Da zero a PYTHON in 45 minuti

ANDREA COLANGELO

UBUNTU DEVELOPER, DEBIAN MAINTAINER

WE'RE OPEN 2013, CESENA – 9 NOVEMBRE 2013

we're open 2013

1

FEATURES

Largamente

UTILIZZATO

BATTERIES

Included

OBJECT

Oriented

ZUCCHERO

Sintattico


INTERPRETATO
non mescolato (semi-cit.)

SHELL
interattiva

2

HELLO WORLD

Hello world in C

```c
#include <stdio.h>

int main(void)
{
    printf("Hello, world!\n");
    return 0;
}
```

Hello world in JAVA

```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

Hello world in PYTHON

```
print "Hello, world!"
```

```
print ("Hello, world!")
```

3

LE BASI


Hands
ON!

4

IF

IF

```
if spam:
    print "It's spam!"
else:
    print "Boo!"
```

IF

```
if foo >= bar:
    print "foo is bigger"
elif foo == bar:
    print "foo equals bar"
else:
    print "bar is bigger"
```

Le insidie della INDENTAZIONE

```
if (foo)
    if (bar)
        do_something(foo, bar);
else
    do_something_else();
```

Le insidie della INDENTAZIONE

```
if (foo) {
    if (bar) {
        do_something(foo, bar);
}
else {
    do_something_else();
}}
```

Le insidie della INDENTAZIONE

```
if (foo)
if (bar)
do_something(foo, bar);
else
do_something_else();
```

Le insidie della INDENTAZIONE

```
if foo:
    if bar:
        do_something(foo, bar)
    else
        do_something_else()
```

5

ERRORI


Look Before You
LEAP

LBYL

```
if denominator == 0:
    print "Oops"
else:
    print numerator/denominator
```

Easier to Ask Forgiveness than

PERMISSION

EAFP

```
try:
    print numerator/denominator
except ZeroDivisionError:
    print "Oops"
```

6

STRUTTURE DATI

LISTE

```
>>> my_list = ['a', 'b', 'c', 'd', 'e']
>>> my_list[2]
'c'
>>> my_list[5]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
>>> my_list[0]
'a'
>>> my_list[-1]
'e'
```

LISTE

```
>>> my_list[0:3]
['a', 'b', 'c']
>>> my_list.append('f')
>>> my_list
['a', 'b', 'c', 'd', 'e', 'f']
>>> my_list.append(45)
>>> my_list
['a', 'b', 'c', 'd', 'e', 'f', 45]
>>> my_list.append(['Foo', 'Bar', 'Spam'])
>>> my_list
['a', 'b', 'c', 'd', 'e', 'f', 45, ['Foo', 'Bar', 'Spam']]
>>> my list[-1][-1]
'Spam'
```

TUPLE

```
>>> my_tuple = (1, 2, 3)
>>> my_tuple
(1, 2, 3)
>>> my_tuple.append('b')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'tuple' object has no attribute 'append'
>>> my_tuple = ('a', [20, 30, 40], True, ("spam"))
>>> my_tuple
('a', [20, 30, 40], True, 'spam')
```

DIR()

```
>>> dir(my_list)
['__add__', '__class__', '__contains__', '__delattr__', '__d
elitem__', '__delslice__', '__doc__', '__eq__', '__format__'
, '__ge__', '__getattribute__', '__getitem__', '__getslice__
', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__',
 '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne
__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '
__reversed__', '__rmul__', '__setattr__', '__setitem__', '__
setslice__', '__sizeof__', '__str__', '__subclasshook__', 'a
ppend', 'count', 'extend', 'index', 'insert', 'pop', 'remove
', 'reverse', 'sort']
```

DIZIONARI

```
>>> my_dict = {}
>>> my_dict[1] = "Lunedi"
>>> my_dict[2] = "Martedi"
>>> my_dict[3] = "Mercoledi"
            ...
>>> my_dict
{1: 'Lunedi', 2: 'Martedi', 3: 'Mercoledi', 4: 'Giovedi', 5:
 'Venerdi', 6: 'Sabato', 7: 'Domenica'}
```

DIZIONARI

