03. Въведение 2.0

15 октомври 2024

Какво видяхте в предишната лекция?

- Приключихме с колекциите `list` и `tuple`
- Започнахме с колекциите `dict` и `set`
- Приключихме с колекциите `dict` и `set`

Въпроси?

Подай Snickers-а там (1)

```
>>> spam = {}
>>> type(spam) # ?
<class 'dict'>
```

Подай Snickers-а там (2)

```
>>> lectures_location = {'Tuesday': '101', 'Thursday': '200'}
>>> lectures_location['Wednesday'] # ?

Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    lectures_location['Wednesday']
KeyError: 'Wednesday'
```

Подай Snickers-а там (3)

Какъв ще е резултатът от следния код?

```
>>> my_favourite_things = ['spam'] * 100
```

Въпрос за живота, вселената и tuples / lists

Какъв ще е резултатът от следния код?

```
>>> other_things = (42, [1, 2, 3], 'baba')
>>> other_things[1] += [4]

Traceback (most recent call last):
  File "<pyshell#42>", line 1, in <module>
    other_things[1] += [4]
TypeError: 'tuple' object does not support item assignment

>>> other_things
(42, [1, 2, 3, 4], 'baba')
```

Впечатляващо, а?

Домашното от четвъртък

Ние, виждайки, че само 2 от вас не са предали

Домашното от четвъртък

- За първи път решаваме да пуснем толкова лесно мини-домашно
- Причината повечето от вас да нямат обратна връзка е, че кодът е прост и няма за какво толкова да се хванем
- А не върви да ви спамим нотификациите с “Изглежда супер”
- Не се притеснявайте, на следващото ще си получите коментарите

