Software development

Course: Object Oriented Programming (OOP)

CTU, FS, U12110

Matouš Cejnek

Contents

- Coding conventions
- Version control
- Software testing
- Documentation
- Refactoring

Coding conventions

Coding conventions

Coding conventions are a set of guidelines for a **specific programming language** that recommend programming style and best practices.

```
sample_number = 1 + 1
SampleNumber = (1+1)
sn=1+1;
```

Coding conventions

It is extremely crucial to follow coding conventions of the given language. Failing to do so may leads to:

- Bugs and difficulties when fixing them
- Impossibility to cooperate with other people and/or robots
- Difficulties to understand your own code in a greater time scope (greater than days, weeks)

Coding conventions - Fizzbuzz example 1

```
for number in range(100):
    if number % 3 == 0 and number % 5 == 0:
        print("fizzbuzz")
        continue
    elif number % 3 == 0:
        print("fizz")
        continue
    elif number % 5 == 0:
        print("buzz")
        continue
    print(number)
```

Coding conventions - Fizzbuzz example 2

```
print("\n".join(
  [(((str(_)if _%5else"buzz")if _%3else"fizz")
   if _%15else"fizzbuzz")for _ in range(100)]))
```

It also works!

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Coding conventions

Few aspects:

- file organization
- indentation
- comments
- statements
- programming principles
- naming conventions
- white space
- declarations
- programming practices

Coding conventions

File organization and naming conventions

Example 1:

```
models/
    class1.py
    class2.py
tests/
   test1.py
main.py
test.py
```

Example 2:

```
huge_file.py
testX.py
test31.py
testB.py
other_huge_file.py
```

Coding conventions

Indentation and white space

Example 1

```
people = {
    "Alice": 20,
    "Bob": 19,
}
```

Example 2

```
l1l11={"Alice":20,"Bob":19}
```

Coding conventions

Naming conventions

Example 1

```
OOOOOOOO = OOOOOOOOO + OOOOOOOO
```

Example 2

```
a = b + c
```

Example 3

```
distance = offset + deviation
```

Version control

Version control

- Version control is a class of systems responsible for managing changes to computer programs or similar projects.
- Version control is one of the greatest advancements in modern programming.
- Other names are: revision control, source control, or source code management

Version systems

Main types are:

- Distributed version control
- Centralized version control

Popular systems:

- **Git** - distributed
- **Subversion** (SVN) - centralized

Git

- Git is free and open source software for distributed version control.
- Git was originally authored by **Linus Torvalds** in 2005 for development of the Linux kernel.

Git

- You can use git locally to trace your project.
- You can add remote addresses and exchange the changes in your project (push / pull)
- You can use git server as a remote address.
- There are third party services providing git hosting for your projects.

Git hosting

There are many services offering remote Git hosting (free and paid plans):

- GitHub - belongs to Microsoft since 2018
- GitLab - similar to GitHub, belongs to GitLab Inc.
- BitBucket - belongs to Atlassian

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Git basic functionality

- Git traces the changes in your files.
- Git allows you to “save the state” of your files (**commit**).
- Git allows you to upload (**push**) or download (**pull**) commits to/from remote storage.
- Git allows you to create parallel version (**fork** or **branch**) of your project.

