

PROJEKT #1 ROBIMY REMONT!

Pomocnik malarza



W pracy malarza jest wiele trudności, a najgorsze jest zapamiętanie i obliczenie tych wszystkich danych... Dlatego stworzymy narzędzie, które wspomóże pracę każdego majstra!

```
Ile jest ścian do pomalowania? 3
Jeśli chcesz przyjąć poprzednią wysokość to wciśnij enter.
Podaj szerokość 1 ściany: 2
Podaj wysokość 1 ściany: 4
Podaj szerokość 2 ściany: 3
Podaj wysokość 2 ściany:
Podaj szerokość 3 ściany: 2
Podaj wysokość 3 ściany:
Powierzchnia do pomalowania wynosi 24.0
Ilość warstw gruntu: 2
Ilość warstw farby: 3
Potrzebujesz 10 litrów gruntu.
Potrzebujesz 6 litrów farby.
```

Na 1 malowanie

wydajność gruntu 5 mkw = 1 litr
wydajność farby 13 mkw = 1 litr

PyStart #13 Listy — trochę lepsze tuple

Cechy dobrej Listy



→ Do deklaracji można użyć nawiasów kwadratowych

```
>> fruits = ['mango', 'peach', 'lemon', 'grapefruit', 'apple', 'banana', 'cherry']  
>> fruits  
['mango', 'peach', 'lemon', 'grapefruit', 'apple', 'banana', 'cherry']
```

PyStart #13 Listy — trochę lepsze tuple

Cechy dobrej Listy



- Do deklaracji można użyć nawiasów kwadratowych
- Do rzutowania używamy funkcji `list()`

```
>> values = (1, 2, 3, 7, 8, 9)
>> list(values)
[1, 2, 3, 7, 8, 9]
```

PyStart #13 Listy — trochę lepsze tuple

Cechy dobrej Listy



- Do deklaracji można użyć nawiasów kwadratowych
- Do rzutowania używamy funkcji `list()`
- Indeksowana -> element wskazywany liczbą

```
>> books = ['Ogniem i mieczem', 'W pustyni i w puszczy', 'Pan Wołodyjowski']  
>> books[0]  
'Ogniem i mieczem'  
>> books[-1]  
'Pan Wołodyjowski'
```

PyStart #13 Listy — trochę lepsze tuple

Cechy dobrej Listy



- Do deklaracji można użyć nawiasów kwadratowych
- Do rzutowania używamy funkcji `list()`
- Indeksowana -> element wskazywany liczbą
- Jest mutowalna
- Kolejność jest zachowana

PyStart #13 Listy — trochę lepsze tuple

Mutowalność listy



→ Jakie możemy na liście wykonywać operacje? Zwiększanie, podstawianie

```
>> books = ['Ogniem i mieczem', 'W pustyni i w puszczy', 'Pan Wołodyjowski']
>> books[0] = 'Quo Vadis'
>> books[-1] = 'Janko Muzykant'
>> books
['Quo Vadis', 'W pustyni i w puszczy', 'Janko Muzykant']

>> books.append('Latarnik')
>> books
['Quo Vadis', 'W pustyni i w puszczy', 'Janko Muzykant', 'Latarnik']
>> 'Latarnik' in books
```

PyStart #13 Listy — trochę lepsze tuple

Usuwanie wartości z listy



Aby usunąć wartość z listy nie trzeba znać jej indeksu

```
>> fruits = ['mango', 'peach', 'lemon', 'grapefruit', 'apple', 'banana', 'cherry']
>> fruits.remove('mango')
>> fruits
['peach', 'lemon', 'grapefruit', 'apple', 'banana', 'cherry']
```

Lub usuwanie na podstawie indeksu

```
>> fruits = ['mango', 'peach', 'lemon', 'grapefruit', 'apple', 'banana', 'cherry']
>> del fruits[1]
>> fruits
['mango', 'lemon', 'grapefruit', 'apple', 'banana', 'cherry']
```

PyStart #13 Listy — trochę lepsze tuple

Cechy dobrej Listy



- Do deklaracji można użyć nawiasów kwadratowych
- Do rzutowania używamy funkcji `list()`
- Indeksowana -> element wskazywany liczbą
- Jest mutowalna
- Kolejność jest zachowana

