UNIWERSYTET EKONOMICZNY W POZNANIU

UNIWERSYTET EKONOMICZNY W POZNANIU · KATEDRA INFORMATYKI EKONOMICZNEJ

Algorytmy i struktury danych

Zajęcia 3 · słowniki, krotki i zbiory

rok akademicki 2026/2027 · dr inż. Szczepan Górtowski

Po tych zajęciach

- budujesz słownik, zmieniasz go i bezpiecznie czytasz przez get
- wiesz, jak słownik znajduje klucz w O(1): hash, szuflady, kolizje
- używasz krotki jako rekordu i klucza — i znasz jej pułapkę
- liczysz sumę, część wspólną i różnice zbiorów
- wybierasz list, tuple, set albo dict do zadania

Dziś: trzy kolekcje, które uzupełniają listę

Notatnik 03_Slowniki_krotki_zbiory.ipynb — wersja z ramkami „Przewidź” i nowymi sekcjami.

01	02	03	04	05	06
Słownik	**Pod maską**	**Kartki**	**Krotka**	**Zbiór**	**Zadania**
klucz → wartość	hash i szuflady	Szafa z szufladami	niemutowalna	unikalne, Venn	3.0–3.4

Ćw. 0 · Przewidź, zanim uruchomisz

5 min

1. Przepisz cztery linie print na kartkę.
2. Obok każdej zapisz, co wypisze Python.
3. Uruchom kod w nowej komórce i policz trafienia.

KOD DO PRZEWIDZENIA

```
d = {"a": 1, "b": 2} | {"b": 9}
print(d['b'])
print(len({1, 1.0, True}))
print(type({}))
print(type((1)), type((1,)))
```

PYTANIE

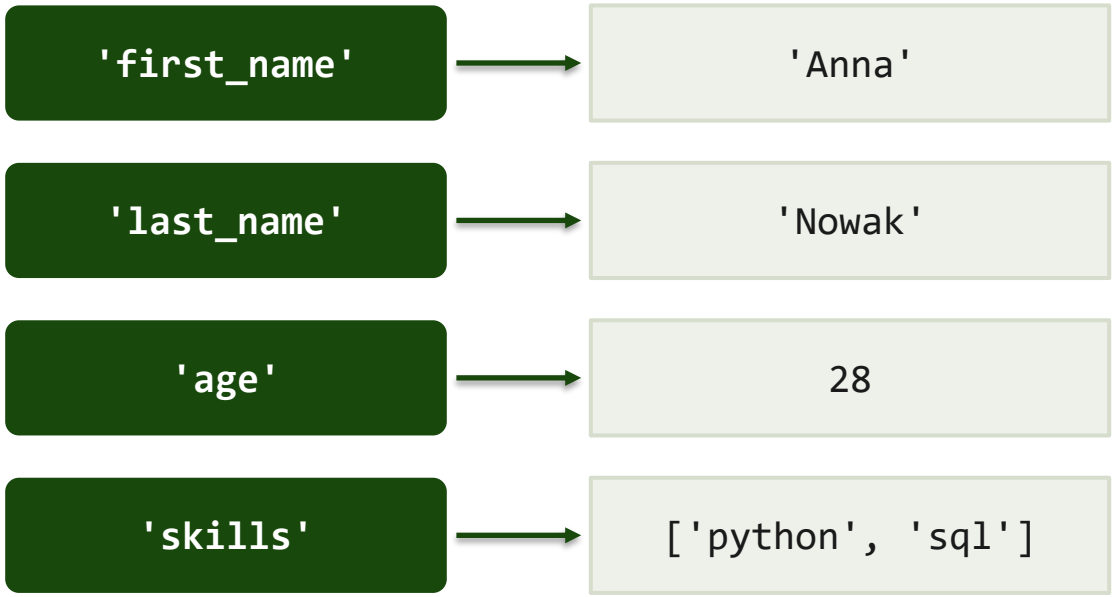
Ile elementów ma zbiór {1, 1.0, True}?

