Lecture 3: Software development I

CEE 690

Moving (slowly) away from Spaghetti code

Fortran code example ~1973

```
                      DIMENSION A(5,5)
000003                MN=5
000004                PRINT81
000010                READ 91,A
000016                IA=MN-1
000020                D=1
000022                DO1 IM=1,IA
000023                IF(A(IM,IM).NE.0)GOTO4
000026              5 J=IM+1
000030                IF(A(IM,J).NE.0)  GOTO6
000034                IF(J.LE.MN) GOTO8
000036                PRINT102
000042                GOTO2101
000043              6 DO9 JJ=IM,MN
000045                A(JJ,IM)=A(JJ,J)+A(JJ,IM)
000054              9 CONTINUE
000056              4 K=0
000057                IB=IM+1
000061                DO3 IN=IB,MN
000062                K=K+1
000064                I=IN-K
000065                DIV=A(IM,IN)/A(IM,IM)
000072                DO3 LM=IM,MN
000073                A(LM,IN)=A(LM,IN)-A(LM,I)*DIV
000103              3 CONTINUE
000107                D=A(IM,IM)*D
000112              1 CONTINUE
000114                D=A(MN,MN)*D
000117                PRINT 101,D
000125           2101 PRINT 707
000131                STOP
000133             81 FORMAT(1H1,//1X,*CALCOLODEI DETERMINANTI PER SVILUPPI SUCESSIVI*)
000133             91 FORMAT(25F5.2)
000133            102 FORMAT(1X,*DETERMINANTE NULLO*)
000133            101 FORMAT(1X,F8.3)
000133            707 FORMAT(1H ,1X,*1HBIANCO*,1H0,1X,*,1H+,1X,*1H+*,///,1X,2X,*2X*,3X,*
                     13X*,1H=,*FINE*)
000133                END
```

“Spaghetti code”