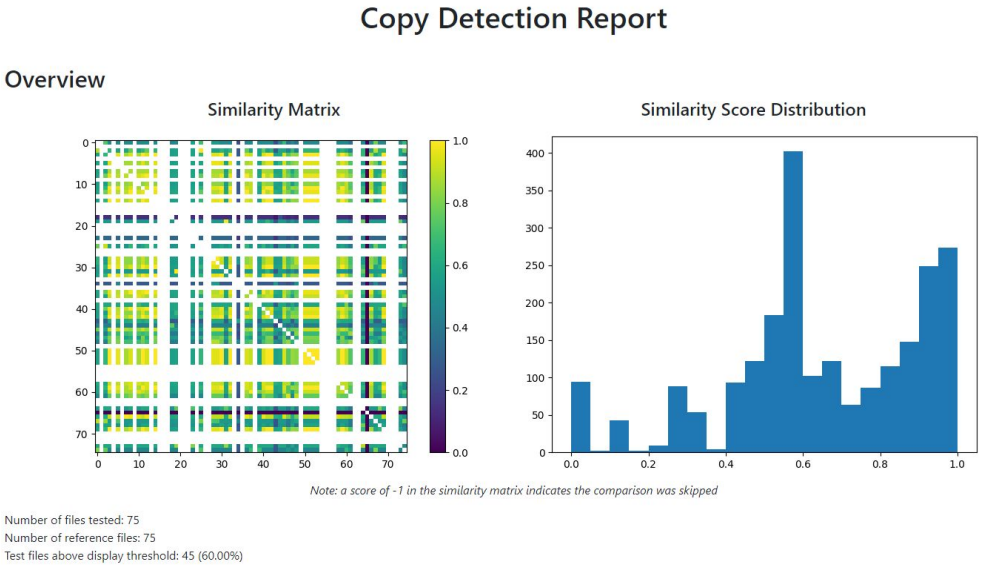
Домашното от четвъртък

- Поради същата причина не сме давали или взимали точки...
- Except for this abomination:

```
(Ω := __import__('cmath')), (لغز := __import__('codecs').decode), (ﻉ := len), (viktors_ingredients
:= str(لغز
(b'//5HBEMESAQ6BDgEOwA0BD4EPAQwBEIEOAQ7ADwEPgRABDoEPgQyBDgEOwBPBDEESgQ7BDoEOAQ7\nAEEEPgQ7BDsARwQ1BE
AENQQ9BCAAPwQ4BD8ENQRABDsAOgQ4BDwEOAQ+BD0EOwA3BDUERQRCBDgE\nPQQ=\n', 'base_64'),
'utf-16').split(';')), (georgis_ingredients := tuple(str(لغز
(b'//5HBEMESAQ6BDgEOwA0BD4EPAQwBEIEOAQ7AD8EMARCBDsEMAQ0BDYEMAQ9BDsAOwROBEIEOAQg\nAEcEQwRIBDoEOAQ7AD
4EOwQ4BD4EOwA3BDAERQQwBEAEOwBHBEMEMQRABDgERgQwBDsARwQ1BEAE\nNQQ9BCAAPwQ4BD8ENQRABDsAMgRABDAERwQwBD0
EQQQ6BDAEIABABDAEOgQ4BE8E\n', 'base_64'), 'utf-16').split(';'))), (shopping_list :=
[*viktors_ingredients, *georgis_ingredients][::int(getattr(Ω.exp(Ω.pi * complex(ﻉ({}), ﻉ([{}]))),
str(لغز(b'cmVhbA==\n', 'base_64'), 'utf-8')))]), (unique_ingredients := set(shopping_list)),
(ingredient_quantities := {k: (lambda 五: 五(lambda へ: へ + (Γ := lambda _: _ * Γ(_ - ﻉ({(),}))) if
_ else ﻉ(({},)))(ﻉ([],)))(ﻉ((),)))(lambda 巛: lambda 巛巛: 巛(巛(巛(巛(巛(巛巛)))))) for k in
unique_ingredients} | {str(لغز(b'c2t5cg==\n', 'base-64'), 'utf-8'): -int(getattr(Ω.exp(Ω.pi *
complex(ﻉ({}), ﻉ([{}]))), str(لغز(b'cmVhbA==\n', 'base_64'), 'utf-8')))}),
(number_of_ingredients_to_buy := len(ingredient_quantities))
```

Домашното от четвъртък

- Поради същата причина не гледаме и следното:



Но ви напомняме, че съществува и не се шегувахме, като ви казахме да не преписвате

Общи коментари

- Няма нужда от `print`-ове
- Напротив, желателно е да ги няма в production код *(освен ако изрично не им е там мястото)*
- Съвсем скоро ще си говорим как работят unit тестовете
- За любопитните:

Но така и така си говорим за print

- Можете (желателно е) да използвате така наречените f-strings:

```
>>> main_character = "Captain Jack Sparrow"
>>> print(f"This is the tale of {main_character}")
"This is the tale of Captain Jack Sparrow"
```

- Имат и интересни опции за форматиране:

```
heights = {"Mount Everest": 8849, "Musala": 2925, "Shaq": 2.16}
for name, height in heights.items():
    print(f"{name:15} ==> {height:10}")
"Mount Everest   ==>       8849"
"Musala          ==>       2925"
"Shaq            ==>       2.16"
```

Повече тук и тук.

Общи коментари (2)

Въпрос - какво ще се случи тук?

```
>>> shopping_list = viktors_ingredients
>>> shopping_list.extend(georgis_ingredients)
>>> shopping_list.reverse()
>>> viktors_ingredients
```

```
['врачанска ракия', 'черен пипер', 'чубрица', 'захар', 'олио', 'люти
чушки', 'патладжан', 'домати', 'чушки', 'зехтин', 'кимион', 'черен
пипер', 'сол', 'ябълки', 'моркови', 'домати', 'чушки']
```

Не забравяйте, mutable обектите се подават “по референция”!

Общи коментари (3)

- Стил
  - Интервали
  - Подравняване
  - Смесени кавички
  - Никое от тези не е драма
  - Но все пак - PEP8
- for цикли - не са зле, но понякога има по-удобни инструменти
- Допълнителна функционалност (сортиране) - четете внимателно условието
- Стига толкова

Какво със сигурност ще видите днес?

- Контролни структури - `if`, `while`, `for`, `switch` (`break`, `continue`)
- Как се дефинират блокове код в Python
- Дефиниране на функции и аргументи на функции
- Супер готина новина

Какво може би ще видите днес?