```
>>> my_dict[4]
'Giovedi'
>>> my_dict.keys()
[1, 2, 3, 4, 5, 6, 7]
>>> my_dict.values()
['Lunedi', 'Martedi', 'Mercoledi', 'Giovedi', 'Venerdi', '
Sabato', 'Domenica']
>>> my_dict.items()
[(1, 'Lunedi'), (2, 'Martedi'), (3, 'Mercoledi'), (4, 'Gio
vedi'), (5, 'Venerdi'), (6, 'Sabato'), (7, 'Domenica')]
>>> my_dict[(6+10)%7]
'Martedi'
```

7

CICLI

WHILE

```
>>> x = 0
>>> while x < 10:
...         print x,
...         x = x + 1
...
0 1 2 3 4 5 6 7 8 9
```

FOR

```
>>> foo_list = ['x', {}, 42, ("This", "Is", "A", "Tuple")]
>>> for element in foo_list:
...     print element
...
x
{}
42
('This', 'Is', 'A', 'Tuple')
```

FOR

```
>>> for key in my_dict:
...     print key,
...
1 2 3 4 5 6 7
>>> for key in my_dict:
...     print my_dict[key],
...
Lunedi Martedi Mercoledi Giovedi Venerdi Sabato Domenica
```

FOR

```
>>> for key, value in my_dict.items():
...     print key, value
...
1 Lunedi
2 Martedi
3 Mercoledi
4 Giovedi
5 Venerdi
6 Sabato
7 Domenica
```

LIST COMPREHENSIONS

```
numbers = [x for x in range(100)]

odds = [x for x in range(100) if x % 2]

squares = [x**2 for x in range(100)]
```

8

FUNZIONI

FUNZIONI

```
def spam():
    print "Hello, spam!"
```

FUNZIONI

```
def cube(x):
    return x ** 3
```

FUNZIONI

```
>>> def cube(x):
...     return x ** 3
...
>>> cube(2)
8
>>> cube(5)
125
>>> cube(10000)
1000000000000
>>> cube(cube(7))
40353607
```

FUNZIONI