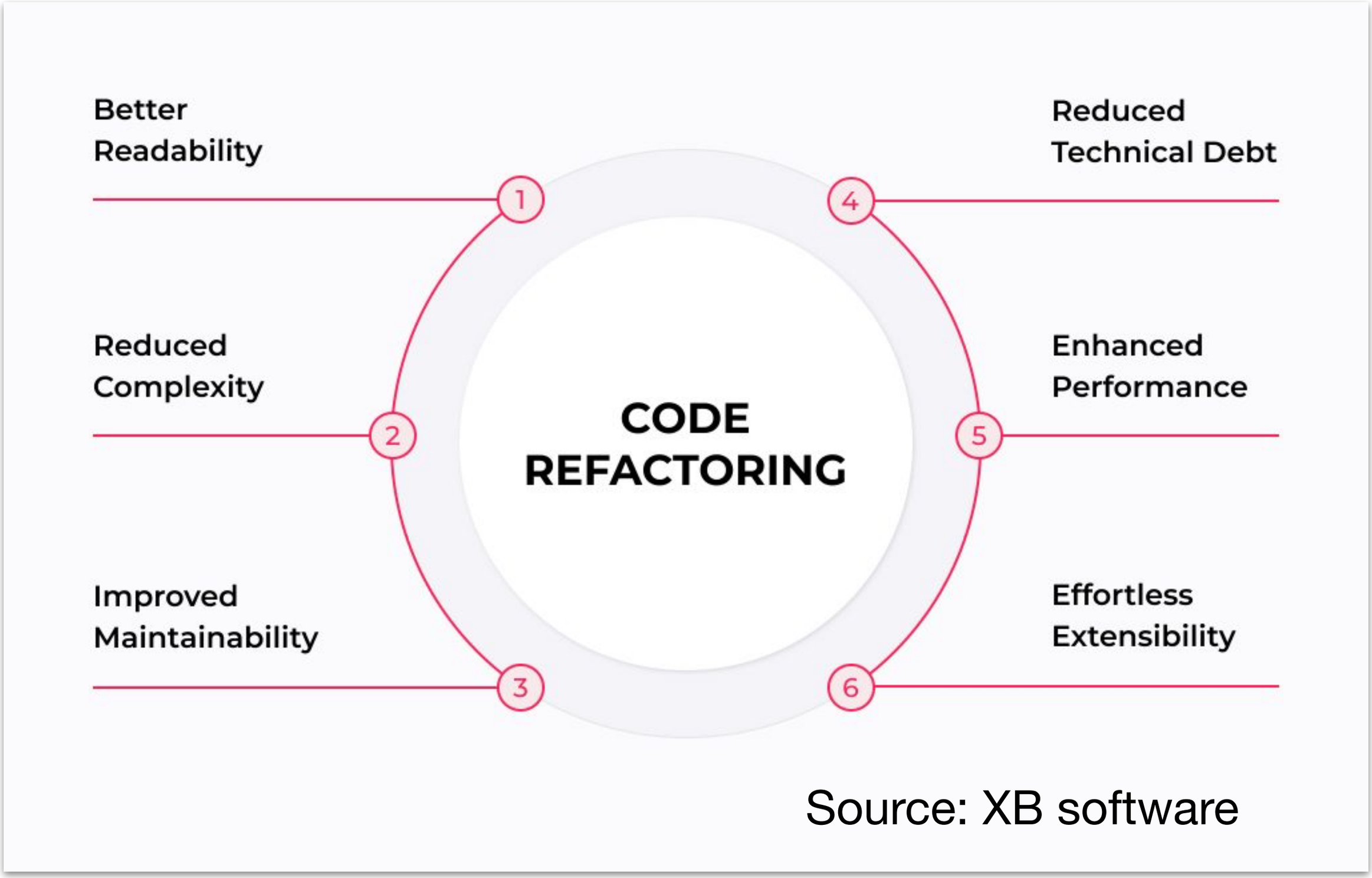
```
import netCDF4 as nc;import numpy as np;import matplotlib.pyplot as plt
d=nc.Dataset('era_interim_monthly_197901_201512_upscaled_annual.nc','r')
r=d.variables['t2m'][:]
a=5;b=50;c=10;d=100;e=0;f=10
A,B=[],[]
for t in range(len(r)):
    if ((t < e) | (t >= f)):continue
    count = 0
    val = 0
    for y in range(len(r[0])):
        lr=[]
        if ((y<a) | (y>=b)):continue
        for x in range(len(r[0][0])):
            if ((x<c) | (x>=d)):continue
            count = count + 1
            val = val + r[t][y][x]
    A.append(val/count)
for t in range(len(r)):
 if ((t < e) | (t >= f)):continue
 count = 0
 val = 0
 for y in range(len(r[0])):
    lr=[]
    if ((y<a) | (y>=b)):continue
    for x in range(len(r[0][0])):
            if ((x<c) | (x>=d)):continue
            count = count + 1
            val = val + (r[t][y][x] - A[t])**2
 B.append(val/count)
A=np.array(A)
B=np.array(B)
plt.plot(A);plt.plot(B)
plt.show()
o=nc.Dataset('out.nc','w')
o.createDimension('t',10)
v=o.createVariable('B','f4',('t',));v[:]=A
v=o.createVariable('A','f4',('t',));v[:]=B
o.close()
```

“Spaghetti code”

```
import netCDF4 as nc;import numpy as np;import matplotlib.pyplot as plt
d=nc.Dataset('era_interim_monthly_197901_201512_upscaled_annual.nc','r')
r=d.variables['t2m'][:]
a=5;b=50;c=10;d=100;e=0;f=10
A,B=[],[]
for t in range(len(r)):
    if ((t < e) | (t >= f)):continue
    count = 0
    val = 0
    for y in range(len(r[0])):
        lr=[]
        if ((y<a) | (y>=b)):continue
    for x in range(len(r[0][0])):
            if ((x<c) | (x>=d)):continue
            count = count + 1
            val = val + (r[t][y][x] - A[t])**2
 B.append(val/count)
A=np.array(A)
B=np.array(B)
plt.plot(A);plt.plot(B)
plt.show()
o=nc.Dataset('out.nc','w')
o.createDimension('t',10)
v=o.createVariable('B','f4',('t',));v[:]=A
v=o.createVariable('A','f4',('t',));v[:]=B
o.close()
```