PyStart #13 Listy — trochę lepsze tuple

Zadania dla nabrania wprawy

W interaktywnym środowisku Pythona


1. Stwórz listę zawierającą trzy nazwy zespołów, których słuchasz.
2. Wymień pierwszy element na inny zespół.
3. Usuń pierwszy element.
4. Korzystając z `append` dodaj oba zespoły na sam koniec.
5. Sprawdź, czy dowolny z zespołów znajduje się na liście, spróbuj zrobić to samo z zespołem, którego na tej liście nie ma.
6. Usuń ostatni zespół z listy.
7. Wyświetl wszystkie zespoły.



Zobacz co się stanie, gdy na listę zamienisz dowolny napis

PyStart #14 Listy — inne operacje

Metody listy



```
first_names = ['Asia', 'Basia', 'Wojtek', 'Krysia']

# dodaje nowy element na końcu
first_names.append('Krzysztof')

# dodaje nowy element na pozycji 3
first_names.insert(3, 'Krzysztof')

# zdejmuję jeden element z listy i przypisuje do zmiennej
# jeśli wartość pusta to ostatni, ale można też podać index
last_first_name = first_names.pop()
third_name = first_names.pop(3)


# ostatni i trzeci element zniknęły
print(first_names)

# Tak czyścimy całą listę
```

PyStart #14 Listy — inne operacje

Mapowanie

```
['asia', 'baSIa', 'WoJtEk', 'krYSIA']
```



```
['Asia', 'Basia', 'Wojtek', 'Krysia']
```

```
names = ['asia', 'baSIa', 'WoJtEk', 'krYSIA']
proper_names = []

for name in names:
    proper_names.append(name.title())


print(proper_names)
```

PyStart #14 Listy — inne operacje

Filtrowanie



```
['Asia', 'Basia', 'Wojtek', 'Krysia']
```



```
['Wojtek']
```

```
names = ['Asia', 'Basia', 'Wojtek', 'Krysia']
male_names = []

for name in names:
    if not name.endswith('a'):
        male_names.append(name.title())

print(male_names)
```

PyStart #14 Listy — inne operacje

Sort czy Sorted?



- `sorted()` zwraca nową listę, posortowaną
- `sort()` nic nie zwraca, sortuje oryginalną listę
- oba sposoby sortowanie odbierają te same argumenty

```
names = ['Asia', 'Basia', 'Wojtek', 'Krysia']  
for name in sorted(names):  
    print(name)  
  
names.sort(reverse=True)  
for name in names:  
    print(name)
```

PyStart #14 Listy — inne operacje

in - czy jest na liście?



→ in zwraca **True** lub **False** w zależności od tego czy owoc znajduje się na liście

```
liked_fruits = ['mango', 'lemon', 'grapefruit', 'apple', 'banana', 'cherry']  
fruit = 'peach'
```

```
if fruit in liked_fruits:  
    print('Lubię ten owoc!')  
else:  
    print('Tego nie lubię!')
```

PyStart #14 Listy — inne operacje

Operatory i listy




- listy możemy mnożyć razy liczbę (tak jak string)
- listy możemy dodawać do siebie

```
# ['Python', 'C++', 'Java', 'Python', 'C++', 'Java', 'Python', 'C++', 'Java']  
['Python', 'C++', 'Java'] * 3
```

```
# ['Gdynia', 'Gdańsk', 'Sopot', 'Katowice', 'Sosnowiec']  
['Gdynia', 'Gdańsk', 'Sopot'] + ['Katowice', 'Sosnowiec']
```

PyStart #14 Listy — inne operacje

Przechodzenie po dwóch listach na raz



```
countries = ['Polska', 'Niemcy', 'Francja']
capitals = ['Warszawa', 'Berlin', 'Paryż']

print('Państwo - Stolica')
for country, capital in zip(countries, capitals):
    print(country, '-', capital)
```


PyStart #14 Listy — inne operacje

Zadania dla nabrania wprawy

1. Napisz program, który zliczy słowa oraz znaki w podanym przez użytkownika zdaniu
2. Napisz program, w którym zapytasz użytkownika o 5 liczb.
 1. Każdą z liczb dodajemy do listy.
 2. Zwracamy najmniejszy, największy element oraz średnią arytmetyczną wszystkich wprowadzonych liczb.
3. Napisz program, w którym odbierzesz od użytkownika imiona oraz nazwiska rozdzielone przecinkami, w odpowiedzi powinny wyświetlić się poprawnie zapisane imiona bez powtórzeń posortowaną w kolejności **Z-A**.