01

Słownik: klucz prowadzi prosto do wartości

Słownik łączy klucz z wartością: person["first_name"] → 'Anna'

Sekcja 1. Wartością może być cokolwiek — także lista. Kluczem tylko coś haszowalnego.




```python
person = {
    "first_name": "Anna",
    "last_name": "Nowak",
    "age": 28,
    "skills": ["python", "sql"],
}
print(person["first_name"])
print(person)
```

WYNIK

```
Anna
{'first_name': 'Anna', 'last_name': 'Nowak', 'age': 28, 'skills': ['python', 'sql'
```

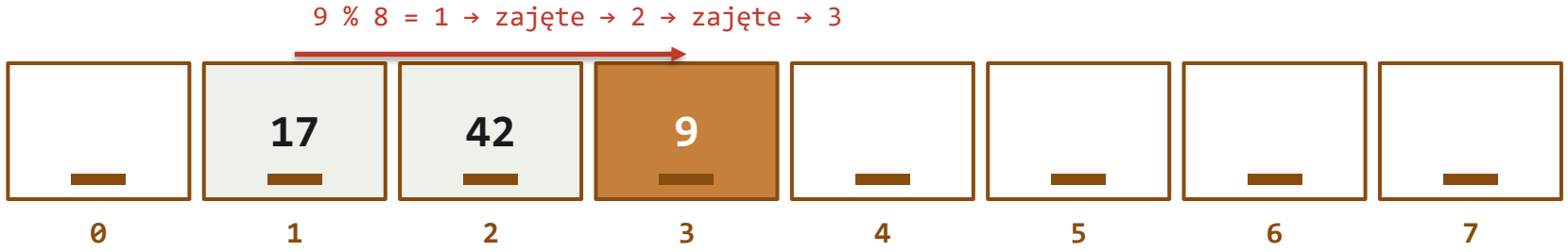
Słownik nie szuka po kolei — liczy, w której szufladzie leży klucz

Szuflada = hash(klucz) % liczba szuflad. Dla liczb całkowitych hash to sama liczba.

17 % 8 = 1

42 % 8 = 2

9 % 8 = 1



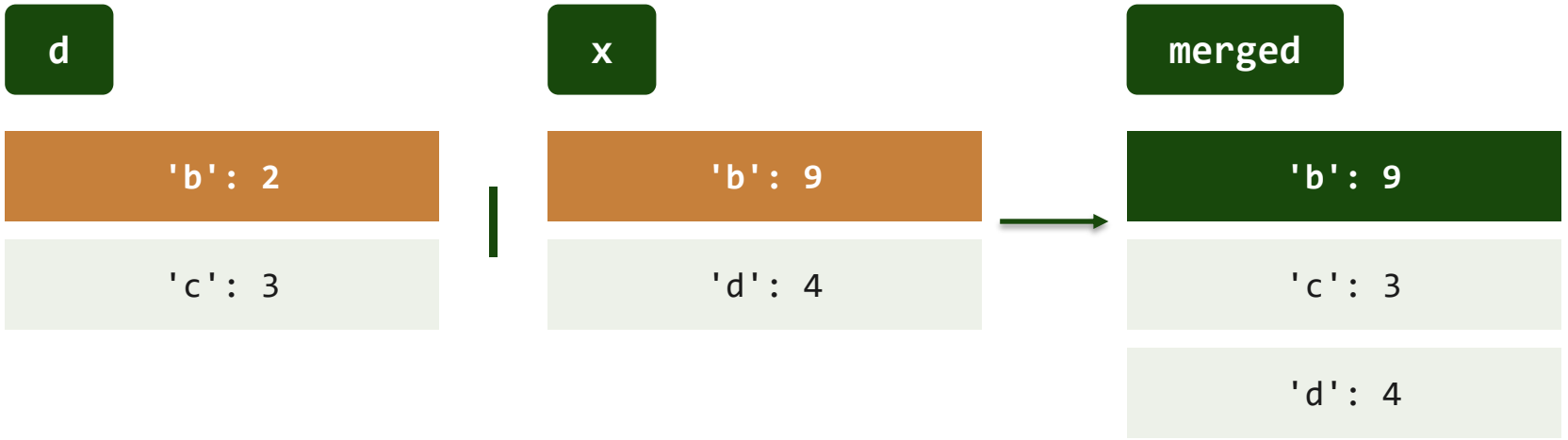

```
SZUFLAD = 8
for klucz in [17, 42, 9, "Anna", "Jan"]:
    h = hash(klucz)
    print(f"{klucz!r:<7} hash = {h:>21}  → szuflada {h % SZUFLAD}")
```

WYNIK

```
17      hash =                   17  → szuflada 1
42      hash =                   42  → szuflada 2
9       hash =                    9  → szuflada 1
'Anna'  hash =  -8487713105859407710  → szuflada 2
'Jan'   hash =   2080564901061387758  → szuflada 6
```

d | x łączy słowniki — przy wspólnym kluczu wygrywa prawy

Sekcja 3. | tworzy nowy słownik (Python 3.9+), update zmienia istniejący.