```
>>> def area_triangle(base, height=10):
...     return base * height / 2.0
...
>>> area_triangle(5)
25.0
>>> area_triangle(5, 5)
12.5
```


In order to understand recursion, one must first understand recursion

RICORSIONE

FATTORIALE

$$n! = \prod_{k=1}^{n} k = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$$

FATTORIALE

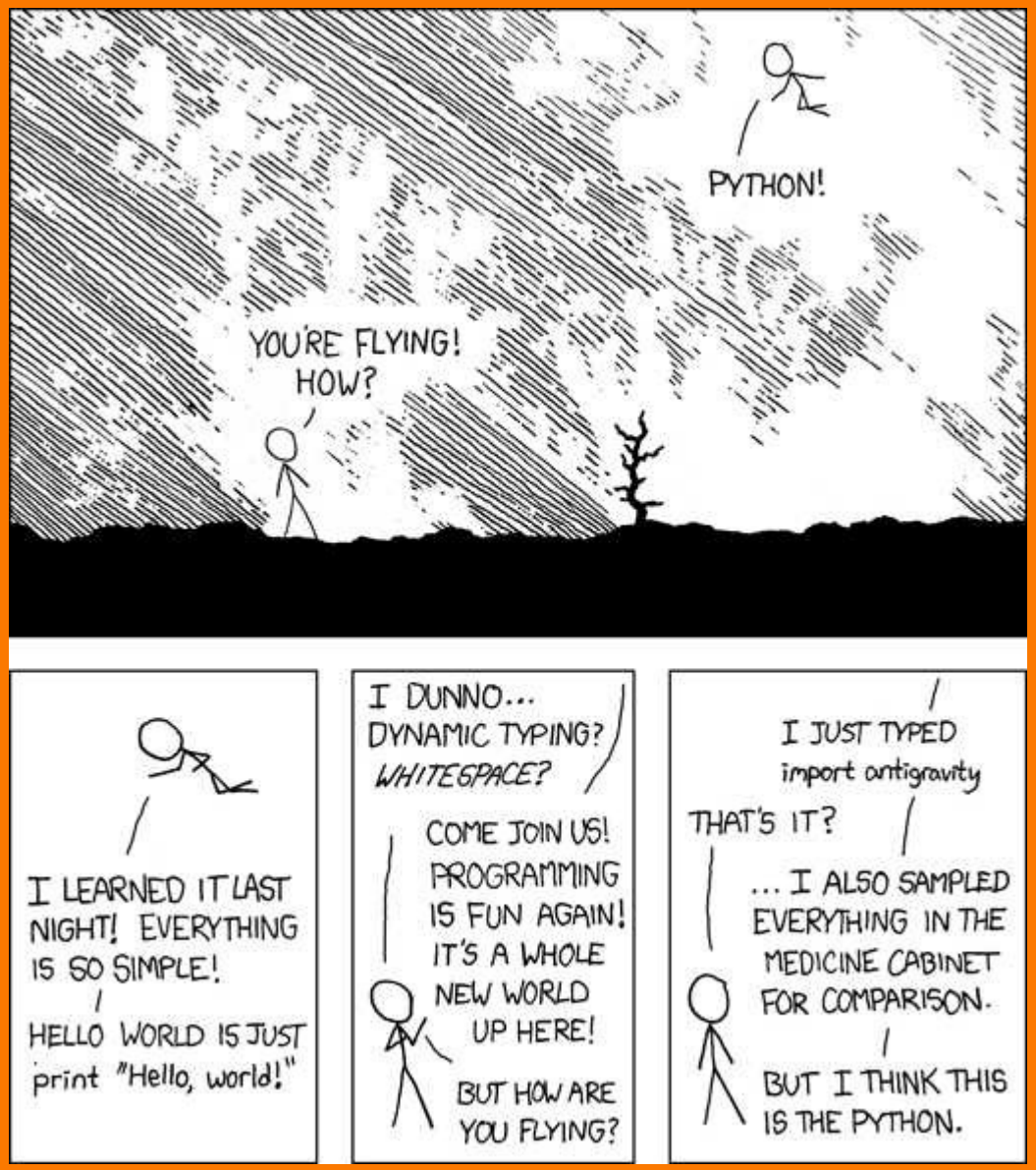
```
def factorial(n):
    result = 1
    for number in range(1, n+1):
        result = result * number
    return result
```

FATTORIALE

```
def factorial(n):
    if n <= 1:
        return 1
    else:
        return n * factorial(n-1)
```

IMPORT

```
>>> random.randint(1, 100)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'random' is not defined
>>> import random
>>> random.randint(1, 100)
83
```



http://xkcd.com/353/

9

OOP

OOP

```
class Superhero():
    def __init__(self, name):
        self.name = name

    def set_can_fly(self, option):
        self.can_fly = option

batman = Superhero("Batman")
batman.set_can_fly(False)
```

10

DOCS

DOCSTRING

```
class Superhero():
    """The class that will save the world"""

    def __init__(self, name):
        """The constructor of our hero"""
        self.name = name

    def set_can_fly(self, option):
        """Method to enable flying superpower"""
        self.can_fly = option
```

HELP()

```
Help on module superhero:

NAME
    superhero

FILE
    /home/warp10/superhero.py

CLASSES
    Superhero

    class Superhero
     |  The class that will save the world
     |
     |  Methods defined here:
     |
     |  __init__(self, name)
     |      The constructor of our hero
     |
     |  set_can_fly(self, option)
     |      Method to enable flying superpower
```

DOCS

dir()

help()

www.python.org

docs.python.org

www.diveintopython3.net

TODO

TODO

Decoratori

Funzioni Lambda

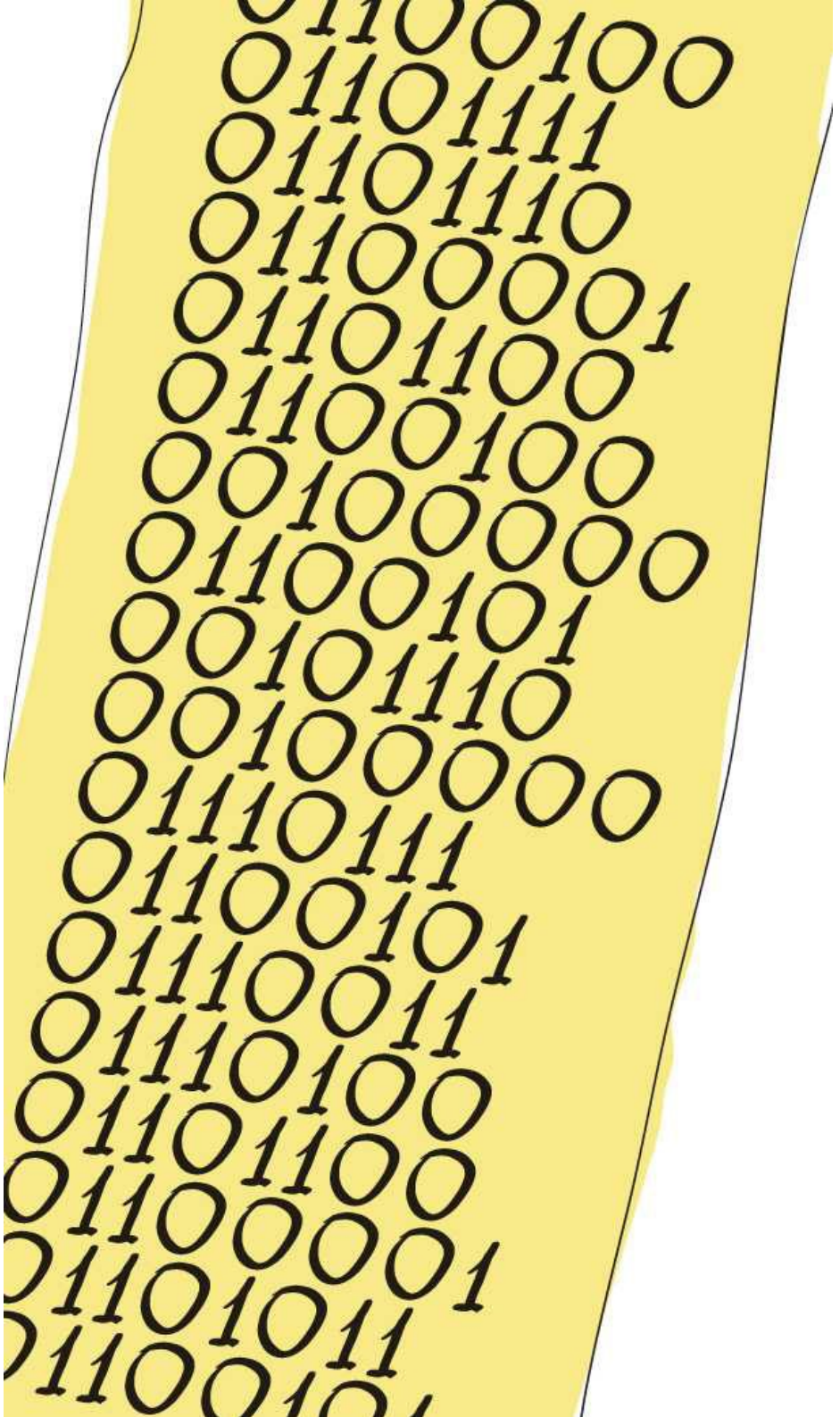
Generatori

Test

...

12

ONE-LINERS



Tratto da “Fahrenheit 2013”, di Makkox
http://tinyurl.com/odoxv9v
Cortesia dell'autore

MAKKOX

