

Ćwiczenia powtórzenie

Ćwiczenie 1 - Intro

Przyjęło się, że pierwszym programem, który pisze się w nowo poznanym języku jest program wypisujący na ekranie napis „Hello, world!”.

```
print("Hello, world!")
```

- ◊ Napisz program, który wypisze na ekranie napis, który w miejsce world wyświetli Twoje imię.
- ◊ Niech program wypisuje drugi raz ten tekst w nowej linii (użyj jednej funkcji print i symbolu końca linii \n).
- ◊ Czy zmieni coś zamiana cudzysłowów "" na apostrofy ''

Ćwiczenie 2 - Typy

Napisz program, który poda jaki jest typ:

- ◊ 300
- ◊ 300.0
- ◊ "300"
- ◊ 300j

Jaki jest typ wyniku 2.+2 ?

Ćwiczenie 3 - Operatory

Jaki otrzymasz wynik ?

- ◊ Operatory arytmetyczne
 - ◊ `1+2+5-(2*2)`
 - ◊ `501.0 - 99.9999`
 - ◊ `2**3`
 - ◊ `10.0 / 4.0`
 - ◊ `10.0 // 4.0`
 - ◊ `5 % 2`
- ◊ Operatory porównania
 - ◊ `2 == 2`
 - ◊ `1 != 1`
 - ◊ `99 > 1.1`

Ćwiczenie 3 – Operatory c.d.

Jaki otrzymasz wynik ?

- ◊ Operatory logiczne
 - ◊ True or False
 - ◊ True and True and False
 - ◊ not False
- ◊ Operatory zawierania
 - ◊ "lis" not in "krowa, owca, pies"
 - ◊ "nic" in "Francja słynie ze swoich winnic"

Ćwiczenie 4 – Zmienne

- ◊ Stwórz zmienną example i przypisz do niej wartość None
- ◊ Wypisz na ekranie typ zmiennej example
- ◊ Nadpisz wartość zmiennej example nową wartością - "yes"
- ◊ Wypisz ponownie typ zmiennej example
- ◊ Stwórz dwie nowe zmienne (o dowolnych nazwach) i przypisz do nich wartości całkowitoliczbowe. Wypisz na ekranie wyniki dodania tych dwóch zmiennych
- ◊ Stwórz nową zmienną source_var i przypisz do niej wartość "*Magic*"
- ◊ Stwórz nową zmienną copy_var i przypisz do niej wartość zmiennej source_var
- ◊ Wypisz na ekranie wartość zmiennej copy_var. Czy jest taka sama jak wartość zmiennej source_var ?

Ćwiczenie 5 – Funkcja print i f-string

Przed wykonaniem poniższych linijek kodu spróbuj powiedzieć jaki będzie wynik.

- ◊ `print("Jaki", "piękny", "dzień")`
- ◊ `print("1", "2", 3, 4, 5)`
- ◊ `fruit = "pomarańcze"`
- ◊ `print("jabłka", "banany", fruit, sep=", ")`
- ◊ `print("Wypisywanie w trakcie...", end="")`
- ◊ `print(f"Lubię {fruit}!")`

Ćwiczenie 6 – Listy (metody i funkcje)

Dla danej listy `l = [1, 2, 3, 4, 5]`

1. wyświetl jej długość (użyj funkcję `len()`)
2. dodaj do niej szósty element o wartości 2 (użyj metody `append()`)
3. wyświetl ile elementów listy `l` ma wartość 2 (użyj metody `count()`)
4. wyświetl ile elementów listy `l` ma wartość 7
5. rozszerz listę `l` o listę `[6, 7, 8]` (użyj metody `extend()`)
6. sprawdź jaki indeks ma element listy `l` o wartości 7 (użyj metody `index()`)
7. dodaj element o wartości 10 na pierwszej pozycji listy (indeks 0) (użyj metody `insert()`)
8. użyj indeksowania do sprawdzenia, czy 10 została umieszczona w prawidłowy miejscu
9. wyświetl wartość ostatniego elementu listy `l` (użyj indeksu -1)
10. wyświetl wartość ostatniego elementu listy `l` usuwając go jednocześnie z listy (użyj metody `pop()`)
11. usuń z listy `l` element o wartości 4 (użyj metody `remove()`)
12. odwróć listę `l` i wyświetl ją (użyj metody `reverse()`)
13. posortuj listę `l` i wyświetl ją (użyj metody `sort()`)
14. wyczysć listę `l` i wyświetl ją (użyj metody `clear()`)