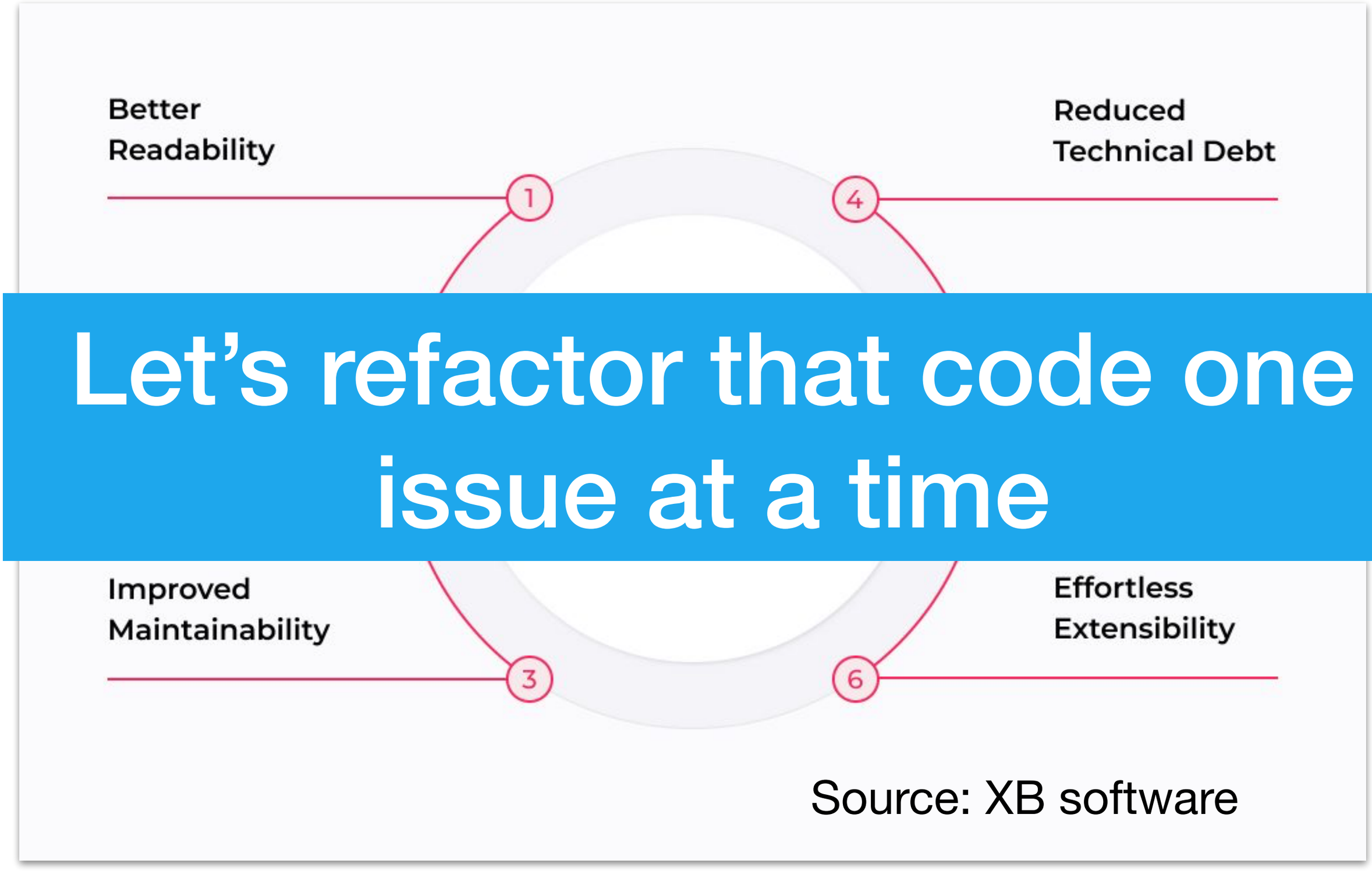
This is far from contemporary coding practices. Let’s learn what is “correct” by refactoring this code

Code refactoring



Source: XB software

Code refactoring



Let’s refactor that code one issue at a time

Source: XB software

But first… Style guide

PEP 8 – Style Guide for Python Code

Author: Guido van Rossum <guido at python.org>, Barry Warsaw <barry at python.org>, Alyssa Coghlan <ncoghlan at gmail.com>

