CS 314 Principles of Programming Languages

Guang Wang

Department of Computer Science

Rutgers University

Deterministic &Nondeterministic FAs

Deterministic FA (DFA)

– At most one transition for any state / character pair

– Every transition consumes one input character

Nondeterministic FA (NFA):

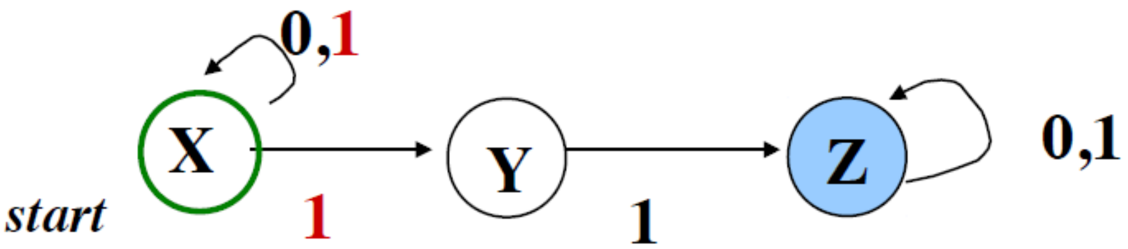
– For some state / character: more than one transition and / or

– ε moves: transition that does not consume a character

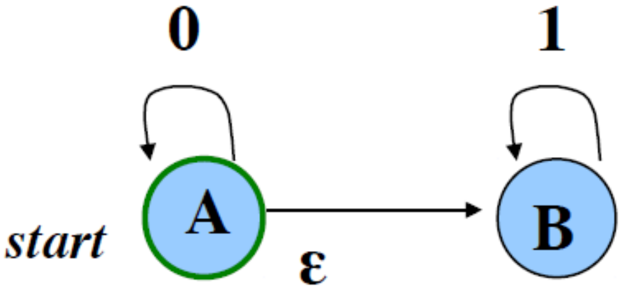
– NFA accepts a string if *ANY* sequence of allowed choices ends in an accepting state

NFAs

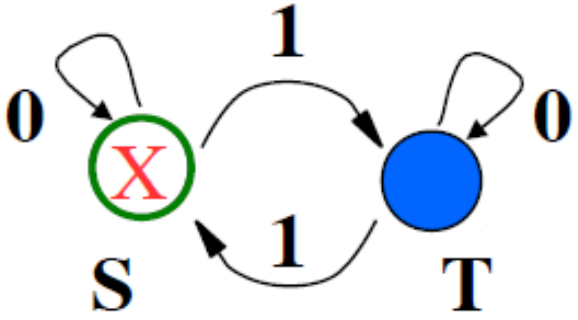
Regular Expression: (0 | 1) * 1 1 (0 | 1) *



Regular Expression: 0* 1*



DFA



DFA & NFA

why would I want to turn an NFA into a DFA?

Because I am doing it in a general way that turns any NFA into a DFA, and thereby showing that the DFA is no less powerful than an NFA.

Besides, with no way to “guess” the right transition to take from any states, NFA would need backtracking, to avoid such complex and time consuming, we need transfer NFA to DFA

RE => NFA => DFA => RE

RE to NFA

– Build an NFA for each operand

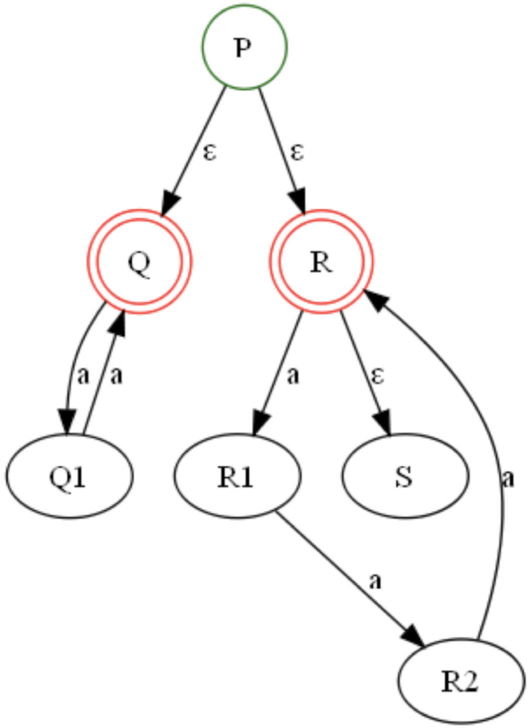
– Put them together in a way that depends on the operator

NFA to DFA

Each state in DFA corresponds to a *set* of states in the NFA

ε-Closure

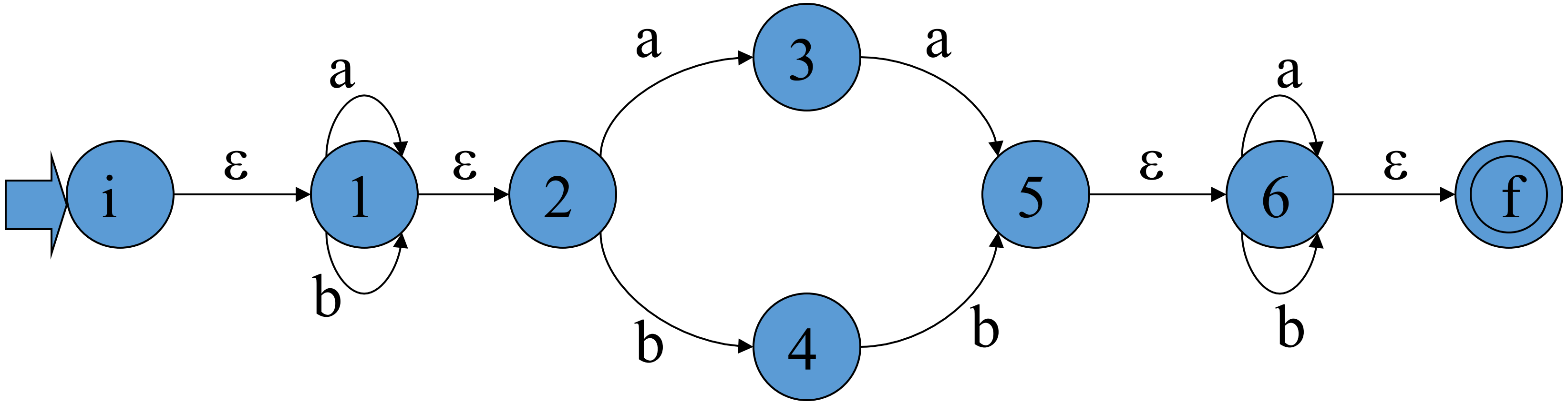
The ε-*closure* of the state q, denoted ECLOSE(q), is the set that contains q, together with all states that can be reached starting at q *by following only* ε-*transitions*.

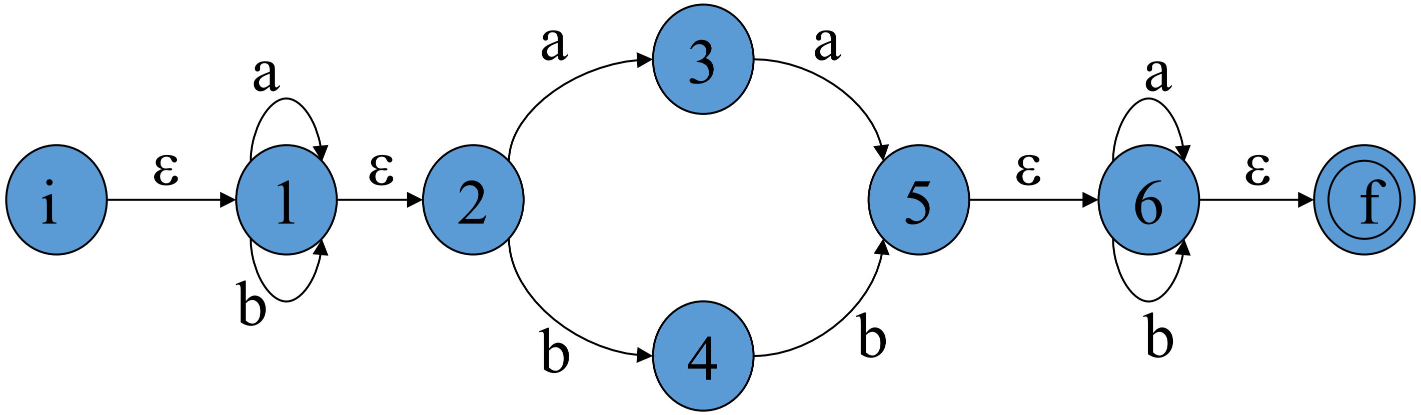


- In the above example:
- ECLOSE(P) ={P,Q,R,S}
- ECLOSE(R)={R,S}
- ECLOSE(x)={x} for the remaining 5 states {Q,Q1,R1,R2,R2}