```
"".join(chr(int(x, 2)) for x in open("/tmp/makkox")
```

WEB SERVER

```
import SimpleHTTPServer
SimpleHTTPServer.test()
```

```
import http.server
http.server.test()
```

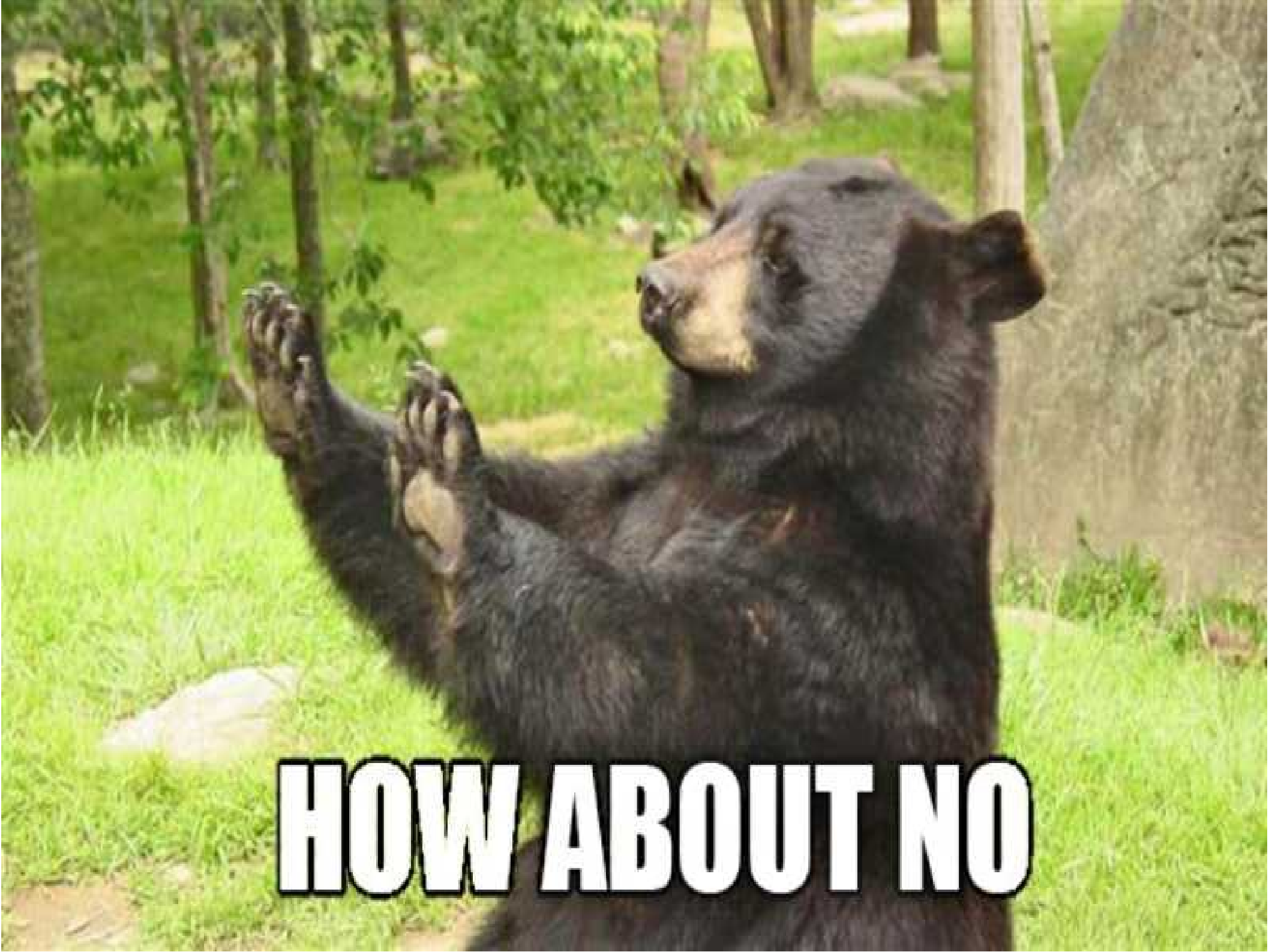
UNIX USERS

```
print '\n'.join(line.split(":",1)[0] for line in open("/etc/passwd"))
```

CRIVELLO DI ERATOSTENE

```
n = 100

print sorted(set(range(2,n+1)).difference(set((p * f) for p in
range(2,int(n**0.5) + 2) for f in range(2,(n/p)+1))))
```


HOW ABOUT NO

ANDREA COLANGELO

@andreacolangelo

warp10@ubuntu.com

andreacolangelo.com