π

Welcome back

Python course 2016 - Week 2 - Day 5

Lucas Sinclair
Alexander Eiler
Ludovic Dutoit

Git cheat sheet

```
$ git clone git@github.com:xapple/python_homework.git

$ cd python_homework

$ vim README.md

$ git status

$
      $ git commit -a -m "Added all my changes"
$

$ git push

[later, if collaborative project]
$ git pull
```

Exercise correction

Live demo

Zen of Python

```
>>> import this
```

```
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!
```

Beautiful is better than ugly

```
"""
Give me a function that takes a list of numbers and returns only
the even ones, divided by two.
"""

#---------------------------#

halve_evens_only = lambda nums: map(lambda i: i/2, filter(lambda
i: not i%2, nums))

#---------------------------#

def halve_evens_only(nums):
    return [i/2 for i in nums if not i % 2]


```

Explicit is better than implicit

```python
"""Load the cat, dog, and mouse models so we can edit instances
of them."""

def load():
    from menagerie.cat.models import *
    from menagerie.dog.models import *
    from menagerie.mouse.models import *

#---------------------------#

def load():
    from menagerie.models import cat as cat_models
    from menagerie.models import dog as dog_models
    from menagerie.models import mouse as mouse_models


```

Simple is better than complex

```
"""
Can you write out these measurements to disk?
"""

measurements = [
{'weight': 392.3, 'color': 'purple', 'temperature': 33.4},
{'weight': 34.0, 'color': 'green', 'temperature': -3.1}
]
```

Simple is better than complex

```python
def store(measurements):
    import sqlalchemy
    import sqlalchemy.types as sqltypes
    db = sqlalchemy.create_engine('sqlite:///measurements.db')
    db.echo = False
    metadata = sqlalchemy.MetaData(db)
    table = sqlalchemy.Table('measurements', metadata,
        sqlalchemy.Column('id', sqltypes.Integer, primary_key=True),
        sqlalchemy.Column('weight', sqltypes.Float),
        sqlalchemy.Column('temperature', sqltypes.Float),
        sqlalchemy.Column('color', sqltypes.String(32)),
        )
    table.create(checkfirst=True)
    for measurement in measurements:
        i = table.insert()
        i.execute(**measurement)


```

Simple is better than complex

```python
def store(measurements):
    import json
    with open('measurements.json', 'w') as f:
        f.write(json.dumps(measurements))
```

```
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!
```

Make your own modules

Live demo

15 minutes break

Generators

Live demo

Example with no loop

Fibonacci sequences

Prime numbers sequences

File example

Imitating python objects

Live demo

```
__len__

__iter__

__str__

__repr__

__getitem__
```

Regular expressions

```
function(string_to_search, pattern)
```

```
"Cats are smarter than dogs"
```

```
"dogs"

"do.+"

"d[1–9]"

"d[^o]"
```

Regular expressions

```
import re
```

```
re.match()
re.search()
re.findall()
```

Regular expressions

```python
import re

line = "Cats are smarter than dogs."

match_obj = re.match('dogs', line)
if match_obj:
    print "match_obj.group() :", match_obj.group()
else:
    print "No match!!"

search_obj = re.search('dogs', line)
if search_obj:
    print "search_obj.group() :", search_obj.group()
else:
    print "No match!!"


# No match!!
# dogs
```

Regular expressions

```python
import re

line = "Cats are smarter than dogs"

match_obj = re.match('dogs', line)
if match_obj:
    print "match_obj.group() :", match_obj.group()
else:
    print "No match!!"

search_obj = re.search('dogs', line)
if search_obj:
    print "search_obj.group() :", search_obj.group()
else:
    print "No match!!"


# No match
# dogs
```

Regular expressions

```python
import re

line = "Cats are smarter than dogs"

list_result = re.findall('dogs', line)

# ['dogs']
```

Regular expressions

Special characters

. ^ $ * + ? { } [ ] \ | ( )

Regular expressions

Live demo

15 minutes break

Exercise: FASTA parser

https://github.com/xapple/python_ebc_2016/tree/master/day_05/exercise/

15 minutes break

Clicker quizz

Suggest subjects for day 6

tinyurl.com/zk38vlm