WYNIK KOMÓRKI

```
{'a': 1, 'b': 2, 'c': 3}
{'b': 2, 'c': 3}
{'b': 9, 'c': 3, 'd': 4}
{'b': 2, 'c': 3, 'e': 5}
```

TRZY SPOSOBY ZMIANY

d[k] = v dodaje albo nadpisuje · del d[k] i d.pop(k) usuwają · d.update(x) dokleja x do d

d["age"] rzuca KeyError, d.get("age", ...) zwraca wartość domyślną

Sekcja 5: bezpieczny dostęp i zagnieżdżone słowniki.

```
person = {"name": "Basia", "meta": {"city": "Poznań"}}
print(person.get("age", "brak danych"))
meta = person.get("meta", {})
print(meta.get("city", "nieznane"))

try:
    print(person["age"])
except KeyError as e:
    print("KeyError:", e)
```

WYNIK

```
brak danych
Poznań
KeyError: 'age'
```

KIEDY CO

d[k] — klucz musi być. d.get(k, domyślna) — klucza może nie być.

dict.fromkeys: ok. 520× szybciej niż pętla z zajęć 2

Od Pythona 3.7 słownik pamięta kolejność kluczy — to część języka, nie przypadek.

```
lista = [3, 1, 3, 2, 1]
print(list(dict.fromkeys(lista)))
```

WYNIK

```
[3, 1, 2]
```



DLACZEGO

x not in wynik przegląda listę — O(n) dla każdego elementu. Słownik sprawdza klucz w O(1).

Pomiar w komórce z sekcji 6: lista 10 000 liczb (5 000 różnych), Python 3.11.9. docs.python.org/3/whatsnew/3.7.html

8 szuflad mieści 5 kluczy: słownik rośnie po zapełnieniu 2/3

Pomiar sys.getsizeof z sekcji 6 zgadza się z regułą z kodu źródłowego CPython.

Szuflad w szafie	Mieści kluczy (2/3)	Słownik rośnie przy
8	5	6. kluczu
16	10	11. kluczu
32	21	22. kluczu
64	42	43. kluczu
128	85	86. kluczu

CPYTHON, DICTOBJECT.C

- PyDict_MINSIZE 8 — najmniejsza szafa
- USABLE_FRACTION(n) = 2n/3 — tyle można zapełnić

PO CO ZAPAS

W pełnej szafie prawie każde wstawienie to kolizja. Wolne szuflady trzymają szukanie blisko O(1).

CPython 3.11, Objects/dictobject.c (sprawdzone 8.10.2026); pomiar: Python 3.11.9.

Klucz do ćw. 0: {} to słownik, a {1, 1.0, True} ma jeden element

Wyniki z uruchomienia kodu. Każda odpowiedź wraca na slajdach dalej.

Kod	Wynik	Dlaczego
`d['b']`	`9`	przy | wygrywa słownik z prawej
`len({1, 1.0, True})`	`1`	1 == 1.0 == True i ten sam hash
`type({})`	`<class 'dict'>`	{} to pusty słownik
`type((1)), type((1,))`	`<class 'int'> <class 'tuple'>`	krotkę tworzy przecinek

02

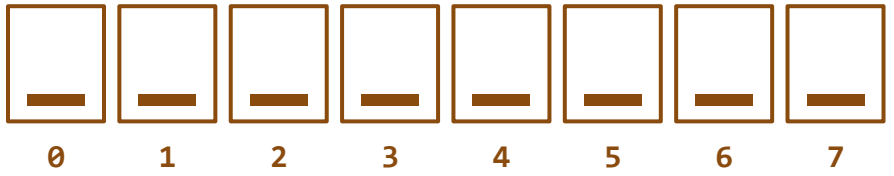
Ćwiczenie na kartkach: Szafa z szufladami