- Голи снимки
- Функции, които генерират колекции
- Функции, които използват колекции
- Функции, които едновременно използват и генерират колекции
- Бикове и крави

Контролни структури

- `if .. elif .. else`
- `while`
- `for`

if

```
if a == 5:
    print("a is five")
elif a == 3 and not b == 2:
    print("a is three, but b is not two")
else:
    print("a is something else or b is two")
```

- Точно каквото очаквахте.
- Не слагайте скоби около условията.
- `and`, `or` и `not`
- **НЕ** &&, ||, !

if (с булеви променливи)

```
a = True

if a:
    print("a is True")
if not a:
    print("a is not True")
```

Истина и лъжа

В контекста на булевите операции като лъжа се интерпретират следните стойности:

- `False`
- `None`
- числото `0` независимо от типа числа (например `0, 0.0, 0j`)
- празният низ
- празни контейнери (`tuple, list, dict, set`)
- наши типове могат да дефинират как да бъдат оценявани като булеви променливи

Всички останали стойности се интерпретират като истина.

В този ред на мисли...

- Вероятно помните, че можем да “cast”-ваме стойности към различни типове.
- В кавички, защото не е баш кастване и определено не е като в статично типизираните езици.

```
int('5') # 5

b = 10
str(b) # '10'

nums = (1, 2, 3)
nums = list(nums)
nums # [1, 2, 3]

str(['baba', 2]) # "['baba', 2]"
```

Разбира се в някакви граници

```
int('диевиетстотин и пидисе')

Traceback (most recent call last):
  File "<pyshell#950>", line 1, in <module>
    int('диевиетстотин и пидисе')
ValueError: invalid literal for int() with base 10: 'диевиетстотин и
пидисе'
```

Тогава - въпрос

На базата на последните 3 слайда, до какво ще се оцени долното:

```
>>> question = "Питона искаш ли да ти покажа?"
>>> bool(question)

True
```

В почивката.

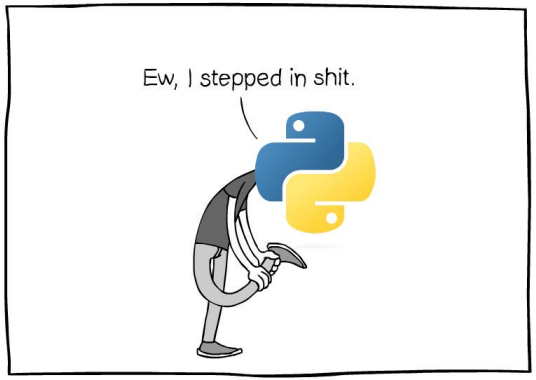
Също така, цитат от преди 3 слайда:
Всички останали стойности се интерпретират като истина.

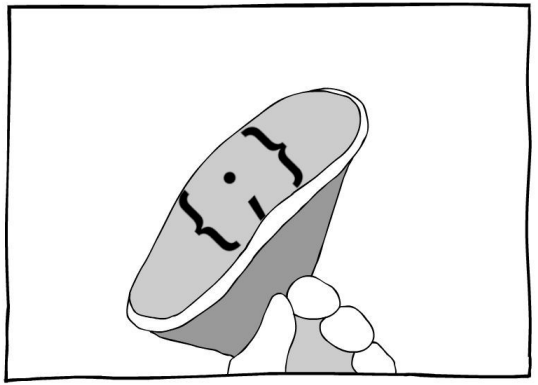
if (тестове за принадлежност)

```
my_list = [1, 2, 3, 4]

if 1 in my_list:
    print('1 is in my list')

if 5 not in my_list:
    print('5 is not in my list')
```

Индентация

- Къде са къдравите скоби?!
- Всеки блок код (тяло на `if`, тяло на функция, и т.н.) се определя с индентацията му спрямо обгръщащия го блок.
- Всеки блок код започва само след двоеточие в края на предишния ред.
- Блокът свършва, когато се върнете към предишната индентация.
- 4 празни места = нов блок.
- **Не 2, не 3, не 8, не табулация**
- Дресирайте редактора си да слага 4 празни места когато натиснете <Tab>





Индентация


C#
Java
C++
New Programmer
Missing ;
Python
IdentationError:


TabError: inconsistent use of tabs and spaces in indentation
Python Programmer

while

```
a = 10
while a > 5:
    a -= 1
    print(f"a is {a}")
```

- Елементарно
- Въпроси?