Example

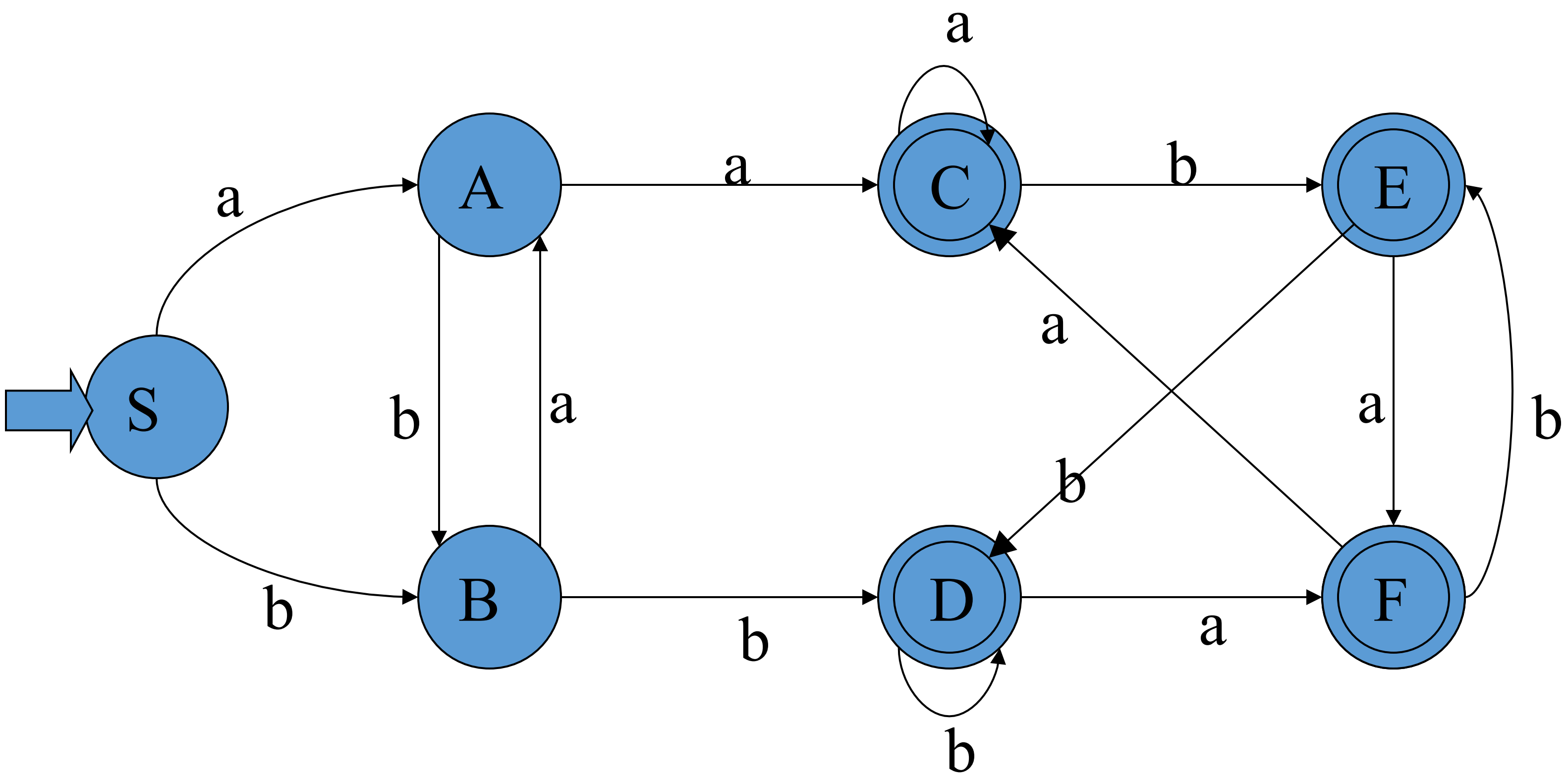
NFA





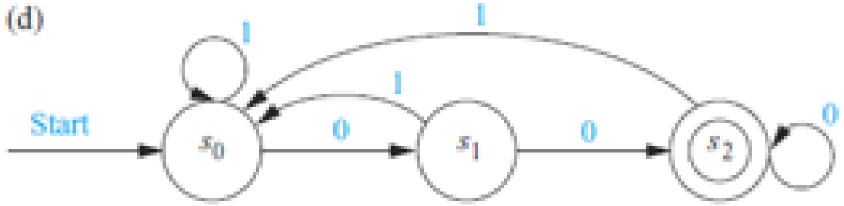
	Ia	Ib
{i,1,2} S	{1,2,3} A	{1,2,4} B
{1,2,3} A	{1,2,3,5,6,f} C	{1,2,4} B
{1,2,4} B	{1,2,3} A	{1,2,4,5,6,f} D
{1,2,3,5,6,f} C	{1,2,3,5,6,f} C	{1,2,4,6,f} E
{1,2,4,5,6,f} D	{1,2,3,6,f} F	{1,2,4,5,6,f} D
{1,2,4,6,f} E	{1,2,3,6,f} F	{1,2,4,5,6,f} D
{1,2,3,6,f} F	{1,2,3,5,6,f} C	{1,2,4,6,f} E

DFA

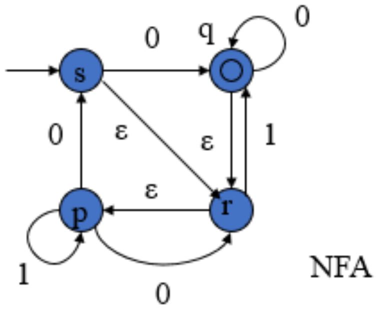


Practice

1. Construct a DFA that recognize the set of bit strings that end with two 0s



2. Convert the following NFA to DFA.



What is Python

- Multi-purpose (Web, GUI, Scripting, etc.)
- Object Oriented
- Interpreted
- Strongly typed and Dynamically typed
- Focus on readability and productivity

Features

- Everything is an Object
- Interactive Shell
- Strong Introspection
- Cross Platform
- PyCharm, Pydev + Eclipse, Wing IDE, Spyder Python, Vim

Who Uses Python

- Google
- PBS
- NASA
- Library of Congress
- the ONION
- ...the list goes on...

Hello World

```
print(“Hello World!”)
print(‘Hello World!’)
print(“‘Hello World!’ ’ ’)
```

```
Hello World!
```