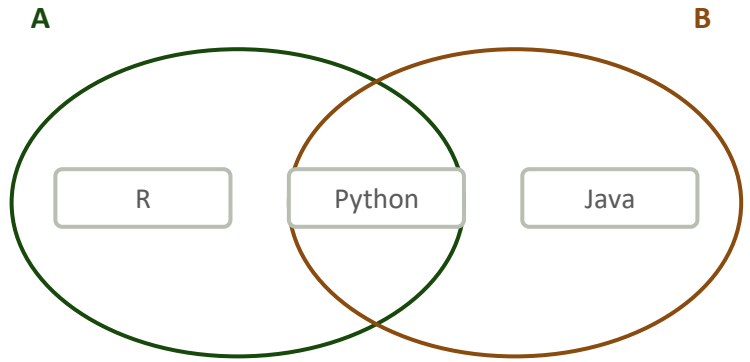
Ćw. 1 · Szafa z szufladami — bez komputera

15 min

1. Grupy 2–3 osób: karta A4 z szafą i diagramem Venna, karty do wycięcia.
2. Runda 1: wkładajcie klucze do szuflad; zajęte — idźcie do następnej.
3. Runda 2: szukajcie kluczy i liczcie otwarte szuflady.
4. Runda 3: układajcie języki Ani i Bartka na diagramie Venna.

```
szuflada = klucz % 8
```



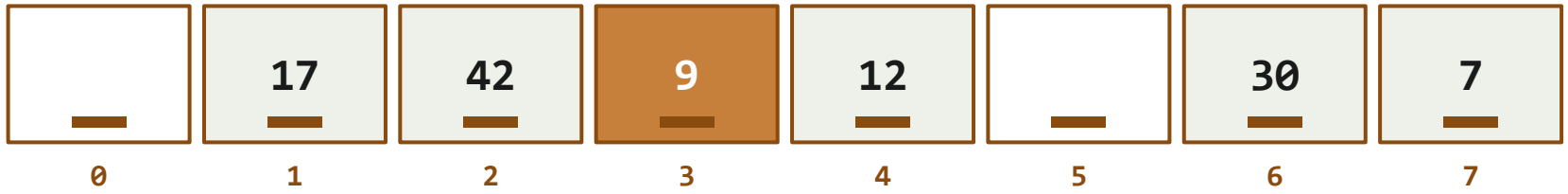


PYTANIE

Ile szuflad otworzycie, szukając 33, którego w szafie nie ma?

Klucz rund 1–2: jedna kolizja, szukanie po kilku szufladach

Szafa po rundzie 1. Na pomarańczowo klucz, który trafił do zajętej szuflady.



Klucz	k % 8	Szuflada	Sprawdzone
17	1	1	1
42	2	2	1
9	1	3	3
30	6	6	1
12	4	4	1
7	7	7	1

