Magic numbers

What’s a magic number?

A magic number, or a magic value, is an unexplained value used directly in code.

For example, let’s say we have some code that works on student data. A student is represented by a list of values, like `['Bilbo', 'Baggins', 1980, 1990]`. The list elements are: * first name * last name * school start year * school end year

Now, let’s say we have some code that works on the students:

```
# Don't do this!

def years_studied(student):
    return student[3] - student[2]

def num_started_in_year(year, students):
    count = 0
    for student in students:
        if student[2] == year:
            count += 1
    return count
```

You might notice the `3` and `2` in the code. We call these values “magic numbers” because they appear directly in the code, without explanation.

What’s wrong with magic numbers?

First, **there is no explanation**. When reading the code, I have no idea what 3 means.

It’s also **very easy to make a mistake**. I might input some other number by mistake, and because the code is hard to read, nobody will notice.

Finally, **it’s hard to change**. Let’s imagine you want to change the data format, adding a new value before start year. You would need to change `2` to `3` and `3` to `4` in the code. However, you need to be careful to change *only the `2`-s related to students*. It will be pretty easy to make a mistake.

What should I do instead?

You should define these numbers as constants. The example before could be rewritten as:

```
# Do this instead:

FIRST_NAME = 0
LAST_NAME = 1
START_YEAR = 2
```

```
END_YEAR = 3

def years_studied(student):
    return student[END_YEAR] - student[START_YEAR]

def num_started_in_year(year, students):
    count = 0
    for student in students:
        if student[START_YEAR] == year:
            count += 1
    return count
```

Now, the code is clear to understand, and it's harder to make a mistake. You can also easily change the values because they are in one place.

If you have multiple types of objects in your code (for instance, student and user), you could also add a prefix to the names so that they are unambiguous:

```
STUDENT_FIRST_NAME = 0
STUDENT_LAST_NAME = 1
...

USER_ID = 0
USER_FIRST_NAME = 1
USER_LAST_NAME = 2
...
```

Some other examples of magic values

Generally, you should treat every value in you code with some suspicion (except for maybe 0 and 1). Some examples:

- Product prices, like `123.45`
- Number of playing cards (`52`)
  - Also card *names*, like `"ace"` or `"king"`
- Chessboard size (`8`)
- Mathematical constants, like π (`3.1415926...`)
  - These are usually already defined for you; like `math.pi` in Python

Further reading

- Magic number (programming) on Wikipedia
- Software Anti-Patterns: Magic Numbers
- What is a magic number, and why is it bad? on Stack Overflow
- Magic Number on C2 wiki