Status: Active

Type: Process

Created: 05-Jul-2001

Post-History: 05-Jul-2001, 01-Aug-2013

1. Indentation

Use 4 spaces per indentation (not tabs)

```python
def greet(name):
    if name:
        print(f"Hello, {name}!")
    else:
        print("Hello, Guest!")
```

```python
def greet(name):
  if name: # Only 2 spaces (Inappropriate)
        print(f"Hello, {name}!") # Tab used instead of spaces
```

2.1 Naming conventions: Variables and functions

```python
user_name = "Alice"

def calculate_total_price(item_price, tax_rate):
    pass
```

```python
# Avoid CamelCase for functions and variables
userName = "Alice"

def CalculateTotalPrice(itemPrice):
    pass
```

2.2 Naming conventions: Classes

```
class UserProfile:
    pass

class SmartAccountManager:
    pass
```

```
class user_profile: # Should not be snake_case
    pass

class smartAccountManager: # Should not be mixedCase
    pass
```

2.3 Naming conventions: Constants

```
MAX_RETRIES = 5
API_ENDPOINT = "https://api.example.com/v1"
```

```
max_retries = 5 # Looks like a regular variable
apiEndpoint = "..." # Incorrect style
```

3.1 Comments: Block comments

```
# Calculate the adjusted score by applying the weighted
# multiplier and subtracting the penalty points.
adjusted_score = (raw_score * multiplier) - penalty
```

```
#Calculate the adjusted score (Missing space after #)
#-------------------------- (Avoid decorative borders)
adjusted_score = (raw_score * multiplier) - penalty
```

3.2 Comments: In-line comments

Away by at least two spaces

```
x = x + 1  # Increment the boundary counter
```

```
x = x + 1 # Too close to the code
x = x + 1          # Way too far away from the code
```

3.3 Comments: Docstrings

Used for all public modules, methods, function, and classes

```python
def fetch_user_data(user_id):
    """
    Retrieve user profile and permissions from the database.

    Args:
        user_id (int): The unique identifier for the user.
    """
    pass
```

```python
def fetch_user_data(user_id):
    '''Uses single quotes (not recommended).'''

    """
    This docstring is missing a summary line or has
    the closing quotes incorrectly placed."""
```

4. Max line length

Not longer than 79 characters*

```
# Wrapped using parentheses and aligned indentation
result = some_function_with_a_very_long_name(
    argument_one, argument_two, argument_three,
    argument_four, argument_five
)
```

```
# This single line is 112 characters long, forcing horizontal scrolling
result = some_function_with_a_very_long_name(argument_one, argument_two, argumen
```

5. Blank lines

Top-level functions and class definitions - Two blank lines

```python
import os


def first_function():
    return "Hello"


def second_function():
    return "World"


class MyClass:
    pass
```

```python
import os
def first_function():
    return "Hello"
def second_function():
    return "World"
```

7. Import order

- Top of file after comments and docstrings and before variable declaration
- Order (blank line between each one): 1) Standard library, 2) third party libraries, 3) local imports

```python
import os
import sys

import requests
from flask import Flask

from my_project.utils import helper_function
```

```python
import requests
from my_project.utils import helper_function
import os # Standard library should be at the top
```

8. Whitespaces

Keep it simple

```
# No space inside the parentheses/brackets
spam(ham[1], {eggs: 2})
```

```
if x == 4: print(x, y); x, y = y, x
```

```
# Consistent spacing (none in this case)
ham[1:9], ham[1:9:3], ham[:9], ham[1:], ham[1:9:]
```

```
# Unnecessary gaps make the expression look fragmented
spam( ham[ 1 ], { eggs: 2 } )
```

```
# Spaces before punctuation are distracting
if x == 4 : print(x , y) ; x , y = y , x
```

```
# Inconsistent spacing
ham[1: 9]
ham[1 :9]
```

8. Whitespaces

Keep it simple

```
# No space inside the parentheses/brackets
spam(ham[1], {eggs: 2})
```

And many more coding conventions from PEP-8

```
spam( ham[ 1 ], { eggs: 2 } )
```

```
# Spaces before punctuation are distracting
if x == 4 : print(x , y) ; x , y = y , x
```

```
# Inconsistent spacing
ham[1: 9]
ham[1 :9]
```

Do I really need to use PEP-8?

