Welcome back

Python course 2016 - Week 2 - Day 6

Lucas Sinclair
Alexander Eiler
Ludovic Dutoit

Boolean order

True or False and True and False or True

Boolean order

True or False and True and False or True

False

False

True

True

Boolean order

The "and" goes first like the multiplication goes first.

The "or" goes second like the addition goes second.

Otherwise it's left to right.

Objects - quick summary

```
class Person(object):
    def __init__(self, last_name):
        self.last_name = last_name


lucas = Person("Sinclair")
```

```
1  class Person(object):
2      def __init__(self, last_name):
3          self.last_name = last_name
4
5
6  lucas = Person("Sinclair")
7  ludo = Person("Dutoit")
8  alex = Person("Eiler")
9
10
11
12
13
```

```
class Person(object):
    def __init__(self, last_name):
        self.last_name = last_name


lucas = Person("Sinclair")
```

```
class Person(object):
    def __init__(self, coconut):
        self.last_name = coconut


lucas = Person("Sinclair")
```

```
1  class Person(object):
2      def __init__(self, coconut):
3          self.last_name = coconut
4          self.kiwi = "Lots of vitamins"
5
6
7  lucas = Person("Sinclair")
8
9
10
11
12
13
```

```
class Person(object):
    def __init__(self, coconut):
        self.last_name = coconut
        self.kiwi = "Lots of vitamins"
        self.last_name = None


lucas = Person("Sinclair")
```

```
class Person(object):
    def __init__(self, coconut):
        self.last_name = coconut
        self.kiwi = "Lots of vitamins"
        self.last_name = None


lucas = Person("Sinclair")
lucas.pineapple = "Hard to cut open"


```

```
class Person(object):
    def __init__(self, coconut):
        self.last_name = coconut
        self.kiwi = "Lots of vitamins"
        self.last_name = None


lucas = Person("Sinclair")
lucas.pineapple = "Hard to cut open"
lucas.kiwi = "Originally small birds"


```

```
class Person(object):
    def __init__(self, coconut):
        self.last_name = coconut
        self.kiwi = "Lots of vitamins"
        self.last_name = None


lucas = Person("Sinclair")
lucas.pineapple = "Hard to cut open"
lucas.kiwi = "Originally small birds"
del lucas.pineapple

```

```
class Person(object):
    def __init__(self, coconut):
        self.last_name = coconut
        self.kiwi = "Lots of vitamins"
        self.last_name = None
        def add(x, y):
            return x+y
        self.salary = add(10000+20000)

lucas = Person("Sinclair")
lucas.pineapple = "Hard to cut open"
lucas.kiwi = "Originally small birds"

```

```
class Person(object):
    def __init__(self, coconut):
        self.last_name = coconut
        self.kiwi = "Lots of vitamins"
        self.last_name = None
        self.salary = self.add(10000+20000)

    def add(self, x, y):
        return x+y

lucas = Person("Sinclair")
print lucas.salary

```

The concept

1. An object packs the data and the logic that operates on the data in one coherent unit.
2. Object attributes -> data.
3. Object logic -> methods.
4. Self-contained and self-sufficient.

```
pH  temp    methane carbon  conductivity   date
7.4 21  90  0.9 20160820
7.2 23  91  0.5 20160819
7.9 23  76  0.6 20160818
8.4 22  87  0.7 20160821
9.5 25  92  0.8 20160820
```

```
measures = [line.strip().split() for line in open("data.tsv")]

def carbon_prediction(measure):
    quotient = measure[1] * measure[3]
    impact = measure[2] - measure[0]
    prediction = (quotient / 6.5) * (impact+1)
    return prediction

measures_early = [m for m in measures if m[-1] < 20160820]
measures_late  = [m for m in measures if m[-1] >= 20160820]

predictions_early = [carbon_prediction(m) for m in measures_early]
predictions_late  = [carbon_prediction(m) for m in measures_late]
```

```
class Measure(object):
     def __init__(self, raw_line):
          measures = line.strip().split()
          self.ph = measures[0]
          self.temp = measures[1]
          self.carbon = measures[2]
          self.cond = measures[3]
          self.date = measures[4]

    def predict_carbon(self):
        quotient = self.temp * self.cond
        impact = self.ph - self.carbon
        return (quotient / 6.5) * (impact+1)

measures = [Measure(line) for line in open("data.tsv")]

measures_early = [m for m in measures if m.date < 20160820]
measures_late  = [m for m in measures if m.date >= 20160820]

predictions_early = [m.predict_carbon() for m in measures_early]
predictions_late  = [m.predict_carbon() for m in measures_late]

```

sys.argv

Alex

How to work clean

Ludo

Where is the bottle neck

Ludo

15 minutes break

Test driven development

```
def add(x,y):
    pass

assert add(0,0) == 0
assert add(1,1) == 2
assert add(2,2) == 4

with pytest.raises(ValueError):
    add("hi", "hello")

with pytest.raises(ValueError):
    add([], [])

```

```
1  def add(x,y):
2      pass
3
4  assert
5
6
7
8
9
10
11
12
13
14
```

Clickers

End

Upload your solutions before Monday to your github repo.

Maybe, see you next week for the advanced part.

15 minutes break