Zaawansowane programowania w Pythonie
Zadania do wykładu 2

dr Agnieszka Zbrzezny

29.10.2024

1 Argumenty funkcji – quiz

1. Co wyświetla poniższy kod i dlaczego?

```
def func(a, b = 4, c = 5):
    print(a, b, c)

func(1, 2)
```

2. Co wyświetla poniższy kod i dlaczego?

```
def func(a, b, c = 5):
    print(a, b, c)

func(1, c = 3, b = 2)
```

3. Co wyświetla poniższy kod i dlaczego?

```
def func(a, *args):
    print(a, args)

func(1, 2, 3)
```

4. Co wyświetla poniższy kod i dlaczego?

```
def func(a, **kwargs):
    print(a, kwargs)

func(a = 1, c = 3, b = 2)
```

5. Jaki będzie wynik poniższego kodu i dlaczego?

```
def func(a, b, c = 3, d = 4):
    print(a, b, c, d)

func(1, *(5, 6))
```

2 Argumenty funkcji – zadania programistyczne

1. Napisz funkcję `mult`, która przyjmuje jeden argument tylko pozycyjny. Załóż, że ten argument jest niepustym obiektem iterowalnym typu krotka, lista, zbiór lub zakres. Jako wynik funkcja zwraca iloczyn elementów swojego argumentu. W funkcji `main` przetestuj działanie funkcji `mult`.

   Przykładowo, oba poniższe wywołania

   ```
   mult([3, 5, 7]
   ```

   ```
   mult(range(2, 8, 2)
   ```

   powinny zwrócić liczbę `105`.

2. Napisz funkcję `mult_ints`, która przyjmuje jeden argument tylko pozycyjny. Załóż, że ten argument jest niepustym obiektem iterowalnym typu krotka, lista, zbiór lub zakres. Jako wynik zwraca iloczyn tych argumentów, które są typu całkowitego. W funkcji `main` przetestuj działanie funkcji `mult_ints`.

   Przykładowo, wywołanie

   `mult_ints(3, 3.14, 5, "abc", 7)` powinno zwrócić liczbę `105`.

3. Napisz funkcję `multiply`, która przyjmuje dowolną liczbę argumentów nienazwanych, a jako wynik zwraca iloczyn tych argumentów. W funkcji `main` przetestuj działanie funkcji `multiply`.

   Przykładowo, wywołanie

   `multiply(3, 5, 7)` powinno zwrócić liczbę `105`.

4. Napisz funkcję `multiply_ints`, która przyjmuje dowolną liczbę argumentów nienazwanych, a jako wynik zwraca iloczyn tych argumentów, które są typu całkowitego. W funkcji `main` przetestuj działanie funkcji `multiply_ints`.

   Przykładowo, wywołanie

   `multiply_ints(3, 3.14, 5, "abc", 7)` powinno zwrócić liczbę `105`.

5. Napisz funkcję `make_car`, która przyjmuje dwa obligatoryjne argumenty: `firma` i `model`, oraz dowolną liczbę argumentów nazwanych. Funkcja `make_car` jako swój wynik zwraca słownik opisujący konkretny samochód. Przykładowo:

   ```
   {"firma": "Kia", "model": "Picanto", "kolor": "cafe
   mocca", "poj_silnika": 900}
   ```

   W funcji `main` utwórz listę kilku samochodów opisanych przez różne możliwe cechy.

3 Listy składane

1. Znajdź wszystkie liczby z przedziału `1-1000`, które są podzielne przez `7`.

2. Znajdź wszystkie liczby z przedziału `1-1000`, które mają w sobie cyfrę `3`.

3. Policz liczbę spacji w ciągu znaków.

4. Znajdź wszystkie słowa w ciągu, które mają mniej niż cztery litery.

5. Znajdź wspólne liczby w dwóch listach (bez użycia krotek lub zbiorów):

   `lista_a = [1, 2, 3, 4]` oraz `lista_b = [2, 3, 4, 5]`.

6. Napisz program, który stosując konstrukcję list składanych utworzy listę liczb całkowitych z przedziału `[1..n]` podzielnych przez 3 lub podzielnych przez 5, gdzie `n` jest liczbą podaną jako pierwszy argument programu

7. Napisz program, w którym stosując konstrukcję list składanych utwórz `n`-elementową listę `liczby` (gdzie `n` jest liczbą podaną jako pierwszy argument programu) liczb typu `float` z przedziału `[0..100)` zaokraglonych do dwóch miejsc po przecinku (użyj funkcji `random.uniform`).

   Następnie wypisz na ekran listę `liczby` oraz stosując konstrukcję list składanych dla każdego z poniższych punktów napisz **jedną** instrukcję wypisującą:

   (a) listę dodatnich liczb z listy `lista`;

   (b) listę dodatnich liczb z listy `lista` przekształconych do typu `int`;

   (c) listę wartości funkcji `math.floor` dla liczb z listy `lista`;

   (d) listę wartości funkcji `math.ceil` dla liczb z listy `lista`;

   (e) listę wartości funkcji `math.log` dla dodatnich liczb z listy `lista`.

   Przykładowo, dla `n = 10` i listy

   ```
   [7.78, -9.64, 3.39, -8.64, -3.22, 9.6, 9.5, -7.7, -5.73,
   7.75]
   ```

   program powinien wypisać następujące linie:

   ```
   [7.78, 3.39, 9.6, 9.5, 7.75]
   [7, 3, 9, 9, 7]
   [8, 4, 10, 10, 8]
   [2.052, 1.221, 2.262, 2.251, 2.048]
   ```

4 Listy składane – zadania dodatkowe

1. Utwórz listę wszystkich spółgłosek w ciągu

   `"Żółte jaki lubią krzyczeć i ziewać, a wczoraj jodłowały podczas jedzenia batatów"`.

2. Pobierz indeks i wartość jako tuple dla elementów z listy

   `["hi", 4, 8.99, "apple", ("t,b", "n")]`. Wynik będzie wyglądał tak: `[(indeks, wartość), (indeks, wartość), ...]`.

3. Znajdź tylko liczby w zdaniu typu: "W 1984 roku w 13 przypadkach doszło do protestu, w którym wzięło udział ponad 1000 osób".

4. Dla dowolnej list `liczby`, utwórz listę zawierającą słowo "even", jeśli liczba w liście `liczby` jest parzysta, oraz słowo "odd", jeśli liczba jest nieparzysta. Wynik dla listy `[6, 8, 11]` będzie wyglądał następująco:

   `["odd", "odd", "even"]`.

5. Wyprodukuj listę krotek składającą się tylko z pasujących liczb z list

   `lista_a = [1, 2, 3, 4, 5, 6, 7, 8, 9]`

   oraz

   `lista_b = [2, 7, 1, 12]`.

   Wynik będzie wyglądał tak: `(4, 4), (12, 12)`.

6. Użyj listy zagnieżdżonej, aby znaleźć wszystkie liczby z zakresu `1-1000`, które są podzielne przez dowolną cyfrę inną niż `1 (2-9)`.