Do I really need to use PEP-8?

No, but it makes your code more interpretable and portable

Do I really need to use PEP-8?

No, but it makes your code more interpretable and portable

Whatever you end up doing, make sure to be consistent

Begin de-spaghettifying

```
 1 import netCDF4 as nc
 2 import numpy as np
 3 import matplotlib.pyplot as plt
 4
 5 # Load dataset
 6 d = nc.Dataset('era_interim_monthly_197901_201512_upscaled_annual.nc','r')
 7 r = d.variables['t2m'][:]
 8
 9 # Configuration variables
10 a = 5
11 b = 50
12 c = 10
13 d = 100
14 e = 0
15 f = 10
16
17 A, B = [], []
18
19 # Calculate temporally varying spatial mean
20 for t in range(len(r)):
21     if ((t < e) | (t >= f)):
22         continue
23
24     count = 0
25     val = 0
26
27     for y in range(len(r[0])):
28         if ((y < a) | (y >= b)):
29             continue
30
31         for x in range(len(r[0][0])):
32             if ((x < c) | (x >= d)):
33                 continue
34
```

Cryptic variables

```python
def calculate_area(width, height):
    return width * height

# Usage
rectangle_area = calculate_area(width=10, height=5)
```

```python
# What are 'a' and 'b'? Units? Names?
def calculate(a, b):
    return a * b

# Usage
val = calculate(10, 5)
```

De-spaghettify some more

```
import netCDF4 as nc
import numpy as np
import matplotlib.pyplot as plt

# Load dataset
file_pointer_input = nc.Dataset('era_interim_monthly_197901_201512_upscaled_annual.nc','r')
t2m_data = file_pointer_input.variables['t2m'][:]

# Configuration variables
LAT_MIN = 5
LAT_MAX = 50
LON_MIN = 10
LON_MAX = 100
TIME_MIN = 0
TIME_MAX = 10

temporal_spatial_mean, temporal_spatial_variance = [], []

# Calculate temporally varying spatial mean
for t in range(len(t2m_data)):
    if ((t < TIME_MIN) | (t >= TIME_MAX)):
        continue

    pixel_count = 0
    total_t2m = 0

    for y in range(len(t2m_data[0])):
        if ((y < LAT_MIN) | (y >= LAT_MAX)):
            continue

        for x in range(len(t2m_data[0][0])):
            if ((x < LON_MIN) | (x >= LON_MAX)):
                continue
```

Hardcoded variables

```python
SALES_TAX_RATE = 0.08

def calculate_total(price):
    return price + (price * SALES_TAX_RATE)
```

```python
def calculate_total(price):
    # Hardcoded tax rate
    return price + (price * 0.08)
```

And more de-spaghettifying

```
 1 import netCDF4 as nc
 2 import numpy as np
 3 import matplotlib.pyplot as plt
 4
 5 # Configuration variables
 6 INPUT_FILE = 'era_interim_monthly_197901_201512_upscaled_annual.nc'
 7 OUTPUT_FILE = 'out.nc'
 8 VAR_NAME = 't2m'
 9 LAT_MIN = 5
10 LAT_MAX = 50
11 LON_MIN = 10
12 LON_MAX = 100
13 TIME_MIN = 0
14 TIME_MAX = 10
15
16 # Load dataset
17 file_pointer_input = nc.Dataset(INPUT_FILE,'r')
18 t2m_data = file_pointer_input.variables[VAR_NAME][:]
19
20 temporal_spatial_mean, temporal_spatial_variance = [], []
21
22 # Calculate temporally varying spatial mean
23 for t in range(len(t2m_data)):
24     if ((t < TIME_MIN) | (t >= TIME_MAX)):
25         continue
26
27     pixel_count = 0
28     total_t2m = 0
29
30     for y in range(len(t2m_data[0])):
31         if ((y < LAT_MIN) | (y >= LAT_MAX)):
32             continue
```