Podpowiedź: użyj instrukcji `help(list)`, żeby wyświetlić krótki opis wszystkich metod listy. Metody, których nazwy rozpoczynają się od `__` (double underscore = dunder) omówimy w bloku Python średnio-zaawansowany.

Ćwiczenie 7 – Listy (szatkowanie)

Dla danej listy `l = ["User1", "UserChris", "User2", "Admin"]`

1. wyświetl fragment listy składający się tylko z elementu `"UserChris"`
2. wyświetl fragment listy składający się ze wszystkich elementów poza pierwszym
3. wyświetl fragment listy składający się ze wszystkich elementów poza elementem `"Admin"`
4. wyświetl fragment listy składający się z pierwszych trzech elementów.

Ćwiczenie 8 – Krotki

Dla danego przepisu `k = ("boil water", "insert egg", "wait 5min", "eat")`

1. wyświetl trzeci krok przepisu (użyj operatora `[]`),
2. wyświetl kawałek listy składający się z dwóch ostatnich elementów przepisu,
3. sprawdź ile razy występuje element `"wait 5min"` w przepisie `k` (użyj metody `count()`),
4. sprawdź czy element `"boil water"` jest pierwszym elementem krotki (użyj metody `index()`).

Ćwiczenie 9 – Napisy (metody i funkcje)

Pamiętaj, że napis to także typ sekwencyjny.

Mamy napis `word = "konstantynopolitańczykowianeczka"`:

1. sprawdź ile razy w zmiennej `word` występuje litera "a" (użyj metody `count`)
2. znajdź indeks pierwszego wystąpienia litery „k” w zmiennej `word` (użyj metody `find`)
3. znajdź indeks pierwszego wystąpienia litery „k” w zmiennej `word` (użyj metody `index`)
4. znajdź indeks pierwszego wystąpienia litery „b” w zmiennej `word` (użyj metody `find`)
5. znajdź indeks pierwszego wystąpienia litery „b” w zmiennej `word` (użyj metody `index`)
6. znajdź indeks ostatniego wystąpienia litery „k” w zmiennej `word` (użyj metody `rfind` lub `rindex`)
7. sprawdź czy zmienna `word` zaczyna się od napisu "konst" (użyj metody `startswith`)
8. sprawdź czy zmienna `word` kończy się na „czka” (użyj metody `endswith`)
9. usuń z napisu „ ala ma kota ” znajdujące się na początku i końcu, białe znaki (użyj metody `strip`)
10. usuń z napisu „ ala ma kota ” znajdujące się na początku, białe znaki (użyj metody `lstrip`)
11. usuń z napisu „ ala ma kota ” znajdujące się na końcu, białe znaki (użyj metody `rstrip`)
12. utwórz z napisu "pierwszy, drugi, trzeci" trzyelementową listę `['pierwszy', 'drugi', 'trzeci']` (użyj metody `split`)

Ćwiczenie 9 – Napisy (metody i funkcje) c.d.