Какво е колекция?

- В Python всички колекции са итерируеми (iterable)
- Един итерируем обект може да бъде обхождан последователно (поне веднъж)
- Някои могат да бъдат обхождани многократно

for

```
primes = [3, 5, 7, 11]
for e in primes:
    print(e ** 2) # 9 25 49 121

people = {'bob': 25, 'john': 22, 'mitt': 56}
for name, age in people.items():
    print("{} is {} years old".format(name, age))
    # bob is 25 years old
    # john is 22 years old
    # ...
```

- `for` е като `foreach` в другите езици
- Няма инициализация, стъпка и проверка, не е fancy `while`
- Обхожда структури от данни

Един пример от предна лекция

```
nice_things = ['coffee', 'cheese', 'crackers', 'tea']
for thing in nice_things:
    print(f'I tend to like {thing}')

# I tend to like coffee
# I tend to like cheese
# ...
```

for като в C

```
for i in range(0, 20):
    print(i)
    # 0 1 2 3 4 5 6 .. 19

for i in range(0, 20, 3):
    print(i)
    # 0 3 6 9 12 15 18
```

Може и наобратно

```
for i in range(20, 0, -1):
    print(i)
    # 20 19 18 17 16 15 .. 1

for i in range(20, 0, -3):
    print(i)
    # 20 17 14 11 8 5 2
```

break и continue

- Работят, както очаквате във `for` и `while`.
- Афектират само най-вътрешния цикъл.

switch/case

- Няма...
- Добре де, нямаше...
- Вече има (Python >= 3.10).
- Все още не сме решили дали е добра идея.
- Засега можете да си поиграете с него.
- И в Python е `match`/`case`.

match/case

```
http_status = 400

match http_status:
    case 400:
        print("Bad request")
    case 401 | 403: # 401 OR 403
        print("Authentication error")
    case 404:
        print("Not found")

# Bad request
```

match/case

```
http_status = 9001

match http_status:
    case 400:
        print("Bad request")
    ...
    case _: # Default
        print("Other error")

# Other error
```

Има и още хиляда синтактични конструкции свързани с `match`/`case`, за момента толкоз.

Функции

```
def say_hello(name, side):
    return "Hello… It's me…"
```

- Функцията приема аргументи
- Функцията може да върне нещо с `return`, а ако няма `return`, връща `None`
- Не се описват типовете на аргументите, нито типа на резултата

Аргументи на функции - позиционни

- Параметрите в предният пример са позиционни
- Ако функцията има 3 позиционни параметъра - трябва да я извикате с 3 аргумента
- Иначе - грешка
- Параметри vs аргументи?!

```
def multiply(a, b):
    return a * b

multiply(5, 10) # 50

multiply(5)
# TypeError: multiply() missing 1 required positional argument: 'b'
```

Аргументи на функции - именувани

```
def multiply(a, b=2):
    return a * b

multiply(5) # 10
multiply(5, 10) # 50

def is_pythagorean(a=2, b=3, c=4):
    return a * a + b * b == c * c

is_pythagorean(b=5, a=3) # c = 4
is_pythagorean(1, c=3) # a = 1, b = 3
```

- Именувани, по подразбиране, опционални
- Могат да бъдат изрично упоменати, ако не - взимат стойността по подразбиране
- Стига позиционните да са подадени в правилен ред, именуваните могат да бъдат в какъвто и да е ред

Променлив брой аргументи

```
def varfunc(some_arg, *args, **kwargs):
    #...

varfunc('hello', 1, 2, 3, name='Bob', age=12)
    # some_arg == 'hello'
    # args = (1, 2, 3)
    # kwargs = {'name': 'Bob', 'age': 12}
```

