

Love is love. Quirks and Easter Eggs of the Python programming language.

Wojciech Szlosek

One of the most popular and universal programming languages.

Hides many secrets and non-intuitive situations.

Intuition, however, sometimes rightly fails us (as in mathematics).

Introductory information



Strings are not always obvious (ex. 1 & 2).



```
a = "lol"  
b = "lol"
```

```
a is b  
# True
```

```
a = "lol!"  
b = "lol!"
```

```
a is b  
# False
```



```
'a' * 20 is 'aaaaaaaaaaaaaaaaaaaaa'  
# True
```

```
'a' * 21 is 'aaaaaaaaaaaaaaaaaaaaa'  
# False
```

Strings are not always obvious. Explanation.

Ex. 1:
CPython optimization (called string
interning).

```
a = "lol"
b = "lol"

a is b
# True

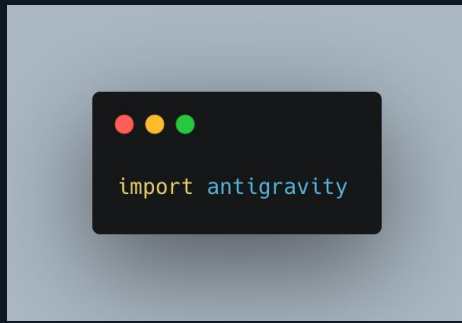
a = "lol!"
b = "lol!"

a is b
# False
```

Ex. 2:
Peephole optimization technique
known as Constant folding.

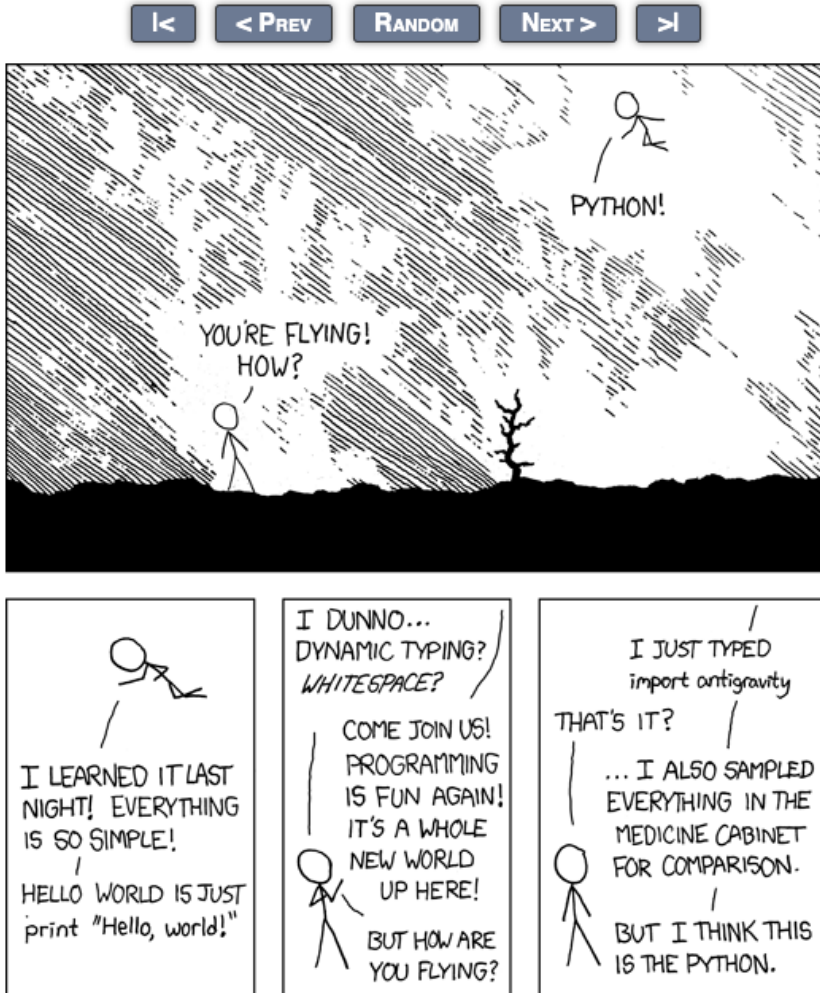
```
'a' * 20 is 'aaaaaaaaaaaaaaaaaaaaa'
# True

'a' * 21 is 'aaaaaaaaaaaaaaaaaaaaa'
# False
```



I believe I can fly.

- `import antigravity` opens up a web browser pointing to the comic about Python
- python developers easter egg
- popular functionality



"goto" is our enemy.

"goto" is bad programming practice for all languages. It is so banned in Python that it cannot be used at this time.



```
from goto import goto, label
for i in range(100):
    if i == 25:
        goto .breakout
    print("The value of i is less than 25.")

label .breakout
print("I am here!")
```

Logical operators are puzzling.

If (True) in [False] is false, why are the other cases different?



```
(False == False) in [False]  
# False
```

```
False == False in [False]  
# True (???)
```

```
True is False == False  
# False
```

Beware of default mutable arguments.

The first two times `num_list()` is invoked, a 1 will be appended to `nums` list both times. The result is `[1, 1]`. To reset the list, you have to pass in an **empty list** to the next invocation.



```
def num_list(nums=[]):  
    num = 1  
    nums.append(num)  
    return nums  
  
print(num_list()) # [1]  
print(num_list()) # [1, 1]  
print(num_list([])) # [1]  
print(num_list()) # [1, 1, 1]  
print(num_list([4])) # [4, 1]
```


It's the Zen of Python!

After typing "insert this" we will get this "text".

The Zen of Python is a collection of 19 "guiding principles" for writing computer programs that influence the design of the Python.



The Zen of Python, by Tim Peters

```
Beautiful is better than ugly.  
Explicit is better than implicit.  
Simple is better than complex.  
Complex is better than complicated.  
Flat is better than nested.  
Sparse is better than dense.  
Readability counts.  
Special cases aren't special enough to break the rules.  
Although practicality beats purity.  
Errors should never pass silently.  
Unless explicitly silenced.  
In the face of ambiguity, refuse the temptation to guess.  
There should be one-- and preferably only one --obvious way to do it.  
Although that way may not be obvious at first unless you're Dutch.  
Now is better than never.  
Although never is often better than *right* now.  
If the implementation is hard to explain, it's a bad idea.  
If the implementation is easy to explain, it may be a good idea.  
Namespaces are one honking great idea -- let's do more of those!
```

Python knows what love is.



```
import this
```

```
love = this  
this is love
```

```
print(love is True) # False  
print(love is False) # False  
print(love is not(True or False)) # True  
  
print(love is love) # True
```

Thank you for your attention.