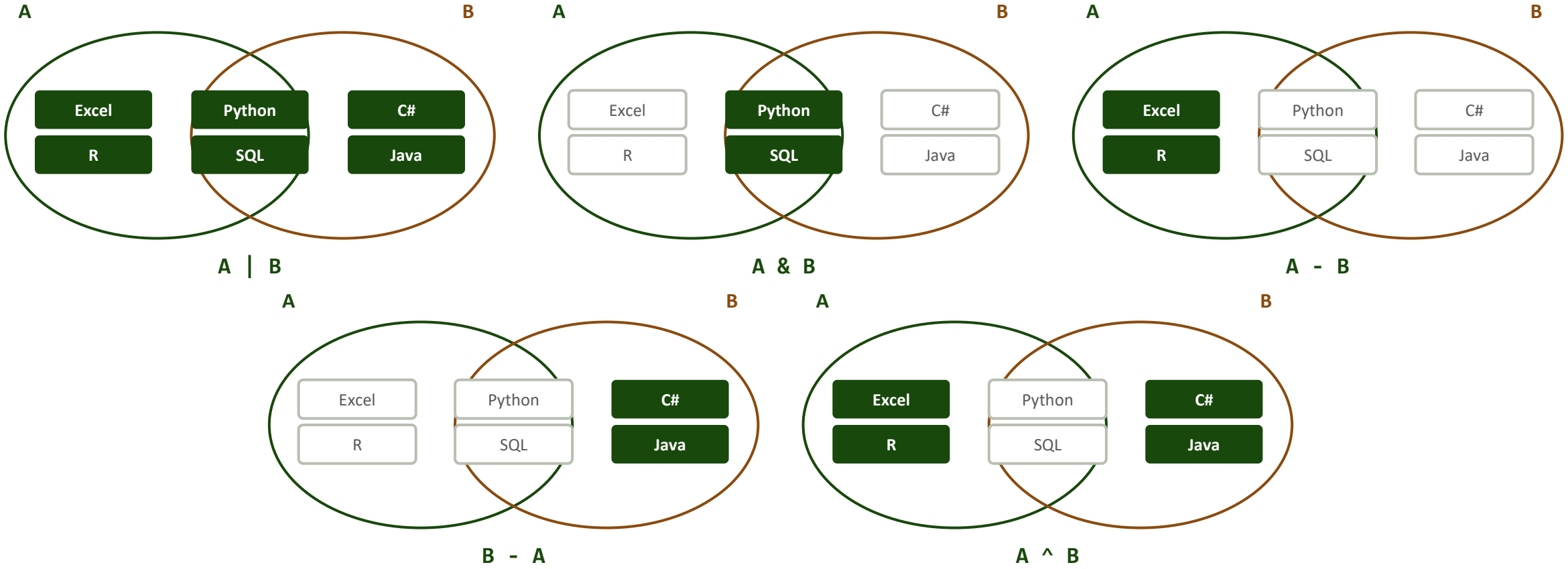
Szukasz	Jest?	Szafa	Lista
9	tak	3	3
30	tak	1	4
33	nie	5	6

RUNDA 2: WSTAW 25

25 % 8 = 1 → szuflada 5, sprawdzone 5. Szafa pełna w 7/8 — Python powiększyłby ją już przy 6. kluczu.

Klucz rundy 3: & to wspólne, ^ to wszystko poza częścią wspólną

Języki Ani (A) i Bartka (B) z sekcji 8 notatnika. Na zielono elementy wyniku.



03

Krotka: lista, której nie zmienisz

Krotka to lista, której nie zmienisz — i 80 B zamiast 104 B

Sekcja 7: rozpakowanie, zamiana zmiennych, funkcja zwracająca kilka wartości.

```python
print(type((1)), type((1,)))

a, b = 1, 2
a, b = b, a                        # zamiana bez zmiennej pomocniczej
print("a =", a, "b =", b)

def min_max(liczby):
    return min(liczby), max(liczby)   # funkcja zwraca krotkę

najmniejsza, najwieksza = min_max([4, 8, 1, 9])
print(najmniejsza, najwieksza)

krotka = (1, 2, 3, 4, 5)
lista_5 = [1, 2, 3, 4, 5]
print("krotka:", sys.getsizeof(krotka), "B")
print("lista: ", sys.getsizeof(lista_5), "B")
```

WYNIK

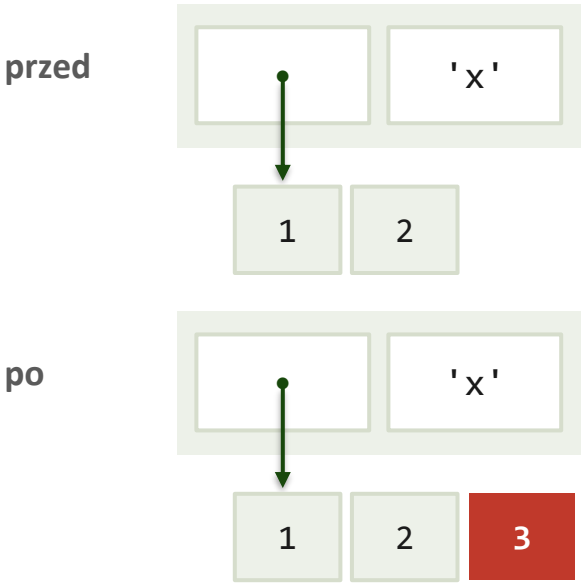
```
<class 'int'> <class 'tuple'>
a = 2 b = 1
1 9
krotka: 80 B
lista:  104 B
```