13. utwórz z napisu „pierwszy\ndrugi” dwuelementową listę (użyj metody `splitlines`).
14. rozdziel napis „Adres: Armii Krajowej 61” przyjmując za separator znak ":" (użyj metody `partition`)
15. utwórz z listy `['pierwszy', 'drugi', 'trzeci']` napis "pierwszy, drugi, trzeci" (użyj metody `join`)
16. zamień wszystkie wystąpienia znaku "X" w napisie "Witaj X! Czy mogę zwracać się do Ciebie X?" na swoje imię (użyj metody `replace`)
17. wyśrodkuj napis "środek" w dziesięcio-znakowym napisie (użyj metody `center`)
18. używając metody `ljust` uzyskaj identyczny wynik jak dla wywołania `"x".center(10)`
19. używając metody `rjust` uzyskaj identyczny wynik jak dla wywołania `"x".center(10)`
20. sprawdź czy napis "192" składa się z samych cyfr (użyj metody `isdigit`)
21. sprawdź czy napis "konstantynopolitańczykowianeczka" składa się tylko ze znaków alfabetycznych (użyj metody `isalpha`)
22. sprawdź czy napis "Warszawa, Graniczna 1" składa się tylko ze znaków alfanumerycznych. Jeżeli nie, popraw go tak, żeby składał się wyłącznie ze znaków alfanumerycznych (użyj metody `isalnum`)
23. sprawdź czy napis " \t\n" składa się wyłącznie z białych znaków (użyj metody `isspace`)
24. zmień wszystkie znaki w napisie "WyMiesZAnE WielKOŚci LiTeR" na małe (użyj metody `lower`)

Ćwiczenie 9 – Napisy (metody i funkcje) c.d.

25. sprawdź, czy w podanym przez Ciebie (dowolnym) słowie wszystkie znaki są małe (użyj metody `islower`)
26. zamień napis "`to jest tytuł`" na napis tytułowy (użyj metody `title`)
27. zamień wszystkie znaki w napisie "`Maroco to piękny kraj`" na wielkie (użyj metody `upper`)
28. sprawdź, czy napis "`iliada`" to napis tytułowy (użyj metody `istitle`). Jeżeli nie, popraw go (użyj metody `capitalize`)
29. zamień wszystkie litery w napisie "`ala ma kota`" na wielkie (użyj metody `upper`)
30. sprawdź, czy w podanym przez Ciebie (dowolnym) słowie wszystkie znaki są wielkie (użyj metody `isupper`)
31. Użyj metody `swapcase` na napisie "`BaRdZo PoMiEsZaNy NaPiS`". Jaki jest efekt działania metody `swapcase` ?
32. Porównaj napisy "`Straße`" i "`strasse`". Potem użyj metody `casifold` na napisie "`Straße`" i porównaj go z napisem "`strasse`".

Podpowiedź: użyj instrukcji `help(str)`, żeby wyświetlić krótki opis wszystkich metod napisu. Metody, których nazwy rozpoczynają się od `__` (double underscore = dunder) omówimy w bloku Python średnio-zaawansowany.

Ćwiczenie 10 – zbiory (metody i funkcje)

Dla danych zbiorów:

```
ryby = {"dorsz", "łosoś", "karp"}  
ptaki = {"bocian", "sroka"}  
zwierzeta = {"dorsz", "bocian"}  
maly_staw = {"karp"}
```

1. sprawdź czy "łosoś" jest elementem zbioru: ryby, zwierzęta (użyj operatora zawierania **in**)
2. sprawdź czy zbiory ryby i ptaki są rozłączne (użyj metody **isdisjoint**)
3. sprawdź czy zbiory ryby i zwierzeta są rozłączne (użyj metody **isdisjoint**)
4. sprawdź czy zbiór maly_staw jest podzbiorem zbioru ryby (użyj metody **issubset**)
5. sprawdź czy zbiór zwierzeta jest nadzbiorem zbioru ptaki (użyj metody **issuperset**)
6. stwórz nowy zbiór zwierzeta zawierający wszystkie elementy zbiorów ptaki i ryby (użyj metody **union**)
7. stwórz nowy zbiór zawierający elementy wspólne zbiorów maly_staw i ryby (użyj metody **intersection**)

Ćwiczenie 10 – zbiory (metody i funkcje) c.d.

7. stwórz nowy zbiór zawierający elementy, które znajdują się w zbiorze `ryby`, a które nie znajdują się w zbiorze `maly_staw` (użyj metody `difference`)
8. zaktualizuj zbiór `zwierzeta` o nowy zbiór `pets = {"cat", "dog"}` (użyj metody `update`)
9. zaktualizuj zbiór `ryby` zostawiając w nim tylko elementy wspólne ze zbiorem `maly_staw` (użyj metody `intersection_update`)
10. Zaktualizuj zbiór `ptaki` zostawiając w nim tylko te elementy, których nie ma w zbiorze `pets` (użyj metody `difference_update`)
11. Zaktualizuj zbiór `a = {1, 2, 3}` zostawiając w nim tylko elementy wspólne ze zbiorem `b = {3, 4, 5}` (użyj metody `symmetric_difference_update`)
12. usuń element `"dorsz"` ze zbioru `zwierzeta` (użyj metody `remove`)
13. usuń i zwróć dowolny element ze zbioru `zwierzeta` (użyj metody `pop`)
14. używając metody `discard` spróbuj usunąć elementy `"karp"` i `"rekin"` ze zbioru `ryby`. Co robi metoda `discard`, kiedy próbujesz usunąć nieistniejący element ?

Podpowiedź: użyj instrukcji `help(set)`, żeby wyświetlić krótki opis wszystkich metod zbioru. Metody, których nazwy rozpoczynają się od `__` (double underscore = dunder) omówimy w bloku Python średnio-zaawansowany.

Ćwiczenie 11 – słowniki (metody i funkcje)

Dla danego słownika `s = {1: "one", 2: "two", 3: "three"}`

1. wyświetl jego długość (użyj funkcję `len()`)
2. dodaj do słownika `s` parę klucz-wartość: `4: "four"` (użyj operatora `[]`)
3. wyświetl wartość pod kluczem `2` (użyj operatora `[]`)
4. wyświetl wartość pod nieistniejącym kluczem `10` (użyj operatora `[]`)
5. wyświetl wartość pod nieistniejącym kluczem `10` (użyj metody `get()`)
6. wyświetl wartość pod nieistniejącym kluczem `10` (użyj metody `get(key, default)`), parametrowi `default` przypisz wartość `"unknown"`)
7. użyj metody `items()`. Co zwraca metoda `items()`? Czy to lista ?

Ćwiczenie 11 – słowniki (metody i funkcje) c.d.

8. użyj metod `keys()` i `values()`. Co zwracają te metody? Zrzutuj to co zwracają na listę.
9. wyświetl wartość pod kluczem `2` przy użyciu metody `pop()`. Wyświetl zawartość słownika po użyciu metody `pop()`).
10. użyj metody `popitem()`. Wyświetl zawartość słownika po użyciu metody `popitem()`).
11. użyj metody `setdefault()` do stworzenia pary klucz-wartość `2: "two"`
12. spróbuj użyć metody `setdefault()` do stworzenia pary klucz-wartość `3: "new-three"`
13. Stwórz nowy słownik `{0: "zero"}`. Zaktualizuj słownik `s` o wartości z nowego słownika (użyj metody `update()`)
14. wyczysć słownik `s` (użyj metody `clear()`)
15. za pomocą listy `["x", "y", "z"]` utwórz słownik ze wszystkimi wartościami zainicjalizowanymi na wartość domyślną `False`.

Podpowiedź: użyj instrukcji `help(dict)`, żeby wyświetlić krótki opis wszystkich metod słownika. Metody, których nazwy rozpoczynają się od `__` (double underscore = dunder) omówimy w bloku Python średnio-zaawansowany.