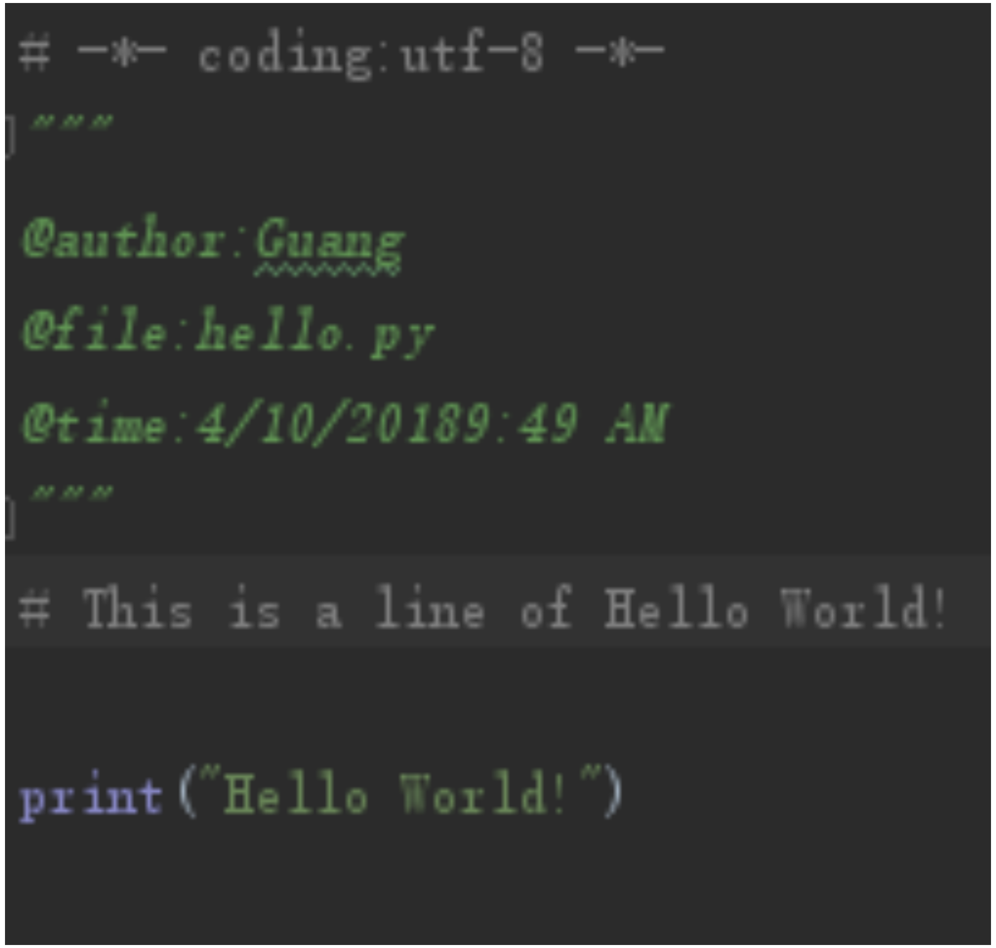
Comments

One line comment

Pound sign: #

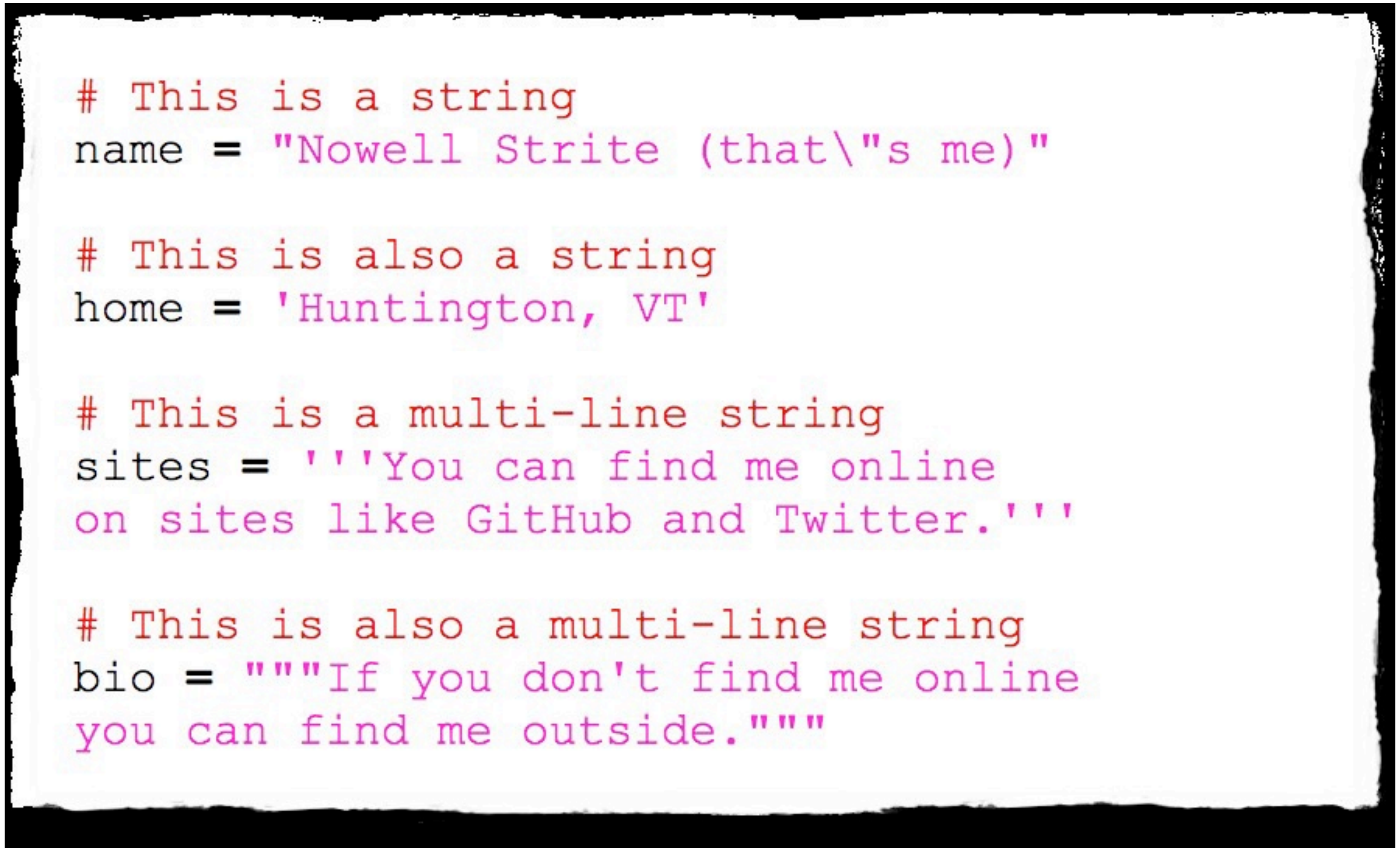
Multi-line comment

“““   ”””



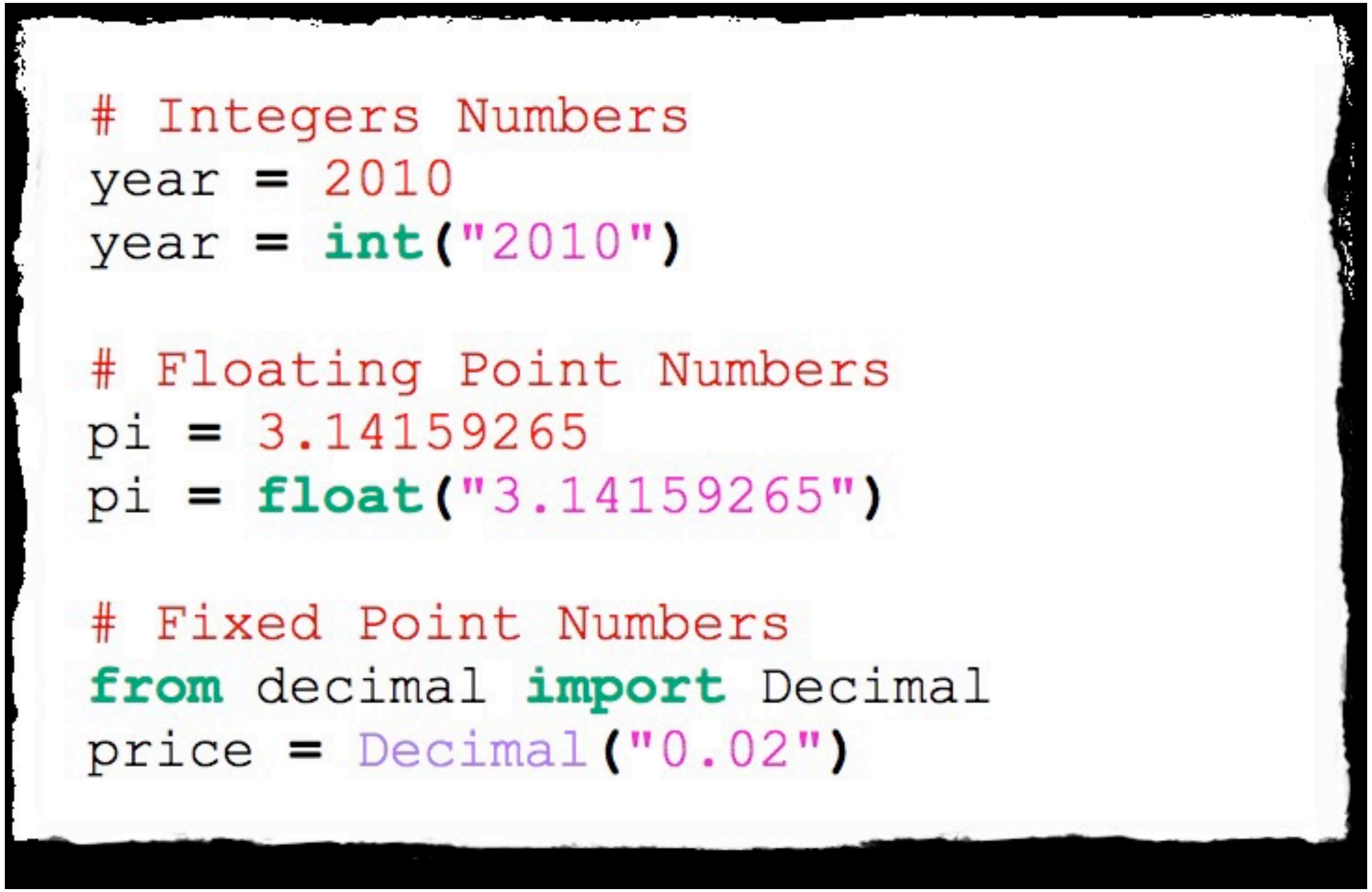
```
# -*- coding:utf-8 -*-
"""

@author:Guang
@file:hello.py
@time:4/10/20189:49 AM
"""
# This is a line of Hello World!


print("Hello World!")
```

Strings



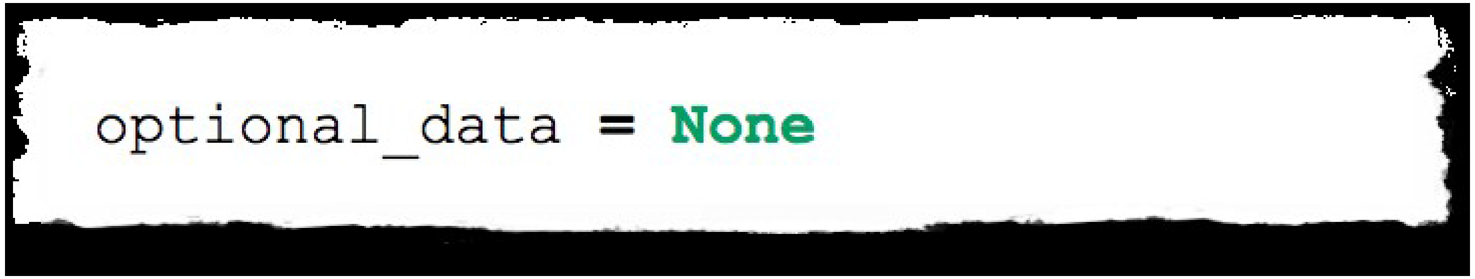
```
# This is a string
name = "Nowell Strite (that\"s me)"

# This is also a string
home = 'Huntington, VT'

# This is a multi-line string
sites = '''You can find me online
on sites like GitHub and Twitter.'''

# This is also a multi-line string
bio = """If you don't find me online
you can find me outside."""
```