- Функциите могат да приемат произволен брой аргументи
- Позиционните аргументи (тези без име) отиват в `args`, което е `tuple` от аргументи
- Именуваните аргументи отиват в `kwargs`, което е `dict` от имена на аргументи и съответните им стойности
- Имената `args` и `kwargs` не са специални, **но са наложена конвенция**
- **Редът е важен!**

Аргументи на функции - позиционни

- Има и други шано неща като positional only, keyword only, optional positional, required keyword и прочие параметри
- Няма да навлизаме в тях
- Повярвайте ни, ще ви заболи главата, а е малко вероятно да ви трябват
- За когато се наложи - знайте, че ги има

```
def baba(a, b, /, c, d, *, e=2.72, f, **kwargs):
    do_stuff()
```

Кой какво сиренье има (всичко дотук - заедно)

```
data = [('John', 'Tilsit'), ('Eric', 'Cheshire'), ('Michael', 'Camembert'),
        ('Terry', 'Gouda'), ('Terry', 'Port Salut'), ('Michael', 'Edam'),
        ('Eric', 'Ilchester'), ('John', 'Fynbo')]

def cheeses_by_owner(cheeses_data):
    by_owner = {}
    for owner, cheese in cheeses_data: # <- tuple unpacking
        if owner in by_owner:
            by_owner[owner].append(cheese)
        else:
            by_owner[owner] = [cheese]
    return by_owner
```

Къде са голите снимки?

Имаме още

Тери Джоунс (да, от Монти Пайтън) в ролята на “Голият органист”

Range

`range` връща итерируемо за интервал от числа

```
numbers = range(3)
for number in numbers:
    print('We can count to ' + str(number))
```

Range

```
numbers = range(10, 13)
for number in numbers:
    print('We can count to ' + str(number))
```

Range

range може и в обратен ред

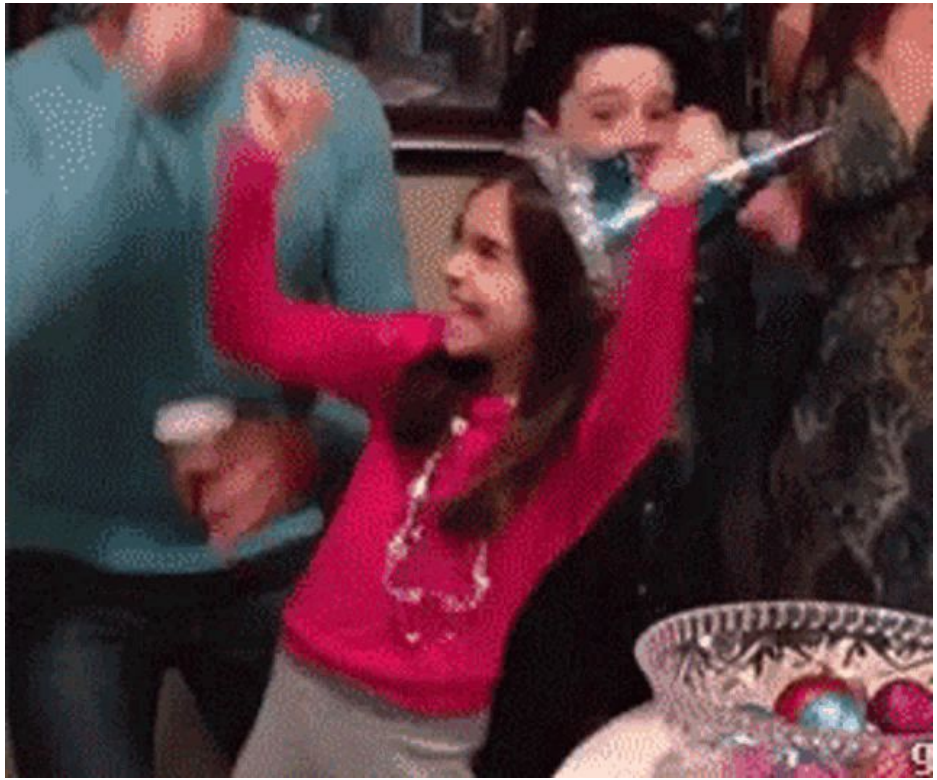
```
numbers = range(13, 0, -1)
for number in numbers:
    print('We can count to ' + str(number))
```