Ensuring headless environment compatibility

```
72 #Visualize the data
73 plt.plot(temporal_spatial_mean, label="Mean")
74 plt.plot(temporal_spatial_variance, label="Variance")
75 plt.legend()
76 plt.savefig(PLOT_FILE)  # Saves directly to disk
77 plt.close()
```

```
69 #Visualize the data
70 plt.plot(temporal_spatial_mean)
71 plt.plot(temporal_spatial_variance)
72 plt.show()
73
```

Options to view images including downloading (scp), jupyterlab, X-window…

Moving to functions

```python
import netCDF4 as nc
import numpy as np
import matplotlib
matplotlib.use('Agg') # Force Matplotlib to not use any X-Windows backend
import matplotlib.pyplot as plt

# Configuration variables
INPUT_FILE = 'era
OUTPUT_FILE = 'ou
PLOT_FILE = 'plot
VAR_NAME = 't2m'
LAT_MIN = 5
LAT_MAX = 50
LON_MIN = 10
LON_MAX = 100
TIME_MIN = 0
TIME_MAX = 10

def calculate_spa

    # Define the
    temporal_spat

    # Calculate t
    for t in rang
        if ((t <
            conti

        pixel_cou
        total_dat

        for y in
            if ((
                c
```

```python

    return

def output_data_to_netcdf(output_file,temporal_spatial_mean,temporal_spatial_variance):

    # Output the data to a netcdf file
    file_pointer_output = nc.Dataset('out.nc','w')
    file_pointer_output.createDimension('t',TIME_MAX-TIME_MIN)

    var_v1 = file_pointer_output.createVariable('temporal_spatial_mean','f4',('t',))
    var_v1[:] = temporal_spatial_mean

    var_v2 = file_pointer_output.createVariable('temporal_spatial_variance','f4',('t',))
    var_v2[:] = temporal_spatial_variance

    file_pointer_output.close()

    return

# Load dataset
t2m_data = load_dataset(INPUT_FILE,VAR_NAME)

# Compute temporal series of spatial mean and spatial standard deviation
temporal_spatial_mean = calculate_spatial_mean(t2m_data,TIME_MIN,TIME_MAX,
                                               LAT_MIN,LAT_MAX,LON_MIN,LON_MAX)
temporal_spatial_variance = calculate_spatial_variance(t2m_data,TIME_MIN,TIME_MAX,
                                                       LAT_MIN,LAT_MAX,LON_MIN,LON_MAX,
                                                       temporal_spatial_mean)


#Visualize the data
visualize_data(temporal_spatial_mean,temporal_spatial_variance,PLOT_FILE)
```

Function portability

```
93
94      return
95
96  def output_data_to_netcdf(output_file,temporal_spatial_mean,temporal_spatial_variance):
97
98      # Output the data to a netcdf file
99      file_pointer_output = nc.Dataset('out.nc','w')
100     file_pointer_output.createDimension('t',TIME_MAX-TIME_MIN)
101
102     var_v1 = file_pointer_output.createVariable('temporal_spatial_mean','f4',('t',))
103     var_v1[:] = temporal_spatial_mean
104
105     var_v2 = file_pointer_output.createVariable('temporal_spatial_variance','f4',('t',))
106     var_v2[:] = temporal_spatial_variance
107
108     file_pointer_output.close()
109
110     return
111
112 # Load dataset
113 t2m_data = load_dataset(INPUT_FILE,VAR_NAME)
114
115 # Compute temporal series of spatial mean and spatial standard deviation
116 temporal_spatial_mean = calculate_spatial_mean(t2m_data,TIME_MIN,TIME_MAX,
117                                                 LAT_MIN,LAT_MAX,LON_MIN,LON_MAX)
118 temporal_spatial_variance = calculate_spatial_variance(t2m_data,TIME_MIN,TIME_MAX,
119                                                         LAT_MIN,LAT_MAX,LON_MIN,LON_MAX,
120                                                         temporal_spatial_mean)
121
122 #Visualize the data
123 visualize_data(temporal_spatial_mean,temporal_spatial_variance,PLOT_FILE)
```

I want to use the functions in this python script for something else… But there is a problem.