KROTKA, GDY

- dane nie powinny się zmieniać: współrzędne, data
- funkcja zwraca kilka wartości
- potrzebujesz klucza złożonego: (rok, miesiąc)

t[0] += [3] kończy się błędem — a lista w krotce i tak urosła

Pułapka z oficjalnego FAQ Pythona: += najpierw zmienia listę, potem próbuje przypisać do krotki.



krotka (nie zmienisz) ↓ lista w środku (zmienisz)

```python
t = ([1, 2], "x")
try:
    t[0] += [3]
except TypeError as e:
    print("TypeError:", e)
print(t)

try:
    {t: "wartość"}
except TypeError as e:
    print("krotka z listą jako klucz → TypeError:", e)
```

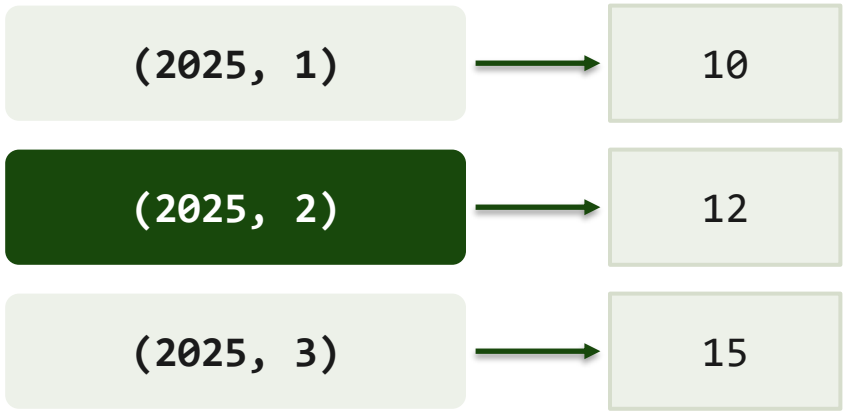
WYNIK

```
TypeError: 'tuple' object does not support item assignment
([1, 2, 3], 'x')
krotka z listą jako klucz
  → TypeError: unhashable type: 'list'
```

docs.python.org/3/faq/programming.html — „Why does a_tuple[i] += ['item'] raise an exception when the addition works?" (sprawdzone 8.10.2026).

Krotka może być kluczem: pomiar[(2025, 2)] — lista nie może

Klucz musi mieć stały hash. Lista może się zmienić, więc Python jej nie przyjmie.




```
pomiar[(2025, 2)]
{[2025, 2]: 12}   # lista jako klucz
```

WYNIK

```
12
TypeError: unhashable type: 'list'
```

ZADANIE 3.1 I 3.4

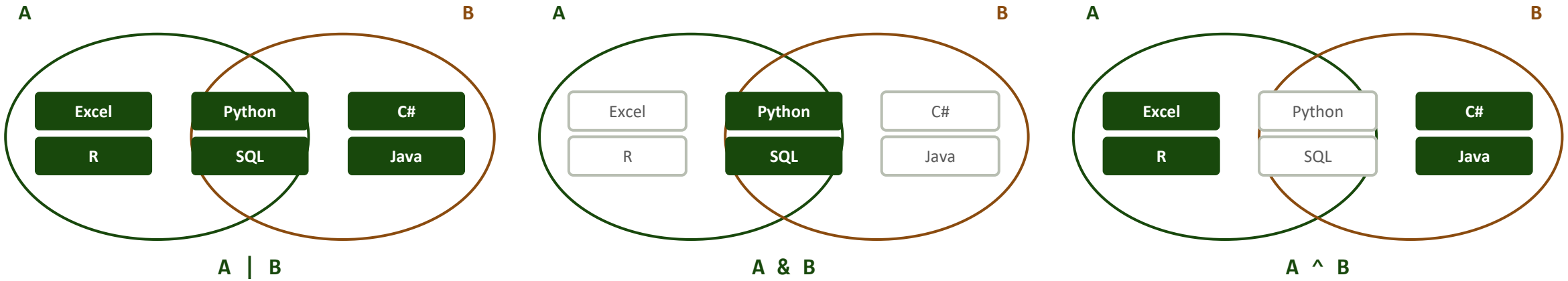
Zadanie 3.1 robi dokładnie taki słownik, a 3.4 używa krotki liter jako klucza grupy anagramów.