Map/Filter/Reduce/All/Any/Lambda

- `map(function, iterable)` създава колекция от резултатите от прилагането на function върху всеки елемент от iterable
- `filter(function, iterable)` създава колекция само с елементите, за които function върне `True`
- `reduce(function, iterable)` вика `function` с елементите на колекцията, докато сведе всичко до една стойност (във `functools` вж. тук)
- `all(iterable)` всички елементи се оценяват на истина
- `any(iterable)` поне един от елементите се оценява на истина
- `lambda` функции - анонимни функции, които често вървят ръка за ръка с горните

За любознателните: `map()` и `filter()` са мързеливи

Ново домашно!



Ново домашно

- До 18:00 на 22.10 (следващият вторник)
- Вече започва да става интересно
- Знаете, ако имате въпроси - в коментарите

Колекции за хора без компания в петък вечер

- `deque` - двупосочни опашки
- `OrderedDict` - речник, който помни реда
- `defaultdict` - речник със стойност по подразбиране
- `Counter` - речник, който брои повтарящи се стойности
- `namedtuple` - кортеж с именувани полета

Deque

```
from collections import deque

adjectives = deque()

def add_adjective(items):
    adjectives.append(items)
def get_adjective():
    return adjectives.popleft()

add_adjective('John')
add_adjective('Terry')
add_adjective('Graham')
add_adjective('Eric')
print(', '.join(adjectives) + ', Michael, Terry') # John, Terry, Graham,
Eric, Michael, Terry
```

Defaultdict

```
from collections import defaultdict

data = [('John', 'Tilsit'), ('Eric', 'Cheshire'), ('Michael', 'Camembert'),
        ('Terry', 'Gouda'), ('Terry', 'Port Salut'), ('Michael', 'Edam'),
        ('Eric', 'Ilchester'), ('John', 'Fynbo')]

def cheeses_by_owner(cheeses_data):
    by_owner = defaultdict(list)
    for owner, cheese in cheeses_data:
        by_owner[owner].append(cheese)
    return by_owner
```

Упражнение - Бикове и Крави (1)

```
import random # Засега - магия. След няколко лекции - ежедневие.

LENGTH = 4 # Константа за дължина на числата

def get_secret():
    """Generate random 4-digit number with unique digits (no zero)."""
    all_digits = list(map(str, range(1, 10)))
    random.shuffle(all_digits)
    return ''.join(all_digits[:LENGTH])
```

Упражнение - Бикове и Крави (2)

```
def compare(num1, num2):
    """Return bulls/cows count between two numbers. See the rules."""
    bulls, cows = 0, 0
    for dig1, dig2 in zip(num1, num2):
        if dig1 == dig2:
            bulls += 1
        elif dig1 in num2:
            cows += 1
    return bulls, cows
```

Упражнение - Бикове и Крави (3)

```
SECRET = get_secret()
print('Намислих си четирицифрено число без повторение на цифрите и без
нули. Опитай да познаеш...')
while True:
    guess = input('Въведи предположение: ')
    bulls, cows = compare(SECRET, guess)
    if bulls == LENGTH:
        print(f'Позна, машино!')
        break
    else:
        print(f'Имаш {bulls} бика и {cows} крави.')
```

Упражнение - Бикове и Крави (4)

```
while True:
    guess = input('Въведи предположение: ')
    # Вмъкваме малко валидация в while цикъла
    if not guess.isdigit():
        print('Това май не е число.')
        continue
    if len(guess) != LENGTH:
        print('Числото трябва да е четирицифрено.')
        continue
    if len(set(guess)) != LENGTH:
        print('Числото трябва да съдържа само уникални цифри.')
        continue
    if '0' in guess:
        print('Числото не може да съдържа нула.')
        continue
    bulls, cows = compare(SECRET, guess)
    …
```

Въпроси?