IN: Adam Mickiewicz, Adam Asnyk, Zbigniew Nienacki

OUT: Zbigniew, Adam

podpowiedzi:

- split
- max
- min
- len
- sort(reverse=True)



PyStart #15 Słowniki, czyli klucze i wartości

Cechy dobrego słownika



- Do deklaracji można użyć nawiasów klamrowych lub `dict()`
- Kluczami najczęściej są stringi, mogą być dowolne obiekty niemutowalne: `int`, `float`, `str`, `tuple`
- Jest mutowalna
- Kolejność elementów nie ma znaczenia

PyStart #15 Słowniki, czyli klucze i wartości

Deklaracja



```
>> character = {'first_name': 'Grzegorz', 'last_name': 'Brzęczyszczkiewicz'}
>> character['first_name']
'Grzegorz'
>> character['last_name']
'Brzęczyszczkiewicz'

>> another_character = dict(first_name='Ambroży', last_name='Kleks')
>> another_character['last_name']
'Kleks'
```

PyStart #15 Słowniki, czyli klucze i wartości

Metoda .get()



```
>> character = {'first_name': 'Grzegorz', 'last_name': 'Brzeczyszczykiewicz'}
```

```
>> character['age']
```

```
KeyError: 'age'
```

```
>> character.get('age')
```

```
None
```

```
>> character.get('age', 21)
```

```
21
```

PyStart #15 Słowniki, czyli klucze i wartości

Metoda .update()



```
>> character = {'first_name': 'Grzegorz', 'last_name': 'Brzęczyszczkiewicz'}
>> tests = {'run': 10, 'shoot': 8, 'hide': 3}
>> character.update(tests)
>> character
{'first_name': 'Grzegorz', 'last_name': 'Brzęczyszczkiewicz', 'run': 10,
'shoot': 8, 'hide': 3}
```

PyStart #15 Słowniki, czyli klucze i wartości

Usunięcie wartości ze słownika



```
>> character = {'first_name':'Grzegorz', 'last_name':'Brzęczyszczkiewicz'}  
>> del character['last_name']  
  
>> character  
{'first_name':'Grzegorz'}
```

PyStart #15 Słowniki, czyli klucze i wartości

Jak iterować po słowniku?



```
1 capitals = {"Poland": "Warsaw", "Maroko": "Rabat", "Estonia": "Tallin"}
2 for country in capitals:
3     print(country)
4     print(capitals[country])
5
6 for country, capital in capitals.items():
7     print(country)
8     print(capital)
9 |
```

PyStart #15 Słowniki, czyli klucze i wartości

Jak zachowuje się in ?



```
>>> capitals = {"Poland": "Warsaw", "Maroko": "Rabat", "Estonia": "Tallin"}
>>> "Poland" in capitals
True
>>> "Warsaw" in capitals
False
>>> "Warsaw" in capitals.values()
True
>>>
```


PyStart #15 Słowniki, czyli klucze i wartości

Jak zachowuje się in ?



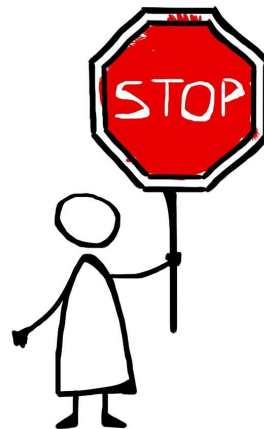
```
>>> capitals = {"Poland": "Warsaw", "Maroko": "Rabat", "Estonia": "Tallin"}
>>> "Poland" in capitals
True
>>> "Warsaw" in capitals
False
>>> "Warsaw" in capitals.values()
True
>>>
```

PyStart #15 Słowniki, czyli klucze i wartości

Zadania dla nabrania wprawy

1. Przygotuj mały słownik języka angielskiego, pytaj użytkownika co chce zrobić i wyświetlaj mu słowo przetłumaczone na język polski lub na język angielski.