Ćwiczenie 12 – operatory porównania i zawierania dla kolekcji

Przed wykonaniem poniższego kodu w konsoli Python spróbuj odgadnąć wynik:

```
{"A": 1, "B": 2} == {"A": 1}  
[0, 1, 2] == [0, 1, 2]
```

```
1 in (1, 2, 3)
```

```
1 not in {"A": 1}
```

```
1 in {"A": 1}.values()
```

```
[5, 4] == (5, 4)
```

```
"abc" < "abcd"
```

```
[10, 0] > [1, 5000, 6000]
```

```
("a", "x") > ("a", "b", "c")
```

```
{"abc", "zzz"} > {"abc", "abc"}
```

Ćwiczenie 13 - Warunek

Mamy książkę adresową postaci:

```
address_book = {  
    "Helena": "ul. Miodowa 3a\n69-896 Bory tucholskie",  
    "Teresa": "ul. Konopnickiej 25/11\n19-305 Ełk",  
    "Jan": "ul. Klonowa 6/47\n65-963 Warszawa",  
    "Wojciech": "ul. Przykładowa 15/6\n03-946 Przykładowo"  
}
```

Napisz program, który poprosi użytkownika o podanie imienia. Jeżeli imię znajduje się w książce adresowej, program wyświetli adres tej osoby. W przeciwnym razie program wyświetli informację o tym, że adres wskazanej osoby jest nieznany.

Ćwiczenie 14 – Pętla while

Napisz program, który będzie wyświetlał kolejne napisy wprowadzone przez użytkownika do czasu wprowadzenia napisu **exit**. Jeżeli użytkownik napisze **exit** program zakończy swoje działanie.

Przykładowe wywołanie:

```
C:\Users\Ania\PycharmProjects\testtt>python zad14.py
Napisz 'exit' jeżeli chcesz wyjść z programu
> ala
Napisałes 'ala'
Napisz 'exit' jeżeli chcesz wyjść z programu
> kot
Napisałes 'kot'
Napisz 'exit' jeżeli chcesz wyjść z programu
> exit
Napisałes 'exit'

C:\Users\Ania\PycharmProjects\testtt>
```

Ćwiczenie 15 – Pętla for

Mamy listę słów, np:

```
l = ['kot', 'elementarz', 'okno', 'komputer']
```

Napisz program, który wyświetli kolejne elementy listy wraz z informacją o ich długości.

Wywołanie programu dla listy l:

```
C:\Users\Ania\PycharmProjects\testtt>python zad15.py
Słowo kot ma długość 3
Słowo elementarz ma długość 10
Słowo okno ma długość 4
Słowo komputer ma długość 8
```

Ćwiczenie 16 – Funkcja range

Przed wykonaniem poniższego kodu w konsoli Python spróbuj odgadnąć wynik:

```
range(3)
list(range(0, 3))
list(range(1, 10, 2))
list(range(0, -10, -1))
range(5) == range(0, 5) == range(0, 5, 1)

for iteration in range(3):
    print(f"Iteracja {iteration}")
```

Ćwiczenie 17 – Instrukcje break i continue

Mamy listę wyrazów, np:

```
l2 = ["spam", "stół", "spam", "brązowy", "powietrze", "maleware", "spam",
"koniec"]
```

Napisz program, który będzie wyświetlał kolejne elementy listy, jeżeli nie mają wartości "spam". Dodatkowo, jeżeli element listy ma wartość "maleware" program powinien natychmiast przerwać swoje działanie.

Wywołanie programu dla listy l2:

```
C:\Users\Ania\PycharmProjects\testtt>python zad17.py
stół
brązowy
powietrze
```

Ćwiczenie 18 – Funkcje

Napisz funkcję, która zlicza ilość wystąpień poszczególnych liter w słowie. Funkcja, na wejściu powinna przyjmować słowo, a na wyjściu zwracać słownik (litera → liczba wystąpień) wskazujący ile razy pojawiła się w słowie każda z liter tego słowa.

Przykładowe wywołanie funkcji:

```
print(count_letters("koniokrad"))
```

powinno zwrócić:

```
{'k': 2, 'o': 2, 'n': 1, 'i': 1, 'r': 1, 'a': 1, 'd': 1}
```