Turun yliopisto
University of Turku

Programming style

Ilia Kosenkov

Tuorla Observatory

2017-11-22

Take a look at the code

```
def f(a,b):
    v = 0; w = 0; f23 = 0; pp = 0
    for var1 in a: v = v + var1
    for w1 in b: w = w1/len(b) + w
    v = 1.0/len(a) * v;
    for var2 in a: f23 = pow(var2 - v, 2)  + f23
    f23 = pow(f23/(len(a) - 1), 0.5)
    for w2 in b: pp = pp + pow(w - w2, 2)/(len(b) - 1)
    pp = pow(pp, 0.5)
    print("%4.2f +- %4.2f\t%4.2f +- %4.2f" % (v, f23, w, pp))
    return
```

Can you guess what happens if I call f([1, 2, 3, 4], [60, 35, 2, 0, 35])?

What about now?

```
def mean_and_sd(sample):
    """ Calcualtes mean and standard deviation
    Args:
        sample as list
    Returns:
        (mean, standard deviation) as tuple
    """
    # Initialization of variables
    mean = 0
    sd = 0
    # Calculates mean
    for item in sample:
        mean += item
    mean /= len(sample)
    # Calculates standard deviation
    for item in sample:
        sd += pow(item - mean, 2)
    sd = pow(sd / (len(sample) - 1), 0.5)
    # Returns tuple
    return (mean, sd)
```

```
def compare_samples(sample_1, sample_2):
    """Compares two samples and prints mean values
    as well as standard deviations.
    Args:
        sample_1 : first sample, expected a list
        sample_2 : second sample, expected a list
    """
    # Statistics of first sample
    result_1 = mean_and_sd(sample_1)
    # Statistics of second sample
    result_2 = mean_and_sd(sample_2)
    # Prints results
    print("%4.2f +- %4.2f\t%4.2f +- %4.2f"
        % (result_1[0], result_1[1],
           result_2[0], result_2[1]))
    return

# Call to method
compare_samples([1, 2, 3, 4], [60, 35, 2, 0, 35])
```

Turun yliopisto
University of Turku

Programming style

Programming style

- Naming convention

Programming style

- Naming convention
- Indentation

Turun yliopisto
University of Turku

Programming style

- Naming convention
- Indentation
- Vertical alignment

Turun yliopisto
University of Turku

Programming style

- Naming convention
- Indentation
- Vertical alignment
- Comments

Turun yliopisto
University of Turku

Programming style

- Naming convention
- Indentation
- Vertical alignment
- Comments
- Documentation

Programming style

- Naming convention
- Indentation
- Vertical alignment
- Comments
- Documentation
- Consistency

Programming style

- Naming convention
- Indentation
- Vertical alignment
- Comments
- Documentation
- Consistency
- And refactoring tools

Turun yliopisto
University of Turku

Naming convention

Naming convention

- Hungarian notation

```
bFlag, strName, lCount, szArr
```

Naming convention

- Hungarian notation

```
bFlag, strName, lCount, szArr
```

- CamelCase

```
flag, objectName, count, arraySize
GetValue, ReadLine
```

Naming convention

- Hungarian notation

```
bFlag, strName, lCount, szArr
```

- CamelCase

```
flag, objectName, count, arraySize
GetValue, ReadLine
```

- Underscores

```
_private_variable,
method_name,
even_longer_method_name
```

Naming convention

- Hungarian notation

```
bFlag, strName, lCount, szArr
```

- CamelCase

```
flag, objectName, count, arraySize
GetValue, ReadLine
```

- Underscores

```
_private_variable,
method_name,
even_longer_method_name
```

- Capitalization and non-case-sensitive languages

```
CAPITALIZED_CONSTANT_VALUE
```

Turun yliopisto
University of Turku

Indentation & Vertical alignment

Indentation & Vertical alignment

- BAD:

```
if(x==y) {something();} else{ somethingElse();}
```

Indentation & Vertical alignment

- BAD:

```
if(x==y) {something();} else{ somethingElse();}
```

- GOOD: use either 4 spaces or 1 tab per each indentation level

```
if(x == y)
{
    something();
    for (int i = 0; i < N; i++)
        someIterativeOperation(i);
}
else
{
    while(x != y)
        somethingElse(&x, arg1, arg2, arg3, arg4,
                &y, arg6, arg7, arg8, arg9);
}
```