2. Przygotuj program, który odbierze od użytkownika dowolny string, a następnie zwróci słownik zawierający informację ile razy wystąpiło każde słowo. Np: "raz raz dwa trzy". powinno zwrócić słownik { "raz": 2, "dwa": 1, "trzy": 1 }



PyStart #16 Łączymy słowniki z listami

Gdzie się to przyda?



Tego typu połączenie przydaje się ogólnie do przechowywania danych. Warto mieć sprawność w żonglowaniu tego typu strukturami.

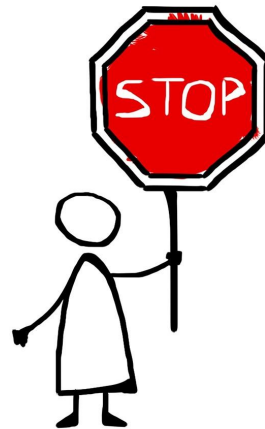
- **Listy** - wszędzie tam gdzie potrzebujemy wielu elementów, indeksowanych
 - ♦ wiersze z bazy danych
 - ♦ dane z pliku
- **Słowniki** - wszędzie tam gdzie potrzebujemy nazywać kolumny
 - ♦ wiersze z bazy danych, ale z nazwami kolumn
 - ♦ dane z plików, które mają kolumny np. csv

PyStart #16 Łączymy słowniki z listami

Zadania dla nabrania wprawy

Korzystając z **list** oraz **słowników** stwórz konstrukcja, które będą przechowywać następujące dane. **Niech ostatnia kolumna będzie listą.**

id	job_title	name	programming_language
1	junior developer	John	python
2	senior developer	Mark	python, php
3	stuff developer	Alex	php, javascript, python
4	junior developer	Bart	javascript, php
5	senior developer	Carl	python, javascript
6	junior developer	Martha	php, javascript

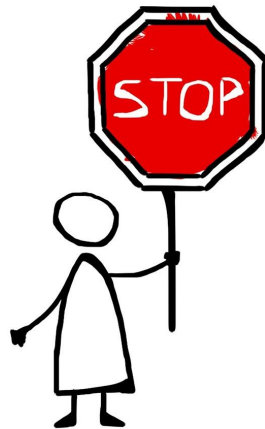


PyStart #16 Łączymy słowniki z listami

Zadania dla nabrania wprawy

Następnie przejdź po tej liście i stwórz następującą strukturę pokazującą osoby pracujące w danych technologiach.

```
{
  'python': {
    'quantity': 4,
    'names': ['John', 'Mark', 'Alex', 'Carl']
  },
  'php': {
    'quantity': 4,
    'names': ['Mark', 'Alex', 'Bart', 'Martha']
  } itd...
}
```



PyStart #17 Pseudolosowość

Informacje wstępne



- Pseudolosowość - “dlaczego pseudo” ?
- Z jakiej biblioteki skorzystamy?
- Pobieranie losowej liczby z przedziału od-do
- Pobieranie losowej wartości z listy

PyStart #17 Pseudolosowość

Z czego skorzystamy?



`random` — Generate pseudo-random numbers

Source code: [Lib/random.py](#)

This module implements pseudo-random number generators for various distributions.

For integers, there is uniform selection from a range. For sequences, there is uniform selection of a random element, a function to generate a random permutation of a list in-place, and a function for random sampling without replacement.

On the real line, there are functions to compute uniform, normal (Gaussian), lognormal, negative exponential, gamma, and beta distributions. For generating distributions of angles, the von Mises distribution is available.

<https://docs.python.org/3/library/random.html>

PyStart #17 Pseudolosowość

Z czego skorzystamy?



random — Generators

Source code: Lib/random.py

This module implements pseudo

For integers, there is uniform selection of a random element, a function to generate a random number without replacement.