04

Zbiór: unikalne elementy bez kolejności

Zbiór działa jak w matematyce: |, &, - i ^

Sekcja 8. Duplikat „Python” w zbiorze Ani znika przy tworzeniu zbioru.




```
jezyki_ani = {"Python", "SQL", "Excel", "R", "Python"}
jezyki_bartka = {"SQL", "Python", "Java", "C#"}
```

SŁOWAMI

- | — co zna ktokolwiek · & — co znają oboje
- - — tylko A · ^ — dokładnie jedno z nich

{1, 1.0, True} to jeden element, a {} to pusty słownik

Dwie pułapki zbiorów. Pusty zbiór tworzysz przez set().

```
print({1, 1.0, True})
print({1: "a", True: "b", 1.0: "c"})
print(type({}), type(set()))
```

WYNIK

```
{1}
{1: 'c'}
<class 'dict'> <class 'set'>
```

DLACZEGO

- 1 == 1.0 == True
- hash(1) == hash(1.0) == hash(True)
- zbiór i słownik widzą jeden klucz

Ten sam zbiór, trzy kolejności: hash napisów zależy od ziarna

Sekcja 8: kod uruchomiony w trzech procesach Pythona z różnym PYTHONHASHSEED.

PYTHONHASHSEED=1	`'Python'`	`'SQL'`	`'Excel'`	`'R'`	`'Java'`
PYTHONHASHSEED=2	`'Python'`	`'Excel'`	`'Java'`	`'SQL'`	`'R'`
PYTHONHASHSEED=3	`'Python'`	`'R'`	`'Java'`	`'Excel'`	`'SQL'`

WNIOSEK

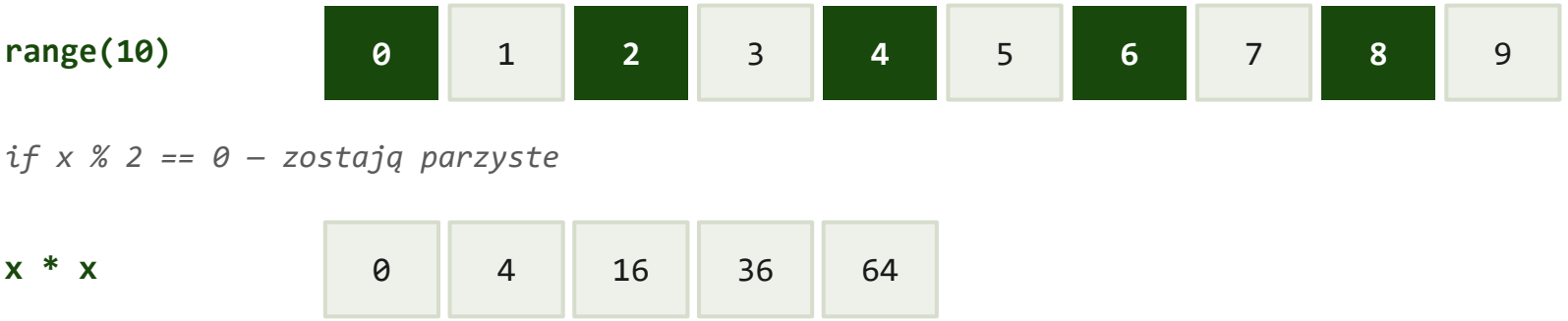
Nie polegaj na kolejności zbioru. Potrzebujesz kolejności — posortuj (sorted) albo użyj listy lub słownika.

05

Comprehensions, wybór struktury i JSON

Comprehension: pętla, warunek i wynik w jednej linii

Sekcja 9. Ta sama pętla co z append, zapisana krócej.




```
set_comp = {x * x for x in range(10) if x % 2 == 0}
dict_comp = {x: x * x for x in range(5)}
print(set_comp)
print(dict_comp)
```

WYNIK

```
{0, 64, 4, 36, 16}
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
```

Wybór struktury: kolejność, zmiany, duplikaty, szukanie

Sekcja 10 notatnika. * krotka — jeśli wszystkie jej elementy są haszowalne.

	list	tuple	set	dict
kolejność	tak	tak	nie	tak (od 3.7)
zmiana po utworzeniu	tak	nie	tak	tak
duplikaty	tak	tak	nie	klucze — nie
x in ...	O(n)	O(n)	O(1)	O(1) po kluczu
kluczem słownika	nie	tak*	nie	nie