Git terminology

Commit - mark (commit) changes in files

Push / pull - upload and download commits

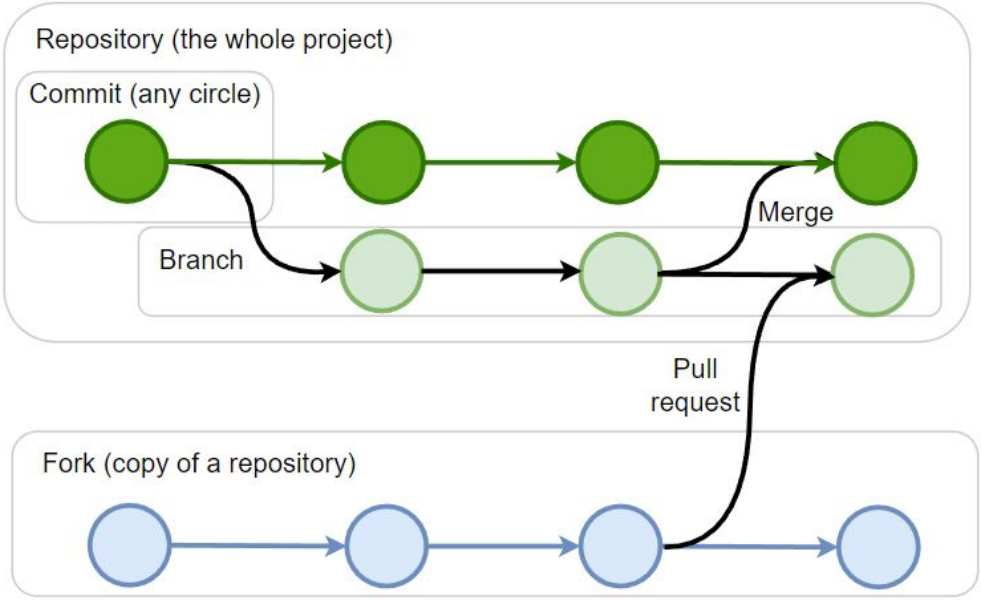
Repository - full stored project

Branch - branch in the tree of a repository

Fork - complete copy a repository

Release - commit marked as a version of the project

Git terminology



Software testing

Why are we testing?

- Program debugging
- Program profiling
- Program compatibility
- Security risk identification

Debugging

- Removing bugs allows us to see another bugs (iterative process)
- Bugs can cause a program crash, or incorrect results.
- Difference between bug and feature is goal dependent.

Program profiling

- It is a form of dynamic program analysis that measures various aspects of a program.
- Common aspects to analyze are:
  - memory utilization,
  - time complexity,
  - usage of particular instructions,
  - frequency and duration of function calls

Program compatibility

- Can program run on all desired hardware?
- Can program run on all target operation systems?
- What all software/hardware dependencies of the program?

Security risk identification

- Vulnerability scanning
- Penetration testing
- Risk Assessment
- …

Testing classification

One of the possible classifications:

- **Static testing** - code assessment
- **Dynamic testing** - testing of a running program
- **Passive testing** - observation of the program outputs

Static testing

- During code writing or when the code is finished
- It can be automated (special tools, IDE, …)
- It can be done by another person(s)

Dynamic testing

- Tests should be cheap on resources
- Testing should be done before and after development
- Boundary conditions and unstandard cases should be tested
- Tests should evaluate outputs and also errors

Passive testing

Logs of the program should be checked for:

- weird behaviour,
- errors, and
- suspicious patterns.

Testing classification (levels)

Another possible classification:

- **Unit testing** - smallest units (functions, etc.)
- **Integration testing** - parts of the program (modules, etc.)
- **System testing** - whole program
- **Acceptance testing** - program success

Software documentation

Software documentation

- Requirements – capabilities of the program
- Architecture/Design – informations about design of the program
- **Technical** – description of code, algorithms, interfaces, and APIs.
- End user – manuals for the end-user
- Marketing – How to market the product and analysis of the market demand.

Technical documentation

- Nowadays it is common to incorporate documentation in code.
- There are different recommendations and standards for code documentation (it is language specific issue)
- Documentation from code is often compiled into documents via automated tools.

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Technical documentation example

```
def add_one(value: int) -> int:
    """Add one to the provided value.
    :param value: Integer value to be increased by one
    :return: Increased integer value
    """
    return value + 1
```

Refactoring

Refactoring

Code refactoring is the process of restructuring existing computer code (changing the factoring) without changing its external behavior.

Refactoring

- For easy refactoring, you need to have a good test battery.
- Refactoring is important investition - it save your future time and problems.

Bonus content

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Stack exchange - code review

You can ask for help with coding conventions for specific code snippet at code review:

https://codereview.stackexchange.com/

For example:

https://codereview.stackexchange.com/questions/158181/function-that-returns-index-of-forex-candle-according-opening-hours-weekly-sess

Python coding conventions

PEP8

https://peps.python.org/pep-0008/

Sphinx

Sphinx is a documentation generator written and used by the Python community. It is written in Python.

https://www.sphinx-doc.org/en/master/

Pylint

Pylint is code analyser for Python.

https://pylint.pycqa.org/en/latest/