On the real line, there are functions for generating gamma, and beta distributions. For

```
random.randrange(stop)
```

```
random.randrange(start, stop[, step])
```

Return a randomly selected element from `range(start, stop, step)`. This is equivalent to `choice(range(start, stop, step))`, but doesn't actually build a range object.

The positional argument pattern matches that of `range()`. Keyword arguments should not be used because the function may use them in unexpected ways.

Changed in version 3.2: `randrange()` is more sophisticated about producing equally distributed values. Formerly it used a style like `int(random()*n)` which could produce slightly uneven distributions.

Deprecated since version 3.10: The automatic conversion of non-integer types to equivalent integers is deprecated. Currently `randrange(10.0)` is losslessly converted to `randrange(10)`. In the future, this will raise a `TypeError`.

Deprecated since version 3.10: The exception raised for non-integral values such as `randrange(10.5)` or `randrange('10')` will be changed from `ValueError` to `TypeError`.

```
random.randint(a, b)
```

Return a random integer N such that $a \leq N \leq b$. Alias for `randrange(a, b+1)`.

<https://docs.python.org/3/library/random.html>

PyStart #17 Pseudolosowość

Z czego skorzystamy?

`random.choice(seq)`

Return a random element from the non-empty sequence `seq`. If `seq` is empty, raises `IndexError`.

`random.shuffle(x)`

Shuffle the sequence `x` in place.

To shuffle an immutable sequence and return a new shuffled list, use `sample(x, k=len(x))` instead.

Note that even for small `len(x)`, the total number of permutations of `x` can quickly grow larger than the period of most random number generators. This implies that most permutations of a long sequence can never be generated. For example, a sequence of length 2080 is the largest that can fit within the period of the Mersenne Twister random number generator.

Deprecated since version 3.9, removed in version 3.11: The optional parameter `random`.

`random.sample(population, k, *, counts=None)`

Return a `k` length list of unique elements chosen from the population sequence. Used for random sampling without replacement.

Returns a new list containing elements from the population while leaving the original population unchanged. The resulting list is in selection order so that all sub-slices will also be valid random samples. This allows raffle winners (the sample) to be partitioned into grand prize and second place winners (the subslices).

Members of the population need not be `hashable` or unique. If the population contains repeats, then each occurrence is a possible selection in the sample.

<https://docs.python.org/3/library/random.html>

PyStart #17 Pseudolosowość

Jak korzystać z tych funkcji?



```
from random import randint, choices, choice

# 81
print(randint(3,100))

fruits = 'mango', 'lemon', 'peach', 'apple'
# ['lemon']
choices(fruits)

# 'lemon'
choice(fruits)
```

PyStart #17 Pseudolosowość

Do czego potrzebuje tej wiedzy?



- Podstawową umiejętnością jest czytanie dokumentacji.
 - ◆ Nie znasz funkcji do wykonania danej czynności.
 - ◆ Nie pamiętasz jakie argumenty przekazać do funkcji.
 - ◆ Wydaje Ci się, że można coś zrobić prościej.

PyStart #17 Pseudolosowość

Do czego potrzebuje tej wiedzy?



- Podstawową umiejętnością jest czytanie dokumentacji.
 - ◆ Nie znasz funkcji do wykonania danej czynności.
 - ◆ Nie pamiętasz jakie argumenty przekazać do funkcji.
 - ◆ Wydaje Ci się, że można coś zrobić prościej.

Bardzo ważne równanie :-)

Dokumentacja + Stackoverflow + Ty = ❤️

PyStart #17 Pseudolosowość

Zadania dla nabrania wprawy

Korzystając z dokumentacji oraz Google:

1. Stwórz tuplę zawierającą papier, kamień, nożyce
2. Zapytaj użytkownika o jego wybór (p, k, n)
3. “Wylosuj” jeden z elementów z tupli i wyświetl informację kto wygrał. Użytkownik czy komputer.

Napisz program, który „wylosuje” liczbę z przedziału od 1 do 100, a następnie korzystając z biblioteki `math` wypisze: kwadrat i pierwiastek kwadratowy zaokrąglony w dół i w górę tej liczby.