Numbers



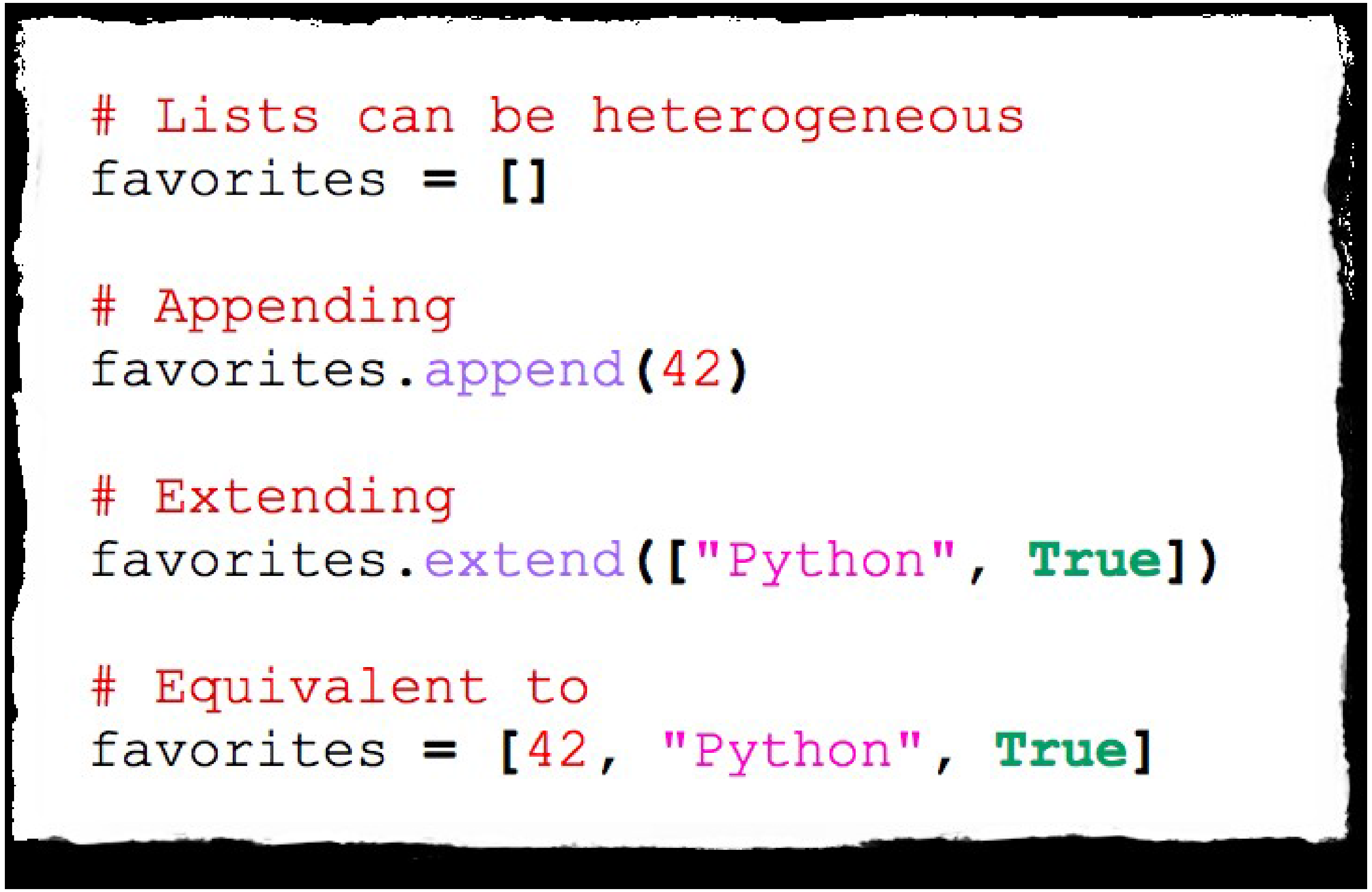
```
# Integers Numbers
year = 2010
year = int("2010")

# Floating Point Numbers
pi = 3.14159265
pi = float("3.14159265")

# Fixed Point Numbers
from decimal import Decimal
price = Decimal("0.02")
```

Null



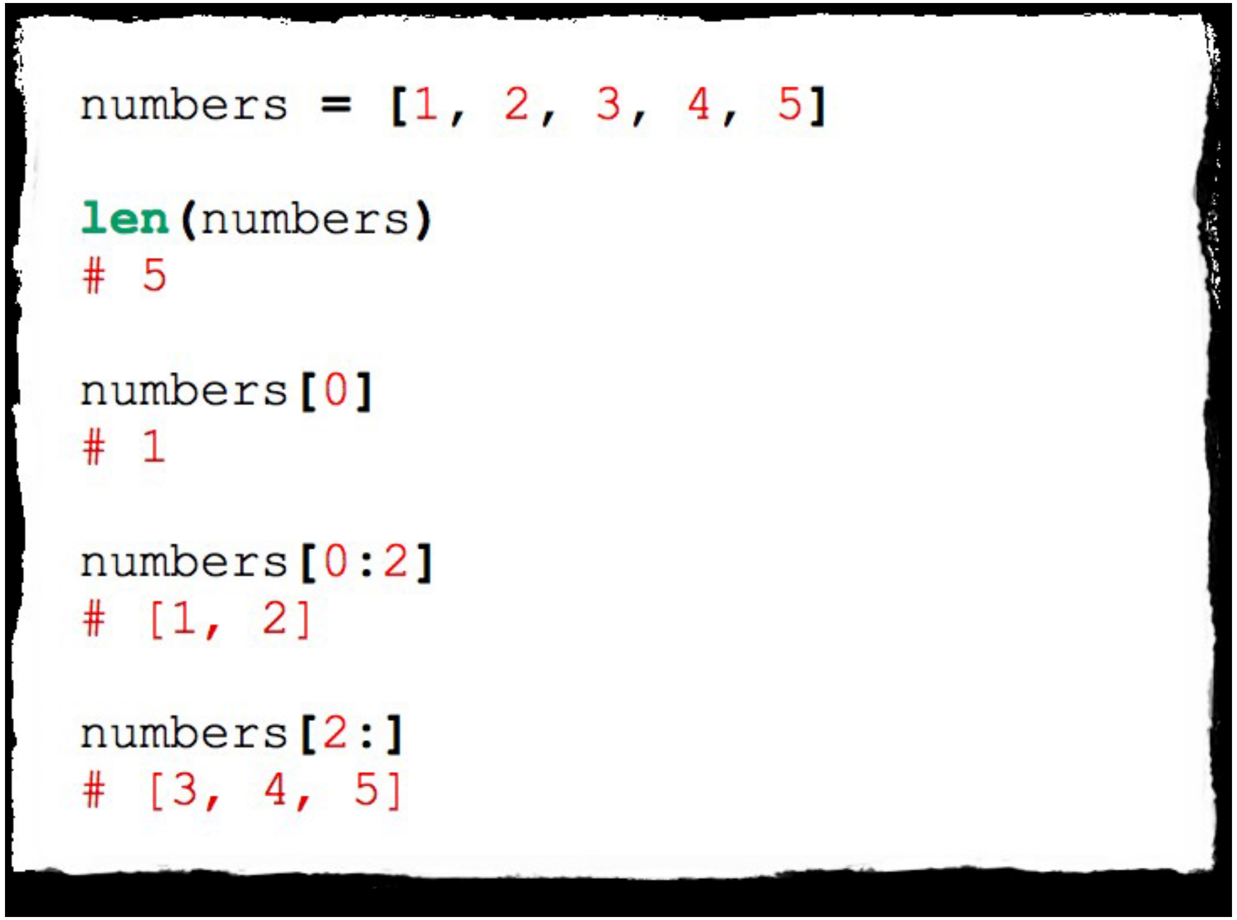
```
optional_data = None
```

Lists



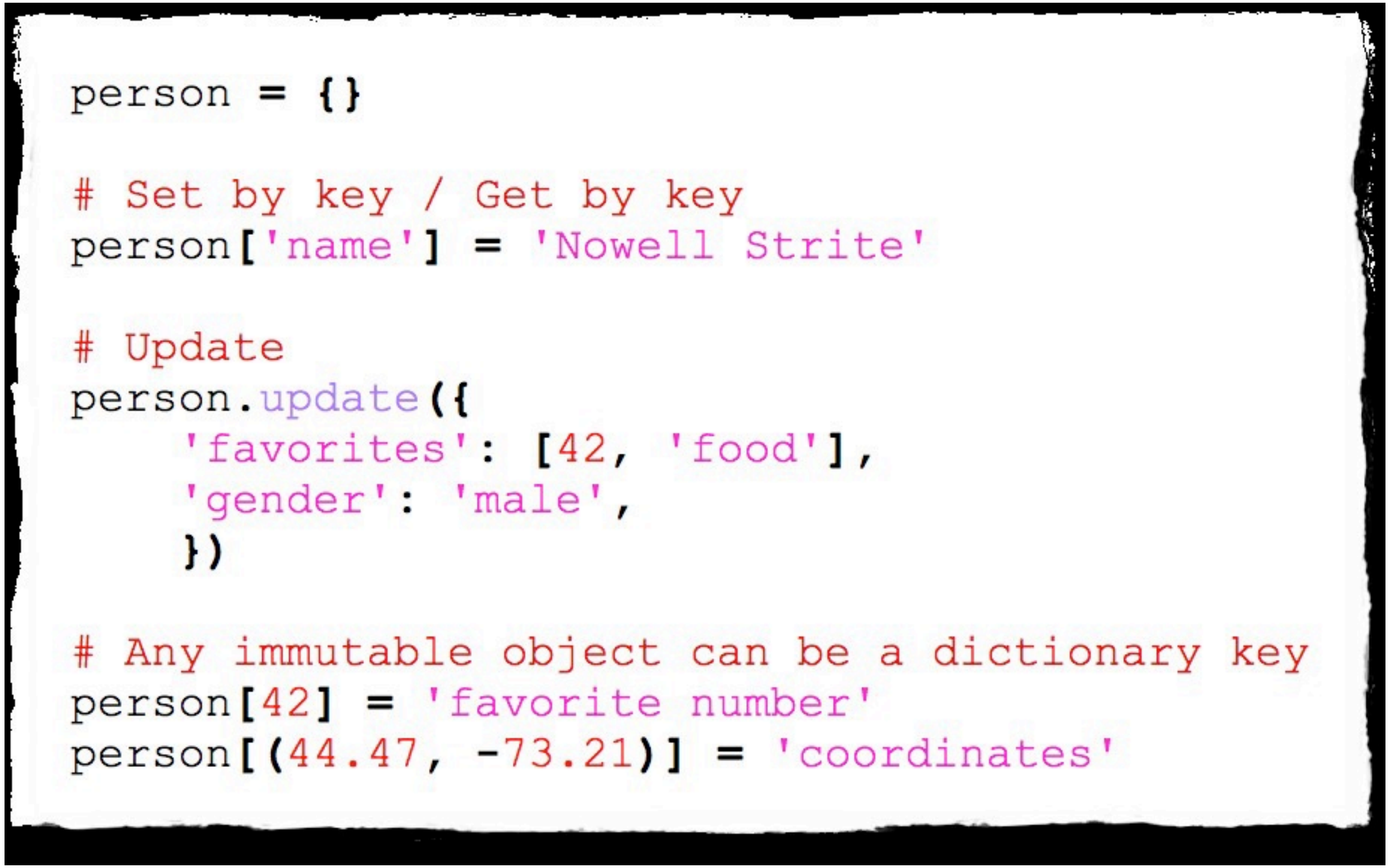
```
# Lists can be heterogeneous
favorites = []

# Appending
favorites.append(42)

# Extending
favorites.extend(["Python", True])

# Equivalent to
favorites = [42, "Python", True]
```

