Object Calesthenics

Recap / Refactor / Reflections

michal.smilowski@css.ch - CSS Insurance

Only One Level Of Indentation Per Method

```python
def mutate(self):
    for i in range(0, self.size):
        if np.random.rand() < self.mutation_probability:
            self.weights[i] = np.random.rand()
```

```python
def mutate(self):
    for i in range(0, self.size):
        self._performMutation(i)

def _performMutation(self, i):
    if self._shouldMutate():
        self.weights[i] = np.random.rand()

def _shouldMutate(self):
    return np.random.rand() < self.mutation_probability
```

Don’t Use The ELSE Keyword

```python
def get_best_solutions_json():
    table_name = config["DB_COLLECTIONS"]["engine_weights"]
    solutions = []
    cursor = db.read_collection(table_name)
    if cursor.count() > 0:
        solution = cursor.next()
        while solution:
            solutions.append(solution)
            if cursor.alive:
                solution = cursor.next()
            else:
                solution = None
    return solutions
```

```python
def get_best_solutions_json():
    table_name = config["DB_COLLECTIONS"]["engine_weights"]
    cursor = db.read_collection(table_name)
    if cursor.count() == 0:
        return solutions
    return _extract_solutions(cursor)

def _extract_solutions(self, cursor):
    solutions = []
    solution = cursor.next()
    while solution:
        solutions.append(solution)
        solution = _process_cursor(cursor)
    return solutions

def _process_cursor(cursor):
    if cursor.alive:
        return cursor.next()
    return None
```

Wrap All Primitives And Strings

```
class DNA(object):
    def __init__(self, size):
        self.mutation_probability = .05
```

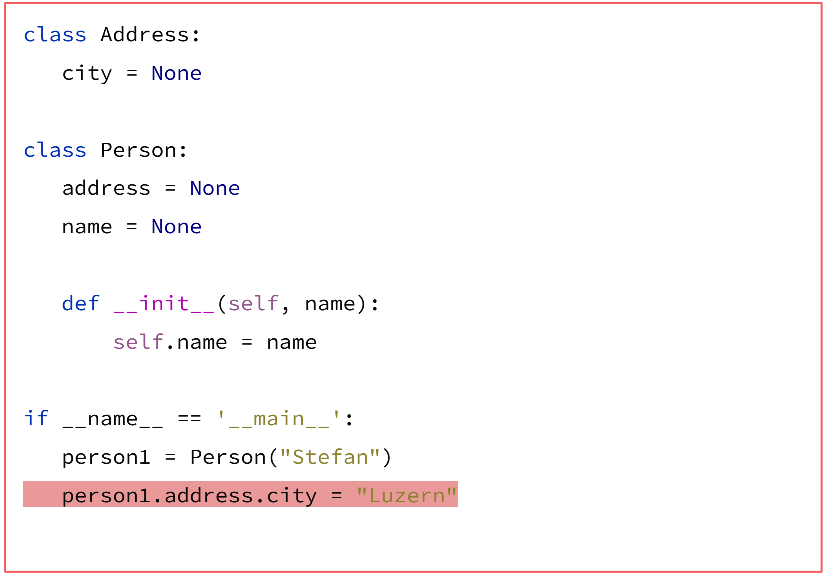
```
class DNA(object):
    def __init__(self, size):
        self.mutation = Mutation(5)

class Mutation(object):
    def __init__(self, probability):
        self.probability(probability)

    @property
    def probability(self):
        return self.probability

    @probability.setter
    def probability(self, probability):
        if not 0 < probability < 100:
            raise ValueError("Probability value not valid")
        self.probability = probability/100
```