JSON to słownik w podróży: tak odpowiada wiele API

Sekcja 11. json.dumps zamienia słownik na tekst, json.loads — z powrotem.

```python
import json

person = {"first_name": "Anna", "age": 28,
          "skills": ["python", "sql"], "student": True}
tekst = json.dumps(person, ensure_ascii=False)
print(tekst)
z_powrotem = json.loads(tekst)
print(z_powrotem["skills"][0], z_powrotem == person)
```

WYNIK

```
{"first_name": "Anna", "age": 28, "skills": ["python", "sql"], "student": true}
python True
```

ZAUWAŻ

True z Pythona w JSON to true. Po loads dostajesz identyczny słownik.

06

Zadania 3.0–3.4: test powie, czy działa

Ćw. 2 · Zadania 3.0 i 3.1: słownik i krotka

8 min

1. Zadanie 3.0: słownik produkty — dodaj id_3, usuń id_2, zmień cenę id_1 na 9.99, użyj get.
2. Zadanie 3.1: krotka punkt, rozpakowanie do x i y, słownik pomiar z kluczami (rok, miesiąc).
3. Po każdym zadaniu uruchom komórkę Sprawdź.

PYTANIE

Co zwróci produkty.get("id_999", 0.0)?

Ćw. 3 · Zadania 3.2 i 3.3: zbiory i comprehensions

9 min

1. Zadanie 3.2: zbiory A i B z częścią wspólną; policz |, &, -, ^ i sprawdź 5 in A.
2. Zadanie 3.3: set_comp — kwadraty parzystych 0–10; dict_comp — x: x**2 dla nieparzystych z range(6).
3. Uruchom komórki Sprawdź.

WZÓR ROZWIĄZANIA 3.3

```
set_comp = {x ** 2 for x in range(11) if x % 2 == 0}
dict_comp = {x: x ** 2 for x in range(6) if x % 2 == 1}
```

PYTANIE

Ile elementów ma set_comp?

Ćw. 4 · Zadanie 3.4 dla chętnych: anagramy

dom

1. Klucz grupy: tuple(sorted(slowo)) — litery słowa posortowane.
2. Nowy klucz: pusta lista. Potem append słowa do jego grupy.
3. Zwróć słownik grup.

SEKCJA 12 — SZABLON (SKRÓCONY)

```
def grupuj_anagramy(slowa: list) -> dict:
    """Słowa z tych samych liter trafiają do jed…"""
    # TODO: klucz = tuple(sorted(slowo))

    return
```

PYTANIE

Dlaczego kluczem jest krotka, a nie lista liter?

Rezultat zadania 3.4: 3 grupy, krotka liter jako klucz

Przykładowe rozwiązanie i wynik dla listy z testu.

```python
def grupuj_anagramy(slowa: list) -> dict:
    """Słowa z tych samych liter trafiają do jednej grupy."""
    grupy = {}
    for slowo in slowa:
        klucz = tuple(sorted(slowo))
        if klucz not in grupy:
            grupy[klucz] = []
        grupy[klucz].append(slowo)
    return grupy
```

KOMÓRKA SPRAWDŹ

```
✅ Zadanie 3.4 zaliczone!
```

SPRAWDŹ U SIEBIE

- kot → ['kot', 'tok', 'okt']
- aal → ['ala']
- aall → ['lala']

Na następne zajęcia

- Dokończ zadania 3.0–3.3; komórki Sprawdź mają pokazać ✅
- Odpowiedz na trzy pytania z końca notatnika: lista jako klucz, milion numerów albumów, setdefault
- Uruchom komórkę z PYTHONHASHSEED u siebie — czy kolejność wyszła jak na slajdzie?
- Dla chętnych: zadanie 3.4 z anagramami

Trzy rzeczy do zapamiętania

O(1)

słownik i zbiór liczą szufladę z hasha zamiast szukać

(1,)

krotkę tworzy przecinek; może być kluczem

set()

pusty zbiór — bo {} to pusty słownik

DZIĘKUJĘ

dr inż. Szczepan Górtowski

Katedra Informatyki Ekonomicznej · Uniwersytet Ekonomiczny w Poznaniu

szczepan.gortowski@ue.poznan.pl · dyżury: kie.ue.poznan.pl/pl/dyzury/