ETH zürich

D BSSE

Introduction to programming in Python

Bowen Fan (slides created by Bastian Rieck)

28 September 2022

Machine Learning and Computational Biology Group

The Zen of Python

Open the python interpreter and enter the following:

```
>>> import this
```

The 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, its 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!
```

The Zen of Python - compact

```
>>> import this
...
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.
...
If the implementation is hard to explain, its a bad idea.
If the implementation is easy to explain, it may be a good
    idea.
...
```

What is clean code? - baking a cake

```
from kitchen import *

from ingredients import *


def bake_a_cake():
    oven.set_temperature(190)
    b = Bowl()
    f = BakingForm()
    fpack = cupboard.take("flour")

    if fpack.is_closed:

      fpack.open()
    b.add(fpack)
    spack = cupboard.take('sugar')

    if spack.is_closed: spack.open()
    b.add(spack)
    e = f.take('eggs')
    einside = e.crack_open()

    b.add(einside)
    b.mix()
    f.fill(b)
```

What is clean code? - baking a cake

```
from kitchen import *

from ingredients import *


def bake_a_cake():
    oven.set_temperature(190)
    b = Bowl()
    f = BakingForm()
    fpack = cupboard.take("flour")

    if fpack.is_closed:

      fpack.open()
    b.add(fpack)
    spack = cupboard.take('sugar')

    if spack.is_closed: spack.open()
    b.add(spack)
    e = f.take('eggs')
    einside = e.crack_open()

    b.add(einside)
    b.mix()
    f.fill(b)
```

Explicit is better than implicit

Beautiful is better than ugly

Readability counts

What is clean code? - baking a cake

```
from kitchen import *
from ingredients import *


def bake_a_cake():
    oven.set_temperature(190)
    b = Bowl()
    f = BakingForm()
    fpack = cupboard.take("flour")
    if fpack.is_closed:
      fpack.open()
    b.add(fpack)
    spack = cupboard.take('sugar')
    if spack.is_closed: spack.open()
    b.add(spack)
    e = f.take('eggs')
    einside = e.crack_open()
    b.add(einside)
    b.mix()
    f.fill(b)
```

Explicit is better than implicit

Beautiful is better than ugly

Readability counts

What is clean code? - baking a cake

```
from kitchen import BakingForm, Bowl, cupboard, fridge, oven
from ingredients import Egg, Package

def add_ingredient_to_bowl(ingredient_name, bowl):
    try:
        ingredient = cupboard.take(ingredient_name)
    except IngredientNotFound:
        # Ingredients not in the cupboard are in the fridge
        ingredient = fridge.take(ingredient_name)

    if isinstance(ingredient, Package) and ingredient.is_closed:
        ingredient.open()
    elif isinstance(ingredient, Egg):
        ingredient = ingredient.crack_open()
    bowl.add(ingredient)

def bake_a_cake():
    oven.set_temperature(190)
    bowl = Bowl()
    form = BakingForm()
    add_ingredient_to_bowl('flour', bowl)
    add_ingredient_to_bowl('sugar', bowl)
    add_ingredient_to_bowl('eggs', bowl)
    bowl.mix()
    form.fill(bowl)
    # Bake cake for one hour
    cake = oven.bake(form, 60*60)
    return cake
```

Why clean code?

```
from kitchen import BakingForm, Bowl, cupboard, fridge, oven
from ingredients import add_ingredient_to_bowl, Egg, Package

def bake_a_quiche():
    oven.set_temperature(180)
    bowl = Bowl()
    form = BakingForm()
    add_ingredient_to_bowl('flour', bowl)
    add_ingredient_to_bowl('eggs', bowl)
    add_ingredient_to_bowl('salt', bowl)
    bowl.mix()
    form.fill(bowl)
    # Bake quiche for one hour, 30 minutes
    quiche = oven.bake(form, 90*60)
    return quiche
```

- It is easier for others to understand
- It is more maintainable
- It usually allows more flexible usage

Code can smell!

Code smells

- Long methods
- Large classes
- Long parameter lists
- Way too many comments
- Duplicate code
- Dead code

Code smells

- Long methods
- Large classes
- Long parameter lists
- Way too many comments
- Duplicate code
- Dead code

Often we encounter these smells, understand that something is probably wrong, but do not know how to fix them.

Code smells

- Long methods
- Large classes
- Long parameter lists
- Way too many comments
- Duplicate code
- Dead code

Often we encounter these smells, understand that something is probably wrong, but do not know how to fix them.

There are useful resources, such as `https://stackoverflow.com/`

Practice makes perfect!

Practice makes perfect

You learn best by *using* a programming language. Python has a great 'ecosystem' and can be a valuable skill to learn, going above and beyond this particular course!