

Why make it long and understandable when you can make it short and terrible ?

Samuel Gaist
Swiss RSE Day
31.08.2026

Agenda

- Introduction
- Short is so great
- Verbose is so great
- Just right is so great
- Why does it matter ?
- Q/A

Story time

- HEVs Electrical Engineer (so not in the chart)
- Learned on the field but with great people
- Discovered research code
- Release manager

What do I know about code ?



Story time

Clever code is usually short and cryptic, written by and for the individual who came up with it, but is hard for anyone else to understand

and nearly impossible to maintain.

– Guido van Rossum

Story time

Maintainable code is more
important than clever code

– *Guido van Rossum*

Story time

Code does not need to be clever to
give a headache

– Me

What are we going to see ?

- Everyday life examples
- Might look silly but are the start of the chaos

Short is so great !

```
if x < y < z: print(x); print(y); print(z)
```

What will happen ?

Short is so great !

Let's rewrite it

```
if x < y < z:  
    print(x)  
    print(y)  
    print(z)
```

Short is so great !

$a = 1$ if $i < 100$ else 2 if $i > 100$ else 0

What's the value of a and under which conditions ?

Short is so great !

Let's rewrite it

```
if i < 100:  
    a = 1  
elif i > 100:  
    a = 2  
else:  
    a = 0
```

Short is so great !

Let's simplify it

```
a = 0
if i < 100:
    a = 1
elif i > 100:
    a = 2
```

Short is so great !

```
int a = b > 10 ? c < 20 ? 50 : 80 : d == 2 ? 4 : 8;
```

What's the value of a and under which conditions ?

Short is so great !

Let's rewrite it

Maybe a bit too long ?

```
int compute_a(int b, int c, int d)
{
    int a = 0;
    if (b > 10) {
        if (c < 20) {
            a = 50;
        } else {
            a = 80;
        }
    } else {
        if (d == 2) {
            a = 4;
        } else {
            a = 8;
        }
    }
    return a;
}
```

Short is so great !

A good compromise

```
int compute_a_v2(int b, int c, int d)
{
    if (b > 10) {
        return c < 20 ? 50 : 80;
    }
    return d == 2 ? 4 : 8;
}
```

Short is so great !

Does this only come from developer code ?

No !

Short is so great !

- E2BIG : Argument list is too long
- EMLINK : Too many links
- ENXIO : No such device or address
- ESRCH : No such process
- EIDRM : Identifier removed
- strcpy : copy string to dst from src
- strcmp : compare two strings byte by byte

Short is so great !

- Why so short ?
 - Space constraints of old times
 - 80 columns was a hard limit
 - Identifiers with external linkage could as low as 6 characters (historical)

Verbose is so great !

```
void updateFromStatus(bool isRight) {  
    if (isRight == true) {  
        _myColor = "green";  
    } else if (isRight == false) {  
        _myColor = "red";  
    } else {  
        _myColor = "blue";  
    }  
    update();  
}
```

Verbose is so great !

First, let's fix it...

```
void updateFromStatus(bool isRight) {  
    if (isRight == true) {  
        _myColor = "green";  
    } else if (isRight == false) {  
        _myColor = "red";  
    }  
    update();  
}
```

Verbose is so great !

Then, let's make it clean...

```
void updateFromStatus(bool isRight) {  
    if (isRight) {  
        _myColor = "green";  
    } else {  
        _myColor = "red";  
    }  
    update();  
}
```

Verbose is so great !

Now, let's make it short...

```
void updateFromStatus(bool isRight) {  
    _myColor = isRight ? "green" : "red";  
    update();  
}
```

Verbose is so great !

```
def pre_process_feature(feature_name: str) -> None:
    if feature_name != "pink" and feature_name != "magenta" and feature_name != "blue" and feature_name != "purple" and feature_name != "yellow":
        raise ValueError("Only feature_name 'pink' and 'magenta' are supported")
    # The rest of the code
```

Verbose is so great !

First, let's make it readable...

```
def pre_process_feature(feature_name: str) -> None:
    if (
        feature_name != "pink"
        and feature_name != "magenta"
        and feature_name != "blue"
        and feature_name != "purple"
        and feature_name != "yellow"
    ):
        raise ValueError(
            "Only feature_name 'pink' and 'magenta' are supported"
        )
    # Rest of the code
```

Verbose is so great !

Let's make it simpler...

```
def pre_process_feature(feature_name: str) -> None:
    if feature_name not in ["pink", "magenta"]:
        raise ValueError(
            "Only feature_name 'pink' and 'magenta' are supported"
        )
    # Rest of the code
```

Verbose is so great !

Let's make it smarter

```
def pre_process_feature(feature_name: str) -> None:
    valid_values = ["pink", "magenta"]
    if feature_name not in valid_values:
        raise ValueError(
            f"Only feature_name {' and '}.join(f" '{v}' " for v in valid_values)} are supported"
        )
    # Rest of the code
```

Verbose is so great !

Did we get it wrong ?

```
def pre_process_feature(feature_name: str) -> None:
    valid_values = ["pink", "magenta", "blue", "purple", "yellow"]
    if feature_name not in valid_values:
        raise ValueError(
            f"Only feature_name {' and '}.join(f" '{v}' " for v in valid_values)} are supported"
        )
    # Rest of the code
```

Verbose is so great !

Does this only come from developer code ?

No !

Verbose is so great !

- Windows :
ConvertStringSecurityDescriptorToSecurityDescriptor 51
- Apple :
CMMetadataFormatDescriptionCreateWithMetadataForm
atDescriptionAndMetadataSpecifications 87
- Java :
supportsDataDefinitionAndDataManipulationTransactions 53
- Ruby :
HasManyThroughCantAssociateThroughHasOneOrManyRe
flection 65

Verbose is so great !

- Why so long ?
 - Self-documenting name
- Use cases ?
 - Languages with no namespacing
 - Behaviour Driven Testing
 - Allergy to comments / documentation

Just right is so great !

```
const extractDataFromResponse = (response) => {
  const [Component, props] = response;

  const resultsEntries = Object.entries({ Component, props });
  const assignIfValueTruthy = (o, [k, v]) => (v
    ? { ...o, [k]: v }
    : o
  );

  return resultsEntries.reduce(assignIfValueTruthy, {});
}
```

Just right is so great !

Let's make it simpler...

```
const extractDataFromResponse = (response) => {  
  const [Component, props] = response;  
  
  const output = {};  
  
  if (Component) {  
    output.Component = Component;  
  }  
  
  if (props) {  
    output.props = props;  
  }  
  
  return output;  
}
```

Long ? Short ?

Kc

**“It’s all a question of equilibrium
between madness and chaos”**

– The guy who made the presentation

Why does it matter ?

- Understandability
- Learnability
- Teachability
- Maintainability

Why does it matter ?

But LLMS ?!?

They are a tool not a replacement

And

What are they trained on ?

Recommendation

Always code as if the guy who ends up
maintaining your code will be a
violent psychopath who knows where
you live.

Code for readability

– *John F. Woods 1991 (maybe)*

Q/A



Now what ?

- Learn the craft
- Write understandable code
- Don't be clever be smart when coding
- Skynet will remember

Links

- Guido : <https://blog.dropbox.com/topics/company/thank-you--guido>
- [C++](#) :
 - <https://stackoverflow.com/a/160295/5843716>
- [Python](#) :
 - <https://stackoverflow.com/a/14029300/5843716>
 - https://docs.python.org/3/reference/compound_stmts.html
- [JavaScript](#) :
 - <https://www.alexkras.com/reverse-engineering-one-line-of-javascript/>
- [Equilibrium constant](#) :
 - <https://www.chemguide.co.uk/physical/equilibria/kc.html>
- Clever code considered harmful:
 - <https://www.joshwcomeau.com/career/clever-code-considered-harmful/>