Turun yliopisto
University of Turku

Comments

Comments

- Add comments to key and important parts of the code

Comments

- Add comments to key and important parts of the code
- Use one-line comments like this

```
// One line comment describing your code
```

Comments

- Add comments to key and important parts of the code
- Use one-line comments like this

```
// One line comment describing your code
```

```
/* Not really a good way to comment in C */
```

Comments

- Add comments to key and important parts of the code
- Use one-line comments like this

```
// One line comment describing your code
```

```
/* Not really a good way to comment in C */
```

- Add descriptions to classes & methods

```
def func(arg):
    """ Short desciption of what function does
        Args:
            arg: Very important parameter
        Returns: Nothing
        Raises: Scary run-time exceptions
    """
    return
```

Turun yliopisto
University of Turku

What else?

What else?

- Think of reusability! If a certain operation is performed several times in different places, consider moving it to a separate method. **Avoid copy-pasting code!**

Turun yliopisto
University of Turku

What else?

- Think of reusability! If a certain operation is performed several times in different places, consider moving it to a separate method. **Avoid copy-pasting code!**
- Avoid obsolete and unclear commands & instructions. In particular, never use `go to`. Follow new syntax and style recomendations that are released once per N years.

Turun yliopisto
University of Turku

What else?

- Think of reusability! If a certain operation is performed several times in different places, consider moving it to a separate method. **Avoid copy-pasting code!**
- Avoid obsolete and unclear commands & instructions. In particular, never use `go to`. Follow new syntax and style recomendations that are released once per N years.
- Use brackets/other delimiters when necessary to improve code structure

```
a1+b1+c1+d1*f2 = (a1 + b1) + c1 + (d1 * f2)
```

What else?

- Think of reusability! If a certain operation is performed several times in different places, consider moving it to a separate method. **Avoid copy-pasting code!**
- Avoid obsolete and unclear commands & instructions. In particular, never use `go to`. Follow new syntax and style recomendations that are released once per N years.
- Use brackets/other delimiters when necessary to improve code structure

```
a1+b1+c1+d1*f2 = (a1 + b1) + c1 + (d1 * f2)
```

- Make your functions short. A rule of thumb - source code of one function should fit in one screen. Try to limit line length to approx. 80 symbols. If your code snippet is significantly larger, break it into smaller pieces.

Turun yliopisto
University of Turku

What else?

What else?

- Add some checks and tests. E.g. ensure that parameter that you get is valid.

```
RESULT calculateIntegral(double lowerLim, double upperLim, double * result)
{
   *result = 0.0;
   if(lowerLim > upperLim)
      return ERR_OUT_OF_RANGE;
   ...
   return SUCCESS;
}
```

What else?

- Add some checks and tests. E.g. ensure that parameter that you get is valid.

```
RESULT calculateIntegral(double lowerLim, double upperLim, double * result)
{
   *result = 0.0;
   if(lowerLim > upperLim)
      return ERR_OUT_OF_RANGE;
   ...
   return SUCCESS;
}
```

- Be extra careful with non-typed languages like Python. Consider also checking if you get parameter of proper type (e.g. you expect number, but get string).

What else?

- Add some checks and tests. E.g. ensure that parameter that you get is valid.

```
RESULT calculateIntegral(double lowerLim, double upperLim, double * result)
{
   *result = 0.0;
   if(lowerLim > upperLim)
      return ERR_OUT_OF_RANGE;
   ...
   return SUCCESS;
}
```

- Be extra careful with non-typed languages like Python. Consider also checking if you get parameter of proper type (e.g. you expect number, but get string).
- You can apply these rules to your LaTeX sources. Make your environemnts more user-friendly, consider defining shortcuts and new commands (but do not overdo).

Turun yliopisto
University of Turku

List of useful links

- PEP 8 - Style guide for Python code
- Google Python style guide
- Google R style guide
- Google C++ style guide
- NASA C guide, very strict
- There is even one for FORTRAN

Turun yliopisto
University of Turku

Tuorla Observatory
University of Turku

Thank you!