        return 5
>>> funkcja_1()  # 5
```

- Paramtery funkcji przypisywane są kolejno przy wywołaniu.

```
>>> def funkcja_2(x, y):
        return x + y
>>> funkcja_2(4, 2)  # 6, x = 4, y = 2
```

Argumenty funkcji

- Jeżeli podamy mniej wartości niż funkcja ma parametrów, interpreter zgłosi błąd.

```
>>> def funkcja_2(x, y):
        pass
>>> funkcja_2(4)  # TypeError
```

- Jeżeli podajemy parametry po nazwach, możemy zmienić ich kolejność.

```
>>> def funkcja_2(x, y):
        pass
>>> funkcja_2(y=4, x=2)  # x = 4, y = 2
```

Argumenty funkcji

- Jeżeli funkcja ma wiele parametrów, możemy wywołać ją w sposób mieszany, część parametrów podając jawnie (*explicit*), *część nie*.

```
>>> def funkcja_3(x, y, z):
        pass
>>> funkcja_3(4, z=1, y=3)
```

- Należy wtedy pamiętać, że nie możemy podać parametru więcej niż raz, oraz podawać paramtery pozycyjnie (niejawnie) za jawnymi.

```
>>> funkcja_3(4, z=1, x=3)  # TypeError
>>> funkcja_3(y=4, x=2, 5)  # SyntaxError
```

Argumenty domyślne

Możliwe jest przypisanie domyślnej wartości parametru funkcji. Wartość ta będzie przypisana, gdy funkcja zostanie wywołana z mniejszą liczbą parametrów (jednak wszystkie parametry pozycyjne muszą być zapewnione). **Nie powinno się używać kolekcji mutowalnych** (np. list) **jako parametrów domyślnych**.

```
>>> def funkcja_4(x, y, z=1):
        pass
>>> funkcja_4(4, 5)  # x=4, y=5, z=1
>>> funkcja_4(4, 5, 6)  # x=4, y=5, z=6
>>> funkcja_4(4)  # TypeError
```

Wartość zwracana

- Jeżeli chcemy, żeby funkcja zwróciła jakąś wartość używamy wyrażenia *return wartosc*.
- Możemy użyć również samego wyrażenia *return*, wówczas zwracana jest wartość `None`.
- Funkcja nie posiadająca w swoim ciele wyrażenia *return* również zwraca wartość `None`.
- Wartość/wartości zwracane przez funkcję możemy przypisać do zmiennej.

```
>>> def square(x):
        return x * x
>>> z = square(4)  # z = 16
```

Zakres zmiennych

Przestrzenie nazw

W Pythonie możemy rozważyć 3 przestrzenie nazw:

- wbudowaną (*built-in*) - nazwy dostępne po uruchomieniu interpretera w każdym module
- globalną (*global*) - zmienne zdefiniowane na poziomie modułu
- lokalną (*local*) - zmienne dostępne w obecnym bloku kodu

Funkcję `locals` i `globals` zwracają słowniki zmiennych dostępnych w odpowiednich przestrzeniach nazw.

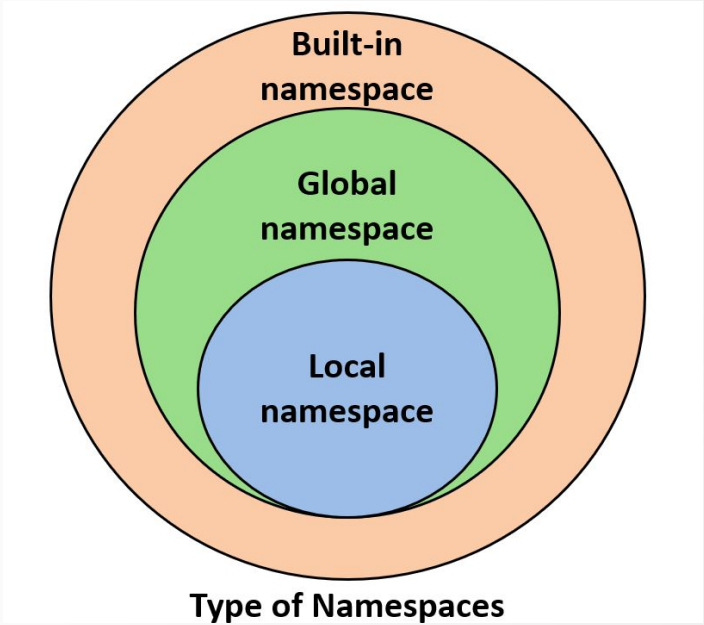

Image (05.06.2019): https://cdncontribute.geeksforgeeks.org/wp-content/uploads/types_namespace-1.png

Zakres zmiennych

- Program będzie próbował znaleźć wywołaną nazwę zmiennej w lokalnym zakresie.
- Jeżeli zmienna nie jest dostępna w danym zakresie to interpreter spróbuję ją znaleźć w *wyższym* zakresie.
- Cykl wyszukiwań trwa do momentu dotarcia do przestrzeni globalnej. Jeżeli i tam interpreter nie znajdzie zmiennej to sprawdza przestrzeń wbudowaną. Jeżeli i tam nie ma podnoszony jest błąd.
- Jeżeli chcemy od razu odwołać się do zmiennej z przestrzeni globalnej to możemy użyć instrukcji:

```
global zmienna
```

- Instrukcja nie może być poprzedzona definicją zmiennej w bloku kodu (także w nagłówku funkcji).

Zakres zmiennych - mutowalność

- Przestrzeń nazw możemy interpretować jako słownik.
- W związku z tym zmienna z obiektem mutowalnym jest *referencją* do tego obiektu. W przypadku zmutowania obiektu (zmienienia stanu np. dodania elementu w liście) gdzieś w lokalnym zakresie, zmiana ta będzie widoczna także w zakresie globalnym.
- Działa to także w drugą stronę - użycie instrukcji `global` może spowodować dodanie klucza do globalnej przestrzeni nazw.

Docstring

Docstring

- String opisujący sposób działania funkcji, jej parametry, wartości zwracane lub sposoby użycia.
- Docstring musi być zdefiniowany pod nagłówkiem funkcji za pomocą wielolinijkowego stringa:

```
def funkcja_4(x, y, z=1):
    """Docstring..."""
    pass
```

- Czasami docstring może być jednolinijkowy, czasami blokiem tekstu. Czasami może być też zupełnie niepotrzebny - choć częściej lepiej go napisać niż później płakać/tracić czas na *reverse engineering* ;)
- PEP 257: https://www.python.org/dev/peps/pep-0257/

Podsumowanie

- Inne kolekcje: słownik
- Funkcja:
  - Definicja
  - Wejście i wyjście
  - Zakres zmiennych
  - Docstring

Thanks!!