Lists



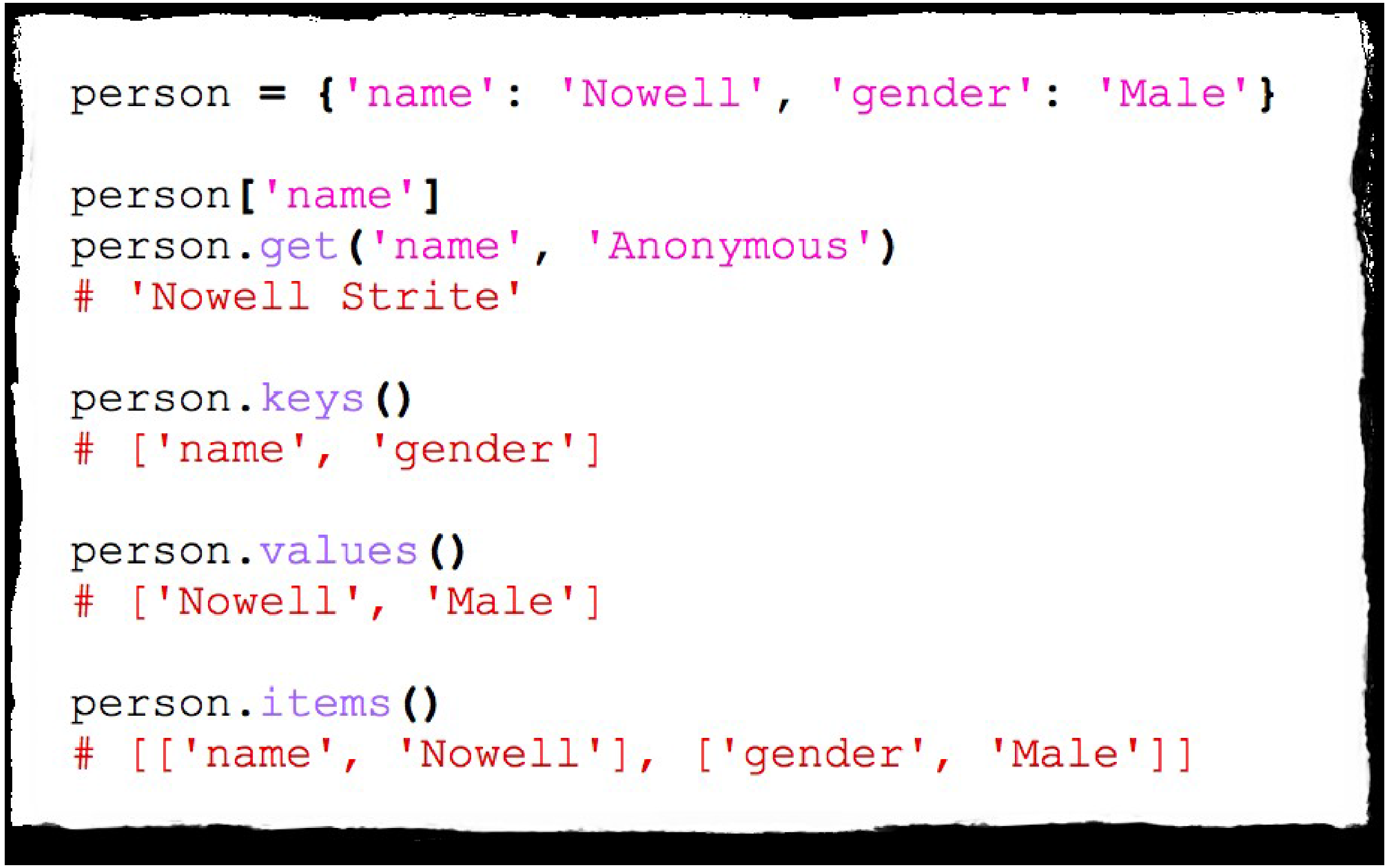
```
numbers = [1, 2, 3, 4, 5]

len(numbers)
# 5

numbers[0]
# 1

numbers[0:2]
# [1, 2]

numbers[2:]
# [3, 4, 5]
```

Dictionaries



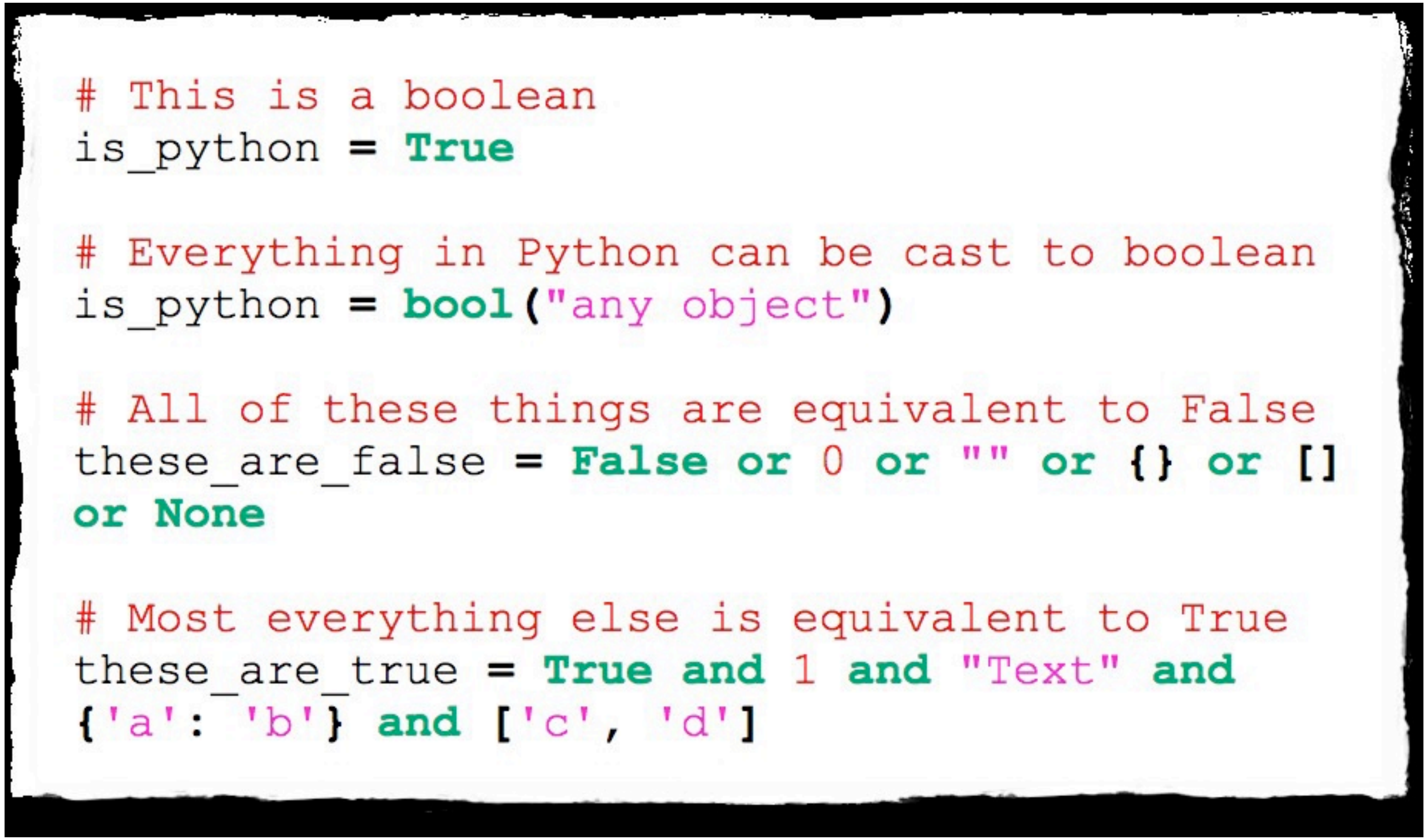
```
person = {}

# Set by key / Get by key
person['name'] = 'Nowell Strite'

# Update
person.update({
    'favorites': [42, 'food'],
    'gender': 'male',
    })

# Any immutable object can be a dictionary key
person[42] = 'favorite number'
person[(44.47, -73.21)] = 'coordinates'
```

Dictionaries



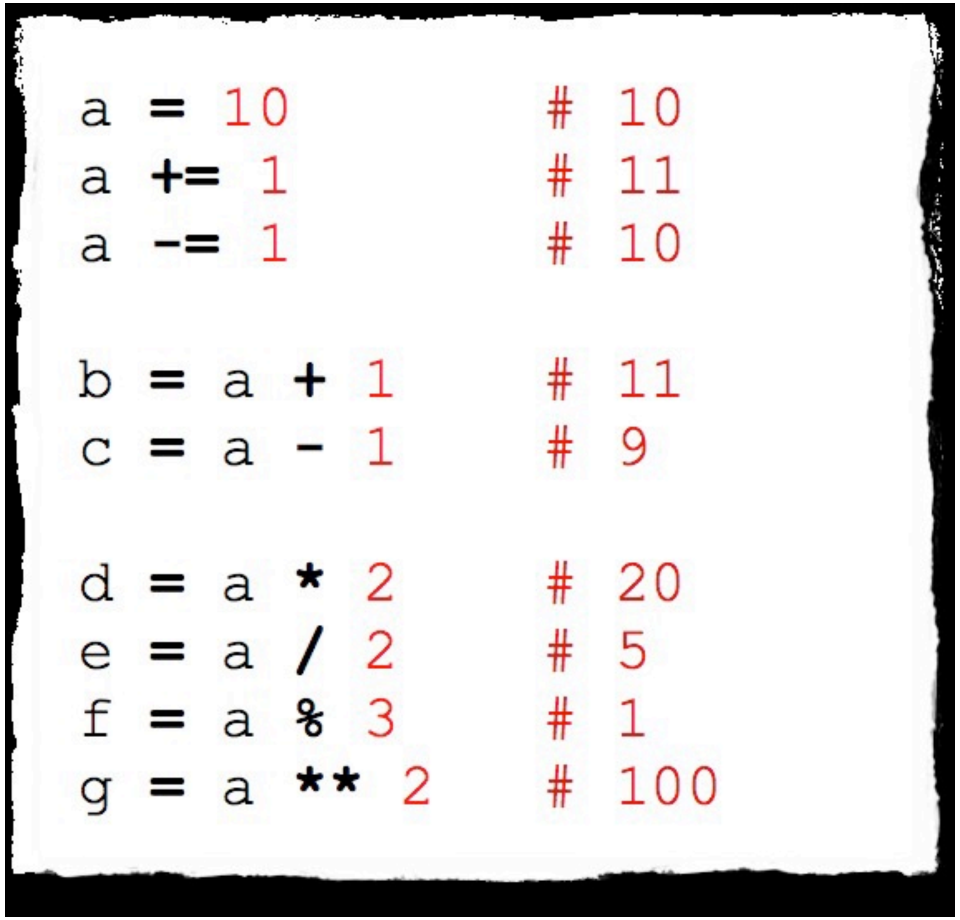
```
person = {'name': 'Nowell', 'gender': 'Male'}

person['name']
person.get('name', 'Anonymous')
# 'Nowell Strite'

person.keys()
# ['name', 'gender']

person.values()
# ['Nowell', 'Male']

person.items()
# [['name', 'Nowell'], ['gender', 'Male']]
```