First Class Collections

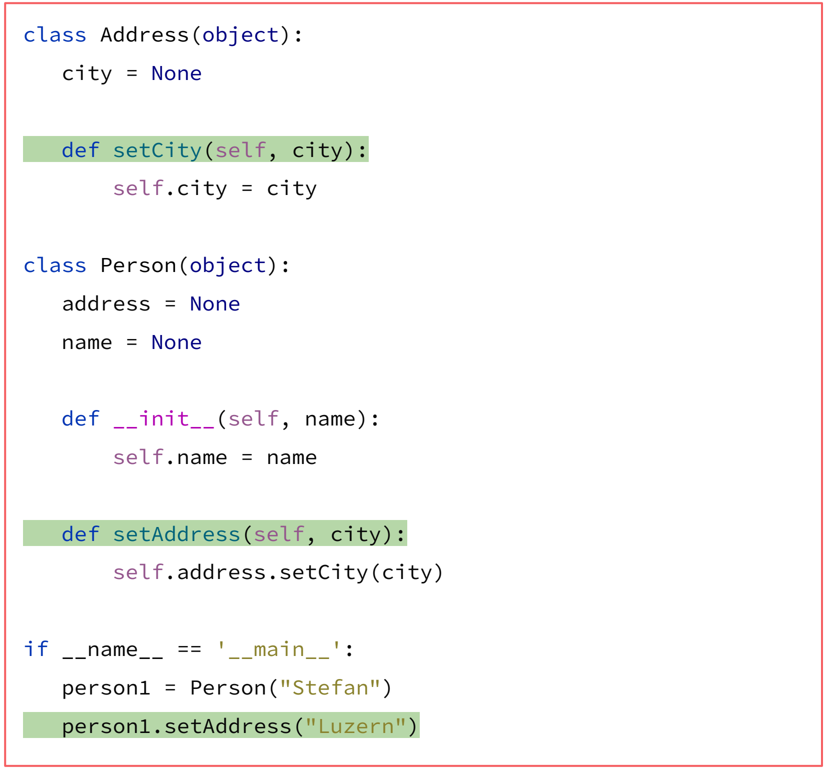
```
class Population:
  def __init__(self, size, aggregation_period):
     self.solutions_list = OrderedDict()
…

self.solutions_list[i] = copy.deepcopy(seed)
…

solutions_list = OrderedDict(sorted(
               self.solutions_list.items(),
               key=lambda x: x[1].fitness
          ))
…

self.solutions_list = OrderedDict()
```

```
class Population:
  def __init__(self, size, aggregation_period):
     self.solutions_list = Solutions()

class Solutions:
  def __init__(self):
     self.solutions = OrderedDict()

  def updateSolution(self, index, newSolution):
     self.solutions[index] = copy.deepcopy(newSolution)

  def sortByFitness(self):
     self.solutions = OrderedDict(sorted(
                    self.solutions_list.items(),
                    key=lambda x: x[1].fitness
               ))

  def resetSolutions(self):
     self.solutions = OrderedDict()
```

One Dot Per Line



```
class Address:
   city = None

class Person:
   address = None
   name = None

   def __init__(self, name):
       self.name = name

if __name__ == '__main__':
   person1 = Person("Stefan")
   person1.address.city = "Luzern"
```



```
class Address(object):
   city = None

   def setCity(self, city):
       self.city = city

class Person(object):
   address = None
   name = None

   def __init__(self, name):
       self.name = name

   def setAddress(self, city):
       self.address.setCity(city)

if __name__ == '__main__':
   person1 = Person("Stefan")
   person1.setAddress("Luzern")
```

Don’t Abbreviate

```
class SomeFunnyObject(object):
    def __init__(self):
        self.cpc = 1.0
        self.userList = UserList

class UserList(object):
    userList = []

    def addUser(self, user):
        self.userList.append(user)

    def removeUser(self, user):
        self.userList.remove(user)
ç
```

```
class SomeFunnyObject(object):
    def __init__(self):
        self.costPerClick = 1.0
        self.userList = UserList

class UserList(object):
    userList = []

    def add(self, user):
        self.userList.append(user)

    def remove(self, user):
        self.userList.remove(user)
```

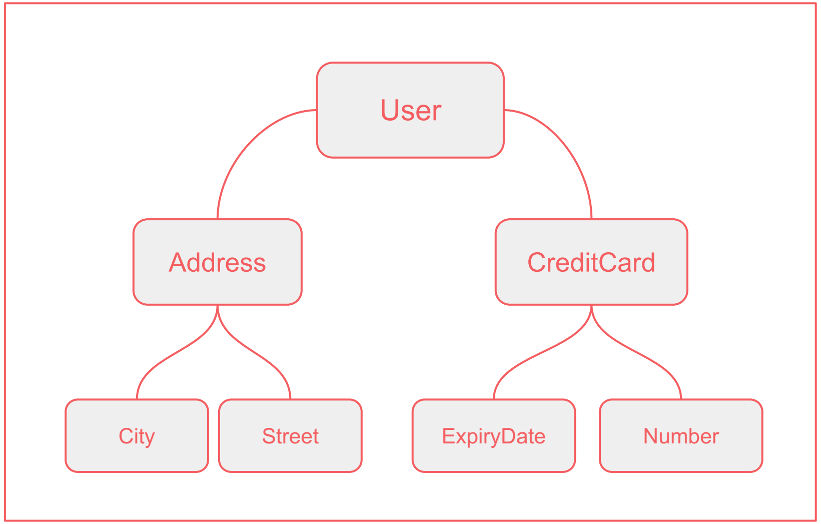
Keep All Entities Small

`No class over 50 lines and no package over 10 files`

The more logic you have the less understandable it is

Take it with a grain of salt

No Classes With More Than Two Instance Variables



- Is it doable on big projects?
- Why 2 not 3?
- Do these count?
    - Logger
    - Database connection
    - MetricHandler
- Think about cohesion and coupling

No Getters/Setters/Properties

```
class Wallet(object):
    def __init__(self):
        self.coins = 0
        self.money = 10

if __name__ == '__main__':
    wallet = Wallet()
    exchengeRate = 0.5
    wallet.coins += 10
    wallet.money -= 10 * exchangeRate
```

```
class Wallet(object):
    def __init__(self):
        self.coins = 0
        self.money = 10

    def buyCoins(self, coins, exchengeRate):
        self.coins += coins
        self.money -= coins * exchengeRate

if __name__ == '__main__':
    wallet = Wallet()
    wallet.buyCoins(10, 0.5)
```

Conclusion

Use common sense