TRZECIA PRACA DOMOWA

1. Napisz program, który stworzy listę zawierającą imiona 5 twoich znajomych. Następnie wyświetl imiona twoich znajomych w kolejności alfabetycznej.
2. Stwórz słownik, w którym kluczami będą nazwy owoców, a wartościami ich ceny. Następnie wyświetl cenę jabłek i bananów.
3. Stwórz słownik zawierający informacje o swoich ulubionych filmach (tytuł, reżyser, rok produkcji). Następnie usuń ze słownika filmy, których reżyserem jest Steven Spielberg.
4. Napisz program, który stworzy słownik zawierający informacje o twoich ulubionych zwierzętach (nazwa zwierzęcia, ilość nóg). Następnie znajdź zwierzę, które ma najwięcej nóg i wyświetl jego nazwę i ilość nóg. **Podpowiedź: Nie sortuj słownika!**
5. Podobnie jak w poprzednim zadaniu napisz program, który stworzy słownik zawierający informacje o twoich ulubionych zwierzętach domowych (nazwa zwierzęcia, rasa). Następnie znajdź zwierzę, które ma najdłuższą nazwę i wyświetl jego nazwę i rasę.
6. Napisz program, który stworzy słownik zawierający informacje o twoich ulubionych krajach (nazwa kraju, stolica, język urzędowy). Następnie dodaj do słownika nowy kraj i wyświetl cały słownik po dodaniu nowego elementu.

Przejdź po słowniku za pomocą drugiej pętli for, zamiast robić dictionary['capital']



TRZECIA PRACA DOMOWA

7. "Word jumble": Napisz program, który losuje słowo z listy i miesza jego literki.

Użytkownik ma odgadnąć, jakie to słowo. Program powinien sprawdzić, czy odpowiedź użytkownika jest prawidłowa i wyświetlić odpowiedni komunikat.

Pozwól użytkownikowi zgadnąć 3 razy. W myśl zasady do trzech razy sztuka.

Podpowiedzi:

`from random import shuffle` - mieszanie elementów listy

`list('python')` - zamiana stringa na listę

`''.join(lista)` - zamiana listy na stringa

Słowo do odgadnięcia: tnohpy

Czy wiesz jakie to słowo? python

Tak, chodziło o to słowo! Zgadłeś za pierwszym razem!



TRZECIA PRACA DOMOWA

8. "Dinner menu generator": Napisz program, który losuje po jednym daniu z każdej z trzech kategorii (np. zupa, drugie danie, deser) i wyświetla je na ekranie. Kategorie i dania mogą być zdefiniowane jako słownik, w którym każdemu kluczowi (np. "zupa") odpowiada lista z dostępnymi daniami z tej kategorii.

```
# przykładowa lista dań
dishes = {
    'soup': ['pomidorowa', 'barszcz', 'rosół'],
    'dinner': ['pulpety', 'schabowy', 'gołąbki'],
    'dessert': ['lody', 'kisiel', 'ciasto']
}
```



TRZECIA PRACA DOMOWA

9. Jesteś programistą w małej firmie tworzącej gry komputerowe. Twój szef poprosił Cię o stworzenie prostego programu, który będzie symulować grę w karty. W grze będzie używana talia składająca się z 52 kart, a Twoim zadaniem jest stworzenie listy zawierającej nazwy wszystkich kart oraz funkcji losującej karty z talii.

10. "Jaki to kraj?" - gra quizowa: Napisz program, który losuje jeden z trzech krajów (np. Polska, Niemcy, Francja) i pyta użytkownika o stolicę tego kraju. Jeśli użytkownik poda prawidłową odpowiedź, program powinien wyświetlić komunikat "Dobrze!", a jeśli błędną - "Źle :(".

11. Program powinien zapytać użytkownika trzy razy po czym wyświetlić mu wynik zależny od ilości punktów. 3 - Bezbłędnie Ci poszło! 2- Tylko jeden błąd! Nieźle! 1 - Jeden punkt na otarcie łez, 0 - Następnym razem pójdzie Ci lepiej!