Booleans



```
# This is a boolean
is_python = True

# Everything in Python can be cast to boolean
is_python = bool("any object")

# All of these things are equivalent to False
these_are_false = False or 0 or "" or {} or []
or None

# Most everything else is equivalent to True
these_are_true = True and 1 and "Text" and
{'a': 'b'} and ['c', 'd']
```

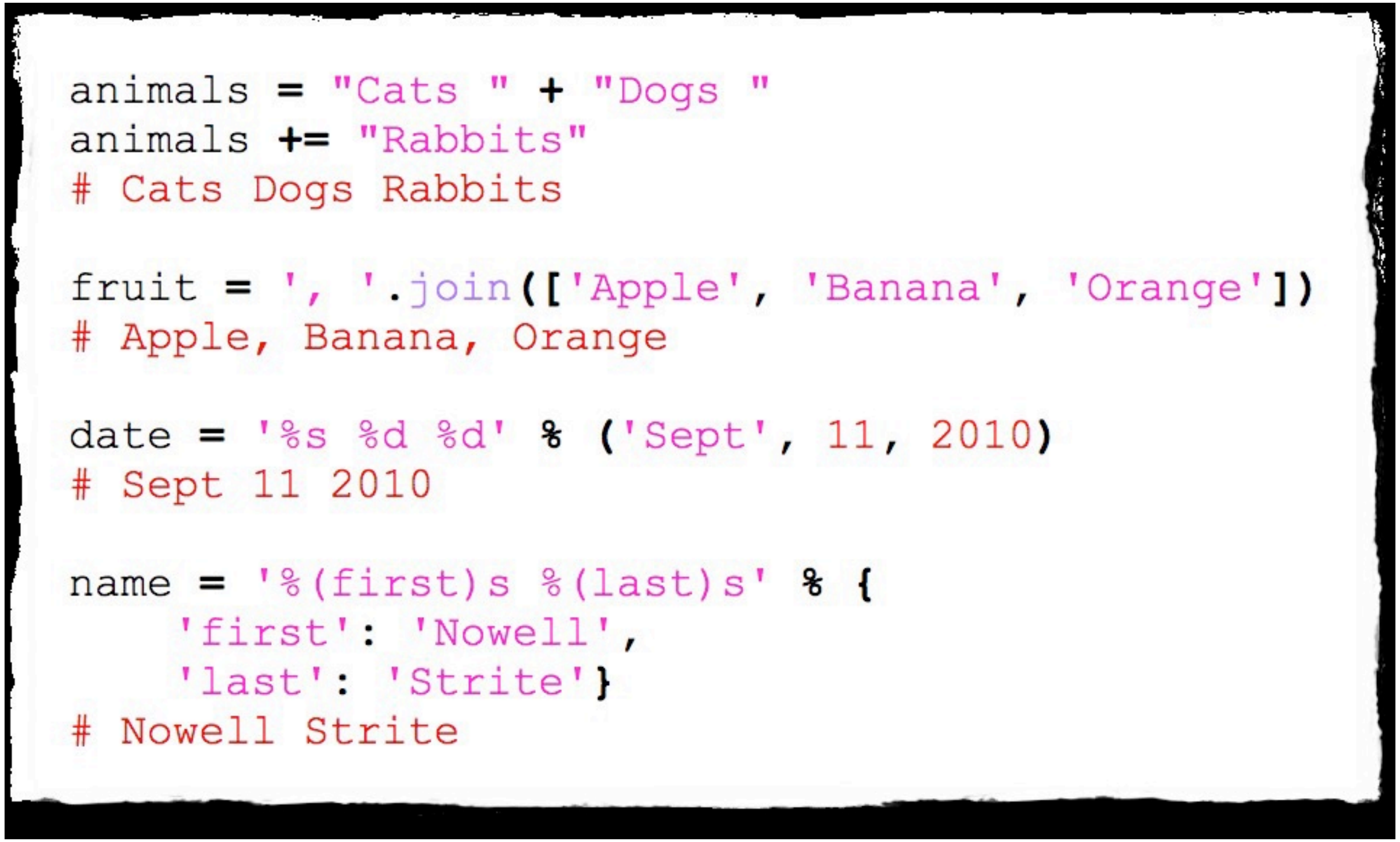
Operators



```
a = 10          # 10
a += 1          # 11
a -= 1          # 10

b = a + 1       # 11
c = a - 1       # 9

d = a * 2       # 20
e = a / 2       # 5
f = a % 3       # 1
g = a ** 2      # 100
```

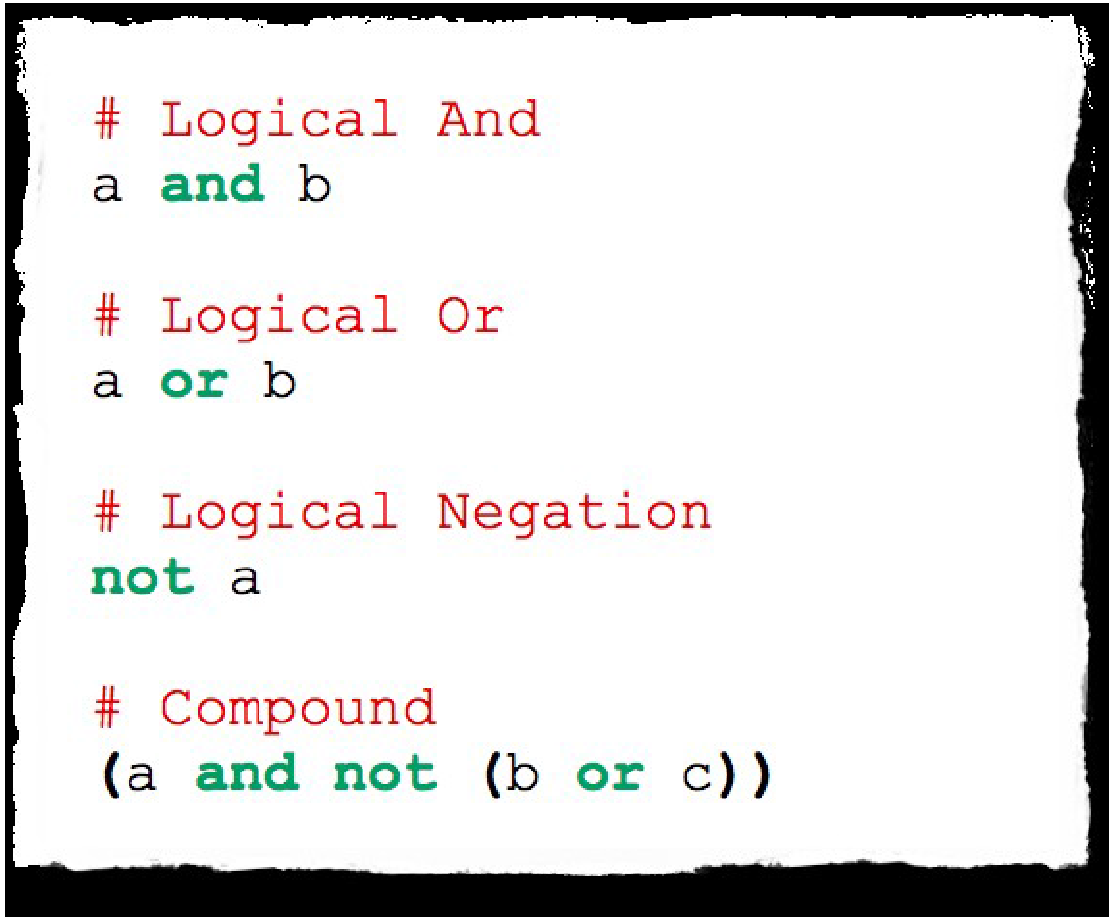
Arithmetic

String Manipulation



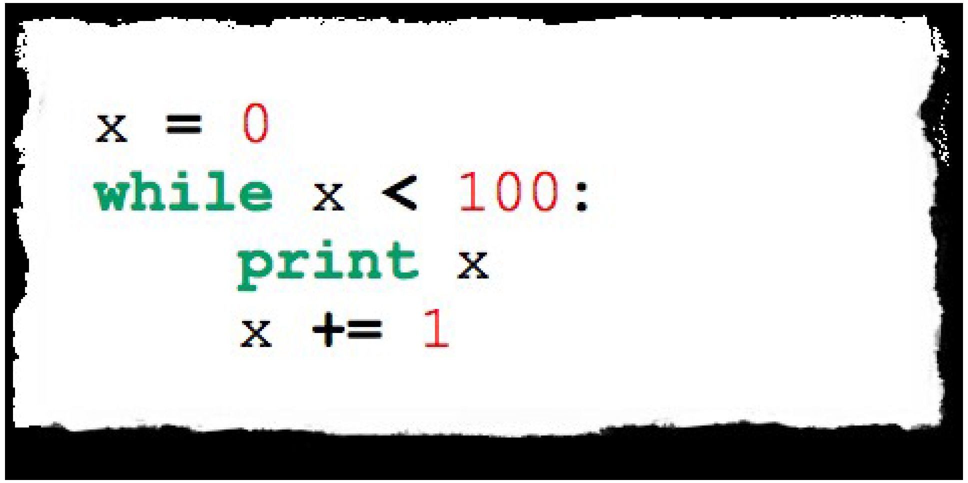
```
animals = "Cats " + "Dogs "
animals += "Rabbits"
# Cats Dogs Rabbits

fruit = ', '.join(['Apple', 'Banana', 'Orange'])
# Apple, Banana, Orange

date = '%s %d %d' % ('Sept', 11, 2010)
# Sept 11 2010

name = '%(first)s %(last)s' % {
    'first': 'Nowell',
    'last': 'Strite'}
# Nowell Strite
```

Logical Comparison



```
# Logical And
a and b

# Logical Or
a or b

# Logical Negation
not a

# Compound
(a and not (b or c))
```

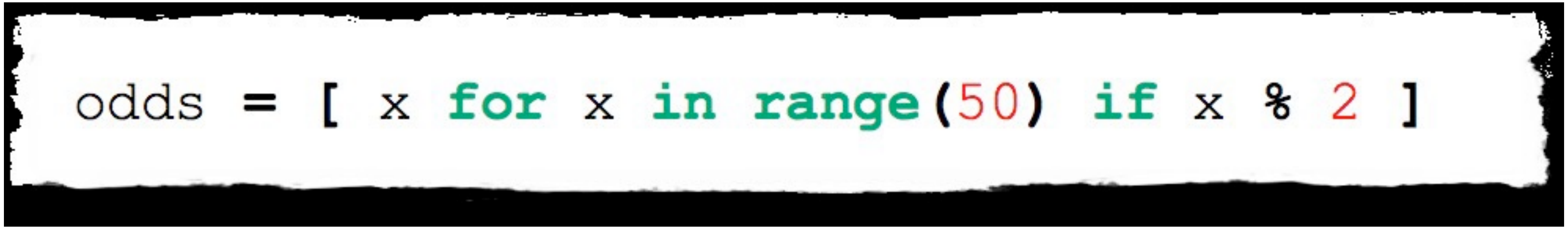
While Loop



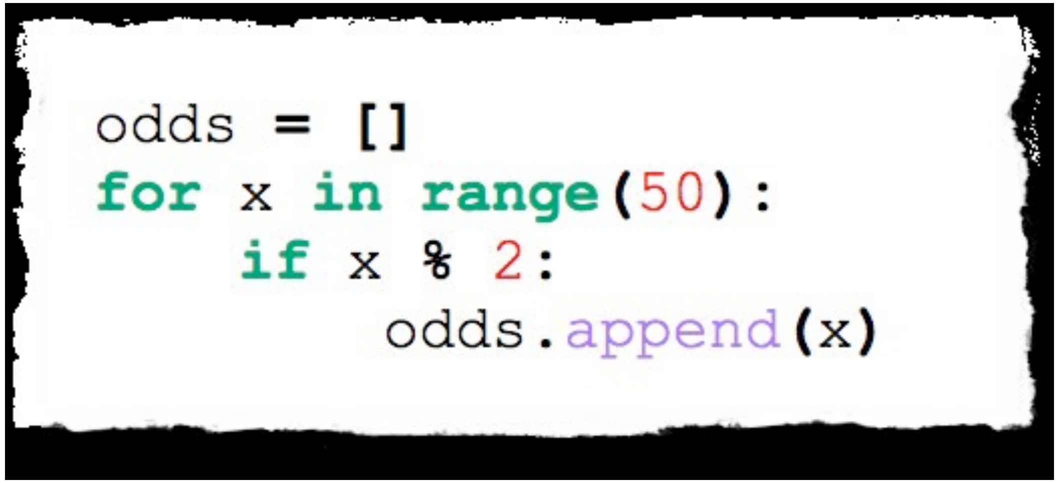
```
x = 0
while x < 100:
    print x
    x += 1
```

Output 0~99 integers

Useful for replacing simple for-loops



```
odds = [ x for x in range(50) if x % 2 ]
```



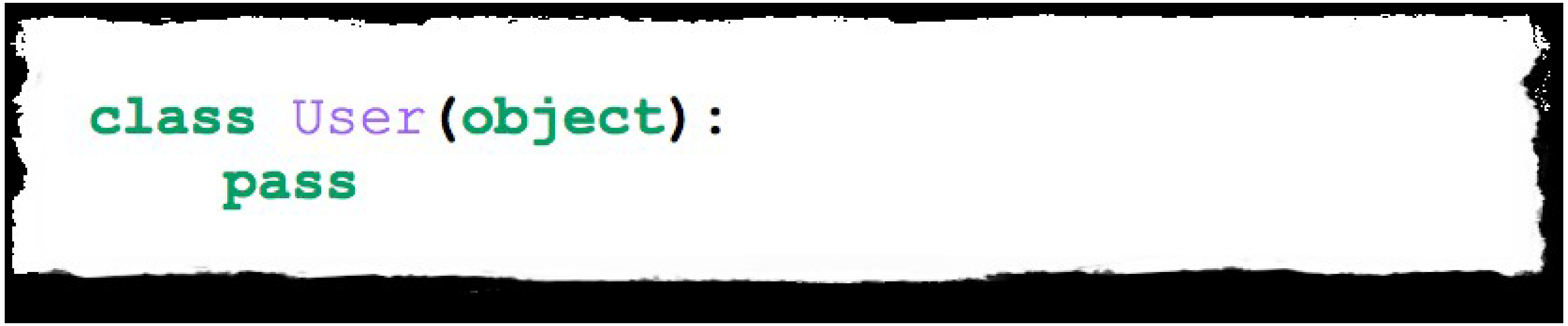
```
odds = []
for x in range(50):
    if x % 2:
        odds.append(x)
```

[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49]

Fibonacci

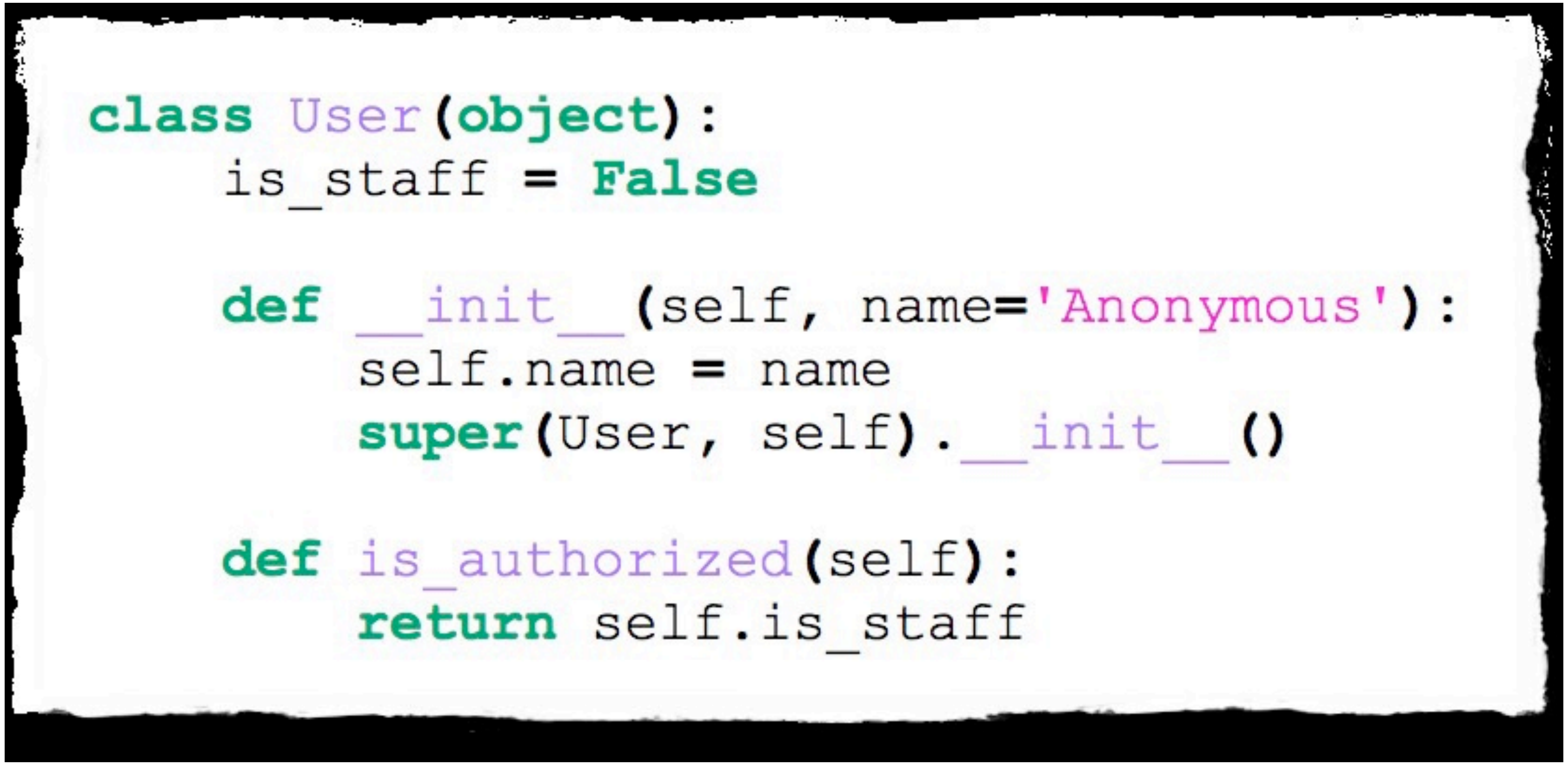
```
def F(n):
    if n == 0: return 0
    elif n == 1: return 1
    else: return F(n-1)+F(n-2)
print(F(10))
```

Classes Declaration



```
class User(object):
    pass
```

Class Methods

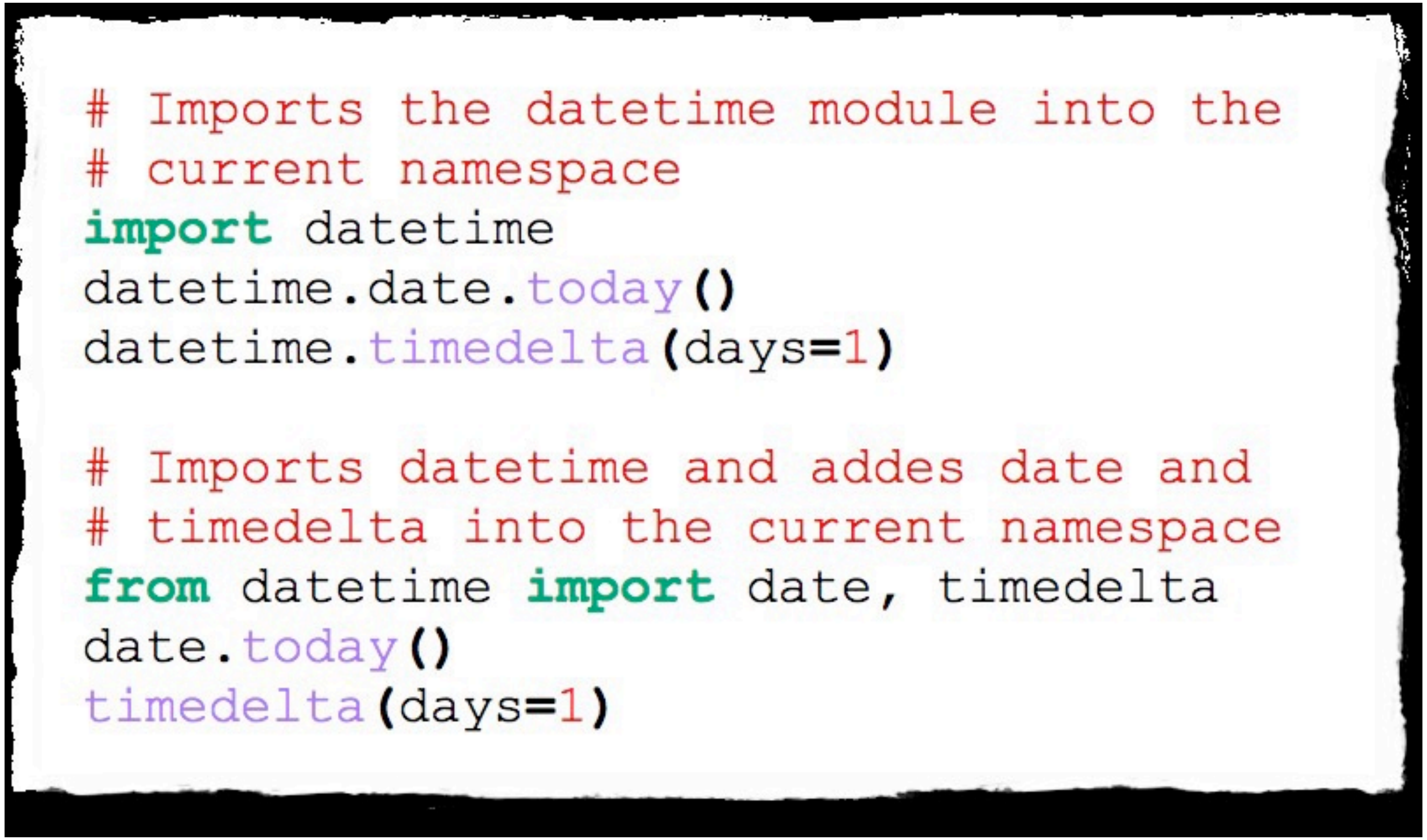


```
class User(object):
    is_staff = False

    def __init__(self, name='Anonymous'):
        self.name = name
        super(User, self).__init__()

    def is_authorized(self):
        return self.is_staff
```

Imports

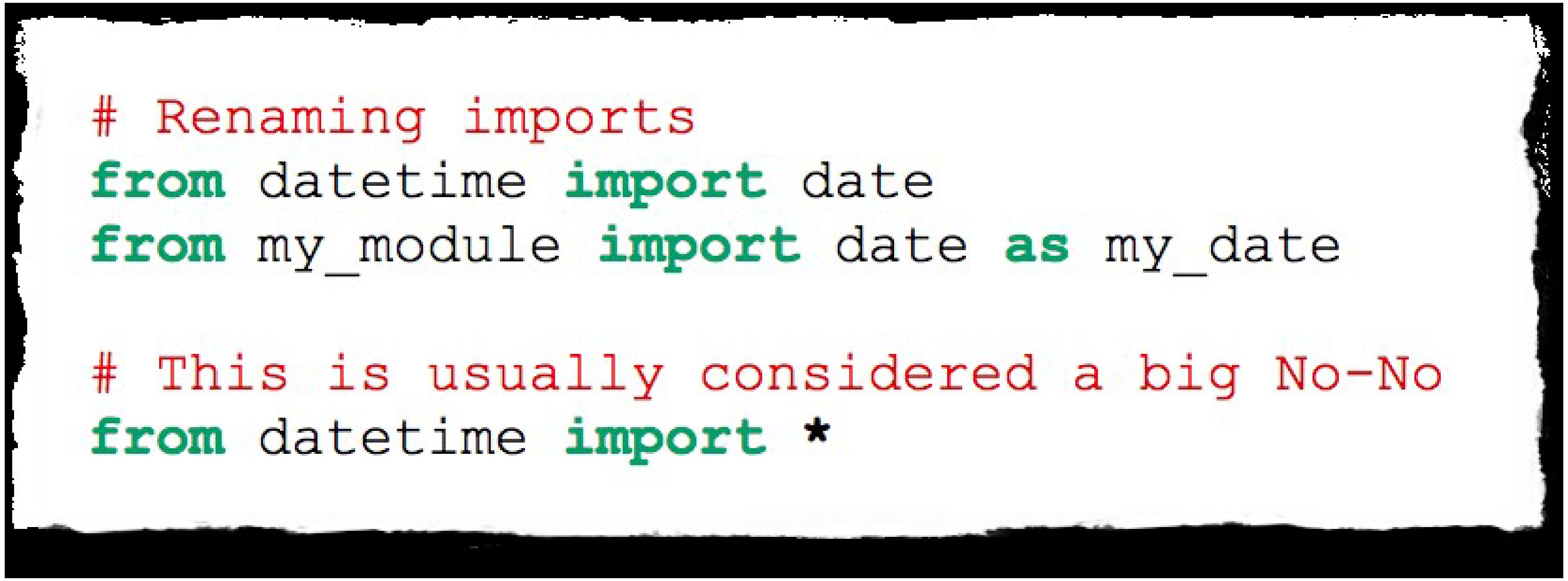
- Allows code isolation and re-use
- Adds references to
- variables/classes/functions/etc. into current
- namespace

Imports



```
# Imports the datetime module into the
# current namespace
import datetime
datetime.date.today()
datetime.timedelta(days=1)

# Imports datetime and addes date and
# timedelta into the current namespace
from datetime import date, timedelta
date.today()
timedelta(days=1)
```

More Imports

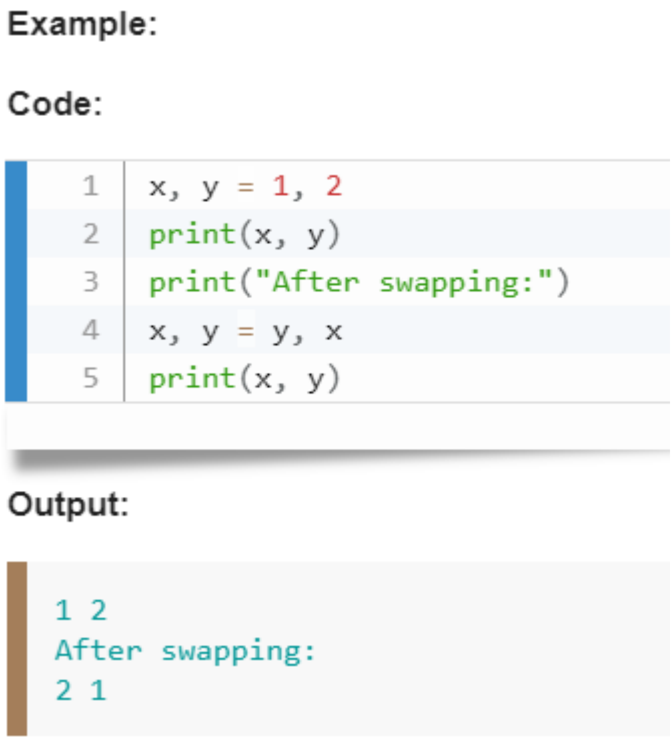


```
# Renaming imports
from datetime import date
from my_module import date as my_date

# This is usually considered a big No-No
from datetime import *
```

Swap

Python provides the following way to swap two variables:

x, y = y, x



Example:

Code:

```python
x, y = 1, 2
print(x, y)
print("After swapping:")
x, y = y, x
print(x, y)
```

Output:

```
1 2
After swapping:
2 1
```

Thanks!!!