SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Metodika písania kódu v jazyku Python

Tím 22 – Predicar
Vedúci: Ing. Miroslav Rác

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

Sparse is better than dense.

Readability counts.

Special cases aren't special enough to break the rules.

Although practicality beats purity.

Errors should never pass silently.

Unless explicitly silenced.

In the face of ambiguity, refuse the temptation to guess.

There should be one—and preferably only one—obvious way to do it.

Although that way may not be obvious at first unless you're Dutch.

Now is better than never.

Although never is often better than *right* now.

If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

Namespaces are one honking great idea—let's do more of those!

The Zen of Python (PEP 20)

1. Dedikácia

Metodika definuje princípy a pravidlá písania kódu v jazyku Python. Opisuje štýl písania, štandardy ktoré používame a dokumentáciu kódu.

2. Zaužívané štandardy

Pri písaní kódu sa riadime zaužívanými štandardmi PEP 8 (Style Guide for Python Code) pre písanie kódu, PEP 257 (Docstring Conventions) pre dokumentáciu kódu a PEP 484 (Type Hints) pre typovanie kódu.

Kód píšeme v anglickom jazyku, premenným, metódam a triedam dávame výstižný názov.

Názvy premenných a metód píšeme celé malými písmenami, jednotlivé slová oddelené podtržníkom (_).

Názvy tried píšeme štýlom UpperCaseCamelCase.

Funkcie a definície tried sa oddeľujú 2 prázdnymi riadkami. Definície metód v triedach sa oddeľujú 1 prázdnym riadkom. Na konci súboru musí byť 1 prázdny riadok.

```
class ExampleClass():

    def example_method():

        example_var1 = 1
        example_var2 = 2

        return example_var1 + example_var2
```

3. Používanie nápovedí typov (type hints)

Pri písaní kódu používame nápovede typov premenných dovážaných do funkcií a typov vracaných funkciami.

```
def example_method(number: int, letter: String) -> String:

    return letter + str(number)
```

4. Dokumentovanie kódu

Komentáre, podobne ako kód, píšeme v **anglickom** jazyku. Riadime sa pri tom štandardom PEP8. Blokové komentáre platia pre kód pod nimi a sú odsadené na rovnakej úrovni ako kód ktorý komentujú. Každý riadok blokového komentáru začína znakom **#** a jednou medzerou.

Pri písaní docstringu pre metódy a triedy sa riadime štandardom PEP257 pre docstring konvencie a používame Google štýl písania docstringov. Každý docstring obsahuje krátky a výstižný opis metódy, alebo triedy. Každý docstring opisuje privezené argumenty ako Args, typ návratu ako Returns a vyvolané výnimky ako Raises.

```
def connect_to_next_port(self, minimum):
    """Connects to the next available port.

    Args:
      Minimum(int): A port value greater or equal to 1024.

      Returns:
        int: The new minimum port.

      Raises:
        ConnectionError: If no available port is found.
    """
    if minimum < 1024:
        # Note that this raising of ValueError is not mentioned in the doc
        # string's "Raises:" section because it is not appropriate to
        # guarantee this specific behavioral reaction to API misuse.
        raise ValueError('Minimum port must be at least 1024, not %d.' % (minimum,))
    port = self._find_next_open_port(minimum)
    if not port:
        raise ConnectionError('Could not connect to service on %d or higher.' % (minimum,)
    assert port >= minimum, 'Unexpected port %d when minimum was %d.' % (port, minimum)
    return port
```

Viac o Google štýle písania docstringov:

https://sphinxcontrib-napoleon.readthedocs.io/en/latest/example_google.html
http://google.github.io/styleguide/pyguide.html