Function portability

```
93
94      return
95
96  def output_data_to_netcdf(output_file,temporal_spatial_mean,temporal_spatial_variance):
97
98      # Output the data to a netcdf file
99      file_pointer_output = nc.Dataset('out.nc','w')
100     file_pointer_output.createDimension('t',TIME_MAX-TIME_MIN)
101
102     var_v1 = file_pointer_output.createVariable('temporal_spatial_mean','f4',('t',))
103     var_v1[:] = temporal_spatial_mean
104
105     var_v2 = file_pointer_output.createVariable('temporal_spatial_variance','f4',('t',))
106     var_v2[:] = temporal_spatial_variance
107
108     file_pointer_output.close()
109
110     return
111
112 # Load dataset
113 t2m_data = load_dataset(INPUT_FILE,VAR_NAME)
114
115 # Compute temporal series of spatial mean and spatial standard deviation
116 temporal_spatial_mean = calculate_spatial_mean(t2m_data,TIME_MIN,TIME_MAX,
117                                                LAT_MIN,LAT_MAX,LON_MIN,LON_MAX)
118 temporal_spatial_variance = calculate_spatial_variance(t2m_data,TIME_MIN,TIME_MAX,
119                                                        LAT_MIN,LAT_MAX,LON_MIN,LON_MAX,
120                                                        temporal_spatial_mean)
121
122 #Visualize the data
123 visualize_data(temporal_spatial_mean,temporal_spatial_variance,PLOT_FILE)
```

Function portability

```
93
94      return
95
96  def output_data_to_netcdf(output_file,temporal_spatial_mean,temporal_spatial_variance):
97
98      # Output the data to a netcdf file
99      file_pointer_output = nc.Dataset('out.nc','w')
100     file_pointer_output.createDimension('t',TIME_MAX-TIME_MIN)
101
102     var_v1 = file_pointer_output.createVariable('temporal_spatial_mean','f4',('t',))
103     var_v1[:] = temporal_spatial_mean
104
105     var_v2 = file_pointer_output.createVariable('temporal_spatial_variance','f4',('t',))
106     var_v2[:] = temporal_spatial_variance
107
108     file_pointer_output.close()
109
110     return
111
112 # Load dataset
113 t2m_data = load_dataset(INPUT_FILE,VAR_NAME)
114
115 # Compute temporal series of spatial mean and spatial standard deviation
116 temporal_spatial_mean = calculate_spatial_mean(t2m_data,TIME_MIN,TIME_MAX,
117                                                 LAT_MIN,LAT_MAX,LON_MIN,LON_MAX)
118 temporal_spatial_variance = calculate_spatial_variance(t2m_data,TIME_MIN,TIME_MAX,
119                                                         LAT_MIN,LAT_MAX,LON_MIN,LON_MAX,
120                                                         temporal_spatial_mean)
121
122 #Visualize the data
123 visualize_data(temporal_spatial_mean,temporal_spatial_variance,PLOT_FILE)
```

I want to use the functions in this python script for something else… But there is a problem.

Intro to “structural encapsulation”

```
 99     return
100
101 def main():
102
103     """
104     The director of the orchestra. When this function is called, it runs the defined
105     sequence of functions. However, it also ensures that other parts of the script can
106     be accessed without running this.
107     """
108
109     # Configuration variables
110     INPUT_FILE = 'era_interim_monthly_197901_201512_upscaled_annual.nc'
111     OUTPUT_FILE = 'out.nc'
112     PLOT_FILE = 'plot.png'
113     VAR_NAME = 't2m'
114     LAT_MIN = 5
115     LAT_MAX = 50
116     LON_MIN = 10
117     LON_MAX = 100
118     TIME_MIN = 0
119     TIME_MAX = 10
120
121     # Load dataset
122     print("Loading the dataset")
123     t2m_data = load_dataset(INPUT_FILE,VAR_NAME)
124
125     # Compute temporal series of spatial mean and spatial standard deviation
126     print("Computing the statistics")
127     temporal_spatial_mean = calculate_spatial_mean(t2m_data,TIME_MIN,TIME_MAX,
128                                                    LAT_MIN,LAT_MAX,LON_MIN,LON_MAX)
```

```
143 if __name__ == "__main__":
144
145     main()
```