Day 3

Richèl Bilderbeek

Table of contents

1 Big Picture

1.1 Breaks

Please take breaks: these are important for learning. Ideally, do something boring (1)!

1.2 Break schedule

- 10:00-10:15
- 11:00-11:15
- 12:00-13:00
- 14:00-14:15
- 15:00-15:15
- 16:00-16:15

1.3 Question

What is mastery in programming?

Write down in the HackMD!

1.4 My answer

To be able to express your ideas in correct and readable code in an elegant way.

2 Express your assumptions in code

2.1 Problem

```
numerator = # something
denominator = # something
# denominator is not zero
value = numerator / denominator
```

How to do better?

Write down in the HackMD!

2.2 Naive answer

```
numerator = 1.2
denominator = 3.4
# denominator is not zero
if denominator == 0.0:
    print("ERROR! QUIT!")
```

What do we express here?

2.3 `assert`

Express: 'I believe this to be true':

```
assert 1 + 1 == 2
```

This presentation will not compile when an `assert` fails!

2.4 `assert` in action

```
numerator = 1.2
denominator = 3.4
assert denominator != 0.0
value = numerator / denominator
```

Use assert extensively (2)(3)(4) (5)(6).

2.5 `assert` references 1/2

- (2) Chapter 68: ' Assert liberally to document internal assumptions and invariants'
- (3) (3rd edition) Advice 24.5.18: 'Explicitly express preconditions, postconditions, and other assertions as assertions'
- (4) Chapter 8.2: 'Use assertions to document and verify preconditions and postconditions'
- (4) Chapter 8.2: 'Use assertions for conditions that should never occur'.
- (5) Chapter 'assert()': 'Use assert freely'

2.6 `assert` references 2/2

- (6) Chapter 2.6: 'The use of assert statements can help to document the assumptions you make when implementing your code
- (7) (4th edition) page 884: '[13] Use static_assert() and assert() extensively'

2.7 Questions about `assert`?

Write down in the HackMD!

3 Express yourself in readable code

3.1 Problem

```
a = 3
b = 4
# Calculate the Euclidean distance
# using Pythagoras' theorem
c = ((a * a) + (b * b)) ** 0.5
```

How to express ourselves in code? How to do better?

Write down in the HackMD!

3.2 Solution

```
a = 3
b = 4
c = calc_euclidian_distance(a, b)
```

We use a function!

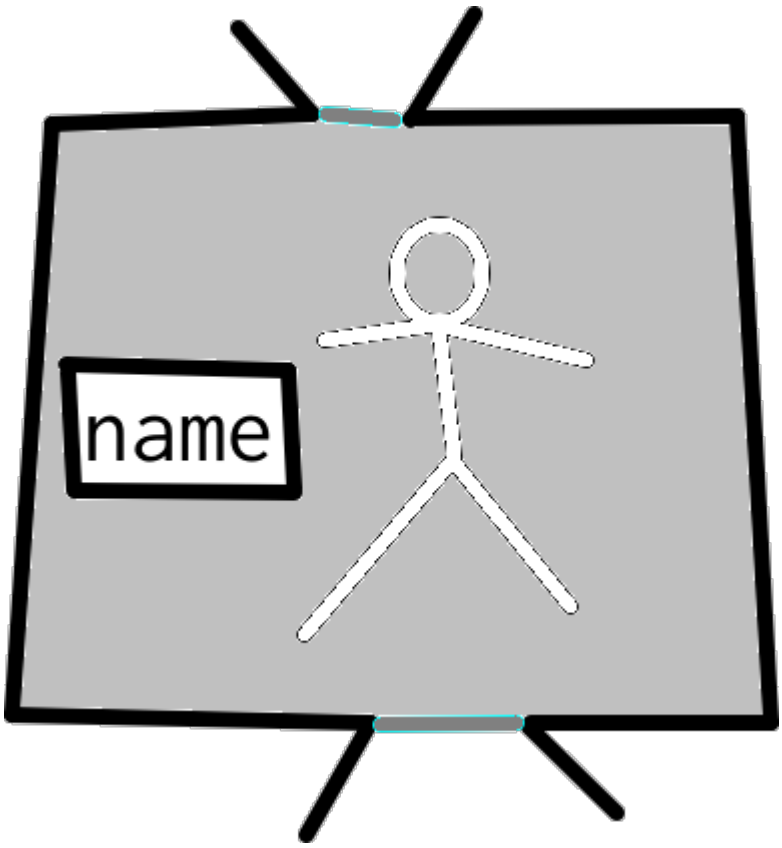
3.3 Question about functions

- What is a function?
- What makes a good function?

Write down in the HackMD!

3.4 My answer

A function is a unit of code, that has a name, optional inputs and optional output.



3.5 About good functions 1/3

- (8) Item 18: 'Make interfaces easy to use correctly and hard to use incorrectly.'
- (2) Item 20: 'Avoid long functions. Avoid deep nesting'
- (3) (4th edition) page 341: '[1] Package meaningful operations as carefully named functions'
- (3) (4th edition) page 341: '[2] A function should perform a single logical operation'
- (3) (4th edition) page 341: '[3] Keep functions short'

3.6 About good functions 2/3

- (6) Chapter 2.6: Document the interfaces so that they are usable by others. Have at least one other developer review each interface
- (9) AV Rule 1: 'Any one function (or method) will contain no more than 200 logical source lines of code.'

3.7 About good functions 3/3

- (3) (4th edition) page 341: '[20] Specify preconditions and postconditions for your functions'
- (7) I.5: State preconditions (if any)
- (7) I.7: State postconditions
- (7) I.1: Make interfaces explicit
- (7) I.4: Make interfaces precisely and strongly typed

3.8 Function parts by importance

1. The name
2. The tests
3. The documentation
4. The body

Your boss will rarely care about the body of your functions.

3.9 Questions about functions?

Write down in the HackMD!

4 Making your code readable

4.1 Function name

The **name** is the most important part of a function.

- A good name makes it natural to use the function correctly.
- A bad name confuses the reader.

4.2 Function name

> There are only two hard things in Computer Science: cache invalidation and naming things.
>
> Phil Karlton

4.3 Function name conventions

- Starts with a verb
- All letters lowercase (10)
- All words separated by underscores (10)
- Common verbs abbreviations are allowed

```
print_hello_world()
```

4.4 Questions 1

Give the Python function name to determine if something ...

- is zero
- is bigger than zero
- is a number
- is a empty list

Write down in the HackMD!

4.5 Answers 1

- is zero: `is_zero`
- is bigger than zero: `is_positive`, `is_bigger_than_zero`
- is a number: `is_number`
- is a empty list: `is_empty_list`

4.6 Questions 2

Give the Python function name to determine if something ...

- is a list of strings
- is a list of integers
- is a list of integers of only zeroes

Write down in the HackMD!

4.7 Answers 2

- is a list of strings: `is_string_list`, `is_text`, `are_strings`
- is a list of integers: `is_integer_list`, `are_integers`, `are_ints`
- is a list of integers of only zeroes: `are_zeroes`

4.8 Exercise

- Fill in the blanks
- Multiple answers are correct

4.9 Question 3

```
def ___(number):
  return number == 0.0

assert ___(0.0)
assert not ___(3.14)
```

4.10 Answer 3

```
def is_zero(number):
  return number == 0.0

assert is_zero(0.0)
assert not is_zero(3.14)
```

4.11 Question 4

```
def ____(number):
    return number > 0.0

assert ___(1.0)
assert not ___(0.0)
assert not ___(-1.0)
```

4.12 Answer 4

```
def is_bigger_than_zero(number):
    return number > 0.0

assert is_bigger_than_zero(1.0)
assert not is_bigger_than_zero(0.0)
assert not is_bigger_than_zero(-1.0)
```

4.13 Question 5

```
def ___(my_list):
    return len(my_list) == 0

assert ___([])
assert not ___([1])
assert not ___([1, 2])
```

4.14 Answer 5

```
def is_empty_list(my_list):
    return len(my_list) == 0

assert is_empty_list([])
assert not is_empty_list([1])
assert not is_empty_list([1, 2])
```

4.15 Question 6

```
def ___(my_list):
  return sum(my_list)

assert ___([1]) == 1
assert ___([1, 2]) == 3
assert ___([1, 2, 3]) == 6
```

4.16 Answer 6

```
def calc_sum(my_list):
    return sum(my_list)

# assert calc_sum([]) == 0
assert calc_sum([1]) == 1
assert calc_sum([1, 2]) == 3
assert calc_sum([1, 2, 3]) == 6
```

4.17 Question 7

```
def ___(my_list):
  return sorted(my_list)

assert ___([2, 1]) == [1, 2]
assert ___([3, 2, 1]) == [1, 2, 3]
```

4.18 Answer 7

```
def create_sorted_list(my_list):
  return sorted(my_list)

assert create_sorted_list([2, 1]) \
  == [1, 2]
assert create_sorted_list([3, 2, 1]) \
  == [1, 2, 3]
```

4.19 Question 8

```
def ___(my_list):
  // Bonus: assert __
  return sum(my_list) / len(my_list)

assert ___([1, 3]) == 2
assert ___([1, 3, 5]) == 3
```

4.20 Answer 8

```
def calculate_mean(my_list):
  assert len(my_list)
  return sum(my_list) / len(my_list)

assert calculate_mean([1, 3]) == 2
assert calculate_mean([1, 3, 5]) == 3
```

4.21 Questions about function names?

Write down in the HackMD!

5 Do work in the right order

5.1 We were all kids once

Imagine you need to write a function, for example, to determine if a number is prime. **Remember** when you just started programming: what were your first lines of code?

Write down in the HackMD!

5.2 Your first code

```
def prime(number)
  for i in range(1, number)
    for j in range(2, i)
      if i %% j:
```

What is the problem?

Write down in the HackMD!

5.3 Next answer

We overload our working memory. We do too many things. Instead, work from big to small: write the tests first!

5.4 Test-Driven Development

Writing test first is a software methodology, called Test-Driven Development.

5.5 Effects of TDD

Test-Driven Development …

- makes developers more productive (11)
- increases quality of the code (11) (12) (13)
  - There are plenty of costly programming mistakes documented!
- helps shape the project architecture (14)
- helps better modularisation (15)
- works great with Xtreme programming and CI

5.6 For us

We 'only' use tests to express what we expect our function to do.

5.7 Example question

```
def is_float(number):
    # Google this!

assert is_float(3.14)
assert not is_float(42)
assert not is_float('Hello')
assert not is_float([1.2, 3.4])
assert not is_float((1.2, 3.4))
```

5.8 Example answer

```
def is_float(number):
    return isinstance(number, float)

assert is_float(3.14)
assert not is_float(42)
```

```
assert not is_float('Hello')
assert not is_float([1.2, 3.4])
assert not is_float((1.2, 3.4))
```

5.9 Question 1

Write the tests for a function to determine if a file exists.

Write down in the HackMD!

5.10 Answer 1

```
assert does_file_exist("day_3.qml")
assert not does_file_exist("abs.ent")
```

5.11 Question 2

Write the tests for a function to read the content of a file into a list of strings.

Write down in the HackMD!

5.12 Answer 2

```
assert len(read_file("my.txt"))
assert is_string(read_file("my.txt")[0])
assert is_text(read_file("my.txt"))
```

'a list of string' can be called 'text' :-)

5.13 Question 3

Write the tests for a function to read the content of a file into a list of strings, yet skipping the first line

Write down in the HackMD!

5.14 Answer 3

```
# Save space
f = read_file_without_header

assert len(f("day_3.qml"))
assert is_string(f("day_3.qml")[0])
assert is_text(f("day_3.qml"))
assert len(f("day_3.qml")) \
 == len(read_file("day_3.qml")) - 1
```

5.15 Question 4

Create a function that returns the text below as a list of strings. Call the function `create_test_table`.

```
First name|Last name
Alita     |Colbert
Brandi    |Lovell
Corrina   |Georgeanna
```

Write down in the HackMD!

5.16 Answer 4

```
# Save space
f = create_test_table

assert len(f()) == 4
assert f()[0] == "First name|Last name"
assert f()[1] == "Alita     |Colbert"
assert f()[2] == "Brandi    |Lovell"
assert f()[3] == "Corrina   |Georgeanna"
```

5.17 Question 5

Assume a table as text, including headers. Extract the `n`th column as a list of strings. Use `create_test_table` in your tests

Write down in the HackMD!

5.18 Answer 5

```
# Save space
f = get_nth_column

table = create_test_table()

# A column has no header
n_rows = len(table)
first_column = f(table, 0)
n_elements = len(first_column)
assert(n_rows - 1 == n_elements)

assert f(table, 0)[0] == "Alita"
assert f(table, 0)[1] == "Brandi"
assert f(table, 0)[2] == "Corrina"
assert f(table, 1)[0] == "Colbert"
assert f(table, 1)[1] == "Lovell"
assert f(table, 1)[2] == "Georgeanna"
```

5.19 Question 6

Split a string into its elements. The elements are separated by commas, e.g. `"A,B,C"`. Remove the whitespace at the edges.

Write down in the HackMD!

5.20 Answer 6

```
assert split_str("A") == ["A"]
assert split_str("A,B") == ["A", "B"]
assert split_str("A,B ") == ["A", "B"]
```

```
assert split_str(" A,B ") == ["A", "B"]
assert split_str("A ,B ") == ["A", "B"]
assert split_str("A, B ") == ["A", "B"]
```

5.21 Questions about function testing?

Write down in the HackMD!

6 Make your function usable by others

6.1 Adding documentation

```
def split_str(x):
  """
  Split 'x' into its comma-seperated
  parts.
  Assumes 'x' is a string
  of at least 1 character;
  will terminate the program
  if not
  """
  # Do it
```

6.2 Questions about function documentation?

Write down in the HackMD!

- More on this tomorrow

7 Writing the function body

7.1 Question

When is a function body good enough?

Write down in the HackMD!

7.2 Writing a good function body

If all test pass, it is good. Done!

7.3 Speed?

> Premature optimization is the root of all evil.
>
> Donald Knuth

7.4 Questions about writing a function body?

Write down in the HackMD!

7.5 Exercise

Do

- `Day 3 -> Exercise 1 - functions`
- `Day 3 -> PyQuiz 3.2`

8 Expressing a set

8.1 Question

What is set?

Write down in the HackMD!

8.2 Answer

'A collection things'

For example, the ages of people in a room:

```
ages = []
ages = [8]
ages = [8]
ages = [8, 18]
```

8.3 Question

Write the **tests** for a function to add values to a list **only** when a value is not present yet.

Write down in the HackMD!

8.4 Answer

```
f = add_value_to_set
ages = []
ages = f(ages, 8)
ages = f(ages, 8)
ages = f(ages, 18)
assert ages == [8, 18]
```

8.5 Question

Would it be possible to express this better? How?

Write down in the HackMD!

8.6 Answer

One can use a Python set.

```
ages = {}
ages.add(8)
ages.add(8)
ages.add(18)
assert ages == {8, 18}
assert ages == [8, 18]
```

8.7 Reflection

Classes ...

- can be built-in, e.g. string, lists and sets
- have a clear purpose
- allow us to express ourselves **elegantly**
- help us to **maintain overview**, e.g. we know things about elements in a set

8.8 Questions about sets?

Write down in the HackMD!

9 Expressing a dictionary

9.1 Trick question

Write the tests for some functions to work with a telephone book.

One function allows to add a name and phone number. If the name already exists, overwrite the phone number.

9.2 Naