

Why naming is important?

Critical for Readability = Maintainability

The naming is important because it is very critical for readability and if you can't read the code, you can't properly maintain it.

Imagine a book that you don't understand, and someone comes to you and asks you to fix the typos in it.

Can you really do it, without understanding it?

There are only two hard things in Computer Science:
cache invalidation and naming things.

- Phil Karlton

Non-Idiomatic

```
func Read(buffer *Buffer, inBuffer []byte) (size
int, err error) {
    if buffer.empty() {
        buffer.Reset()
    }
    size = copy(
        inBuffer,
        buffer.Buffer -
r[buffer.Offset : buffer.Offset + size])
    buffer.Offset += size
    return size, nil
}
```

This code is unnecessarily verbose. Everything has been declared in English words, which generally should be avoided. From the readability and maintainability perspective, this code is not good.

Idiomatic

```
func Read(b *Buffer, p []byte) (n int, err error) {
    if b.empty() {
        b.Reset()
    }
    n = copy(p, b.buf[b.Offset:])
    b.Offset += n
    return n, nil
}
```

This code is very concise and idiomatic and it's easy to understand and maintain.

References

[Abbreviation in Go](#)

[golang bytes standard library](#)

[Inanc Gumus - Learn Go Programming](#)

Use the first few letters of the words

```
var fv string // flag value
```

Use fewer letters in smaller scopes

```
var bytesRead int // number of bytes read ✗
var n int // number of bytes read ✓
```

Use the complete words in larger scopes

```
package file
var fileClosed bool
```

Imagine that this variable is declared in the package block of the `file` package.

It's a package level variable and therefore it's in a larger scope. Don't use abbreviations there and don't mix caps in the name. `file` starts with a lowercase letter.

Use mixedCaps like this

```
type playerScore struct
```

Use all caps for acronyms

```
var localApi string ✗  
var localAPI string ✓
```

Do not stutter

```
player.PlayerScore ✗  
player.Score ✓
```

Do not use under_scores oR LIKE_THIS

```
const MAX_TIME int ✗  
const MaxTime int ✓  
const N int ✓
```



By [deleted]
cheatography.com/deleted-70653/

Published 15th November, 2018.
Last updated 15th November, 2018.
Page 2 of 4.

Sponsored by **Readable.com**
Measure your website readability!
<https://readable.com>

Abbreviation - Rules

Sound/Spelling

Abbreviations should be **pronounceable**.

Abbreviations should have **at least one vowel**.

Abbreviations should not split up **plosive/liquid** combinations but as **plosive/plosive**, for example, the **ct** in **dictionary** or **pt** in **caption**.

Abbreviations should not have more than **three consonants** in a row and should usually **end in a consonant**, unless the vowel is needed for discrimination, for example, **alg** and **algo**.

All of the letters in the abbreviation should be present in the long form and in the same order, and need not appear in sequence in the long form, for example, **recv** and **receive**.

Abbreviation - Rules (cont)

Length/Meaning and Interpretation

An abbreviation should be less than or equal to half the length of the original form.

Abbreviations should be **at least three letters long**.

Abbreviations should **not be whole words** that mean something else.

Abbreviations should not just consist of the prefix of a word, for example, **sym** for **symbol** or **syl** for **syllable**.

Abbreviations **shouldn't be ambiguous**. However, if the names are different that **no confusion** can result, they are **OK**.

Exceptions/Limitations

There are a few exceptions to the above rules for common, well-established forms.

ct and **pt** can be used for **ction** and **ption** if the abbreviation would be too short otherwise, for example, **act** and **opt**.

There are also other types of prefixing, for example, the three--letter prefixes used to distinguish field names in the same database table.

Examples would include **cusID** for **customer ID** and **ordID** for **order ID**.

Those prefixes don't need to follow the same rules.



Abbreviation	
var a int	// array
var arg []string	// argument
var b []byte	// buffer
var b byte	// byte
var bs bytes	// bytes
var buf []byte	// buffer
var c int	// capacity
var c int	// character
var dst int	// destination
var err error	// error value
var fv string	// flag value
var i int	// index
var l int	// length
var m int	// another number
var msg string	// message
var n int	// number or number of
var num int	// number
var off int	// offset
var op int	// operation
var parsed bool	// parsed ok?
var pkg string	// package
var pos int	// position
var r rune	// rune
var r io.Reader	// reader
var s string	// string
var seen bool	// has seen?
var sep string	// separator

Abbreviation (cont)	
var src int	// source
var str string	// string
var v string	// value
var val string	// value
var w io.Writer	// writer
...the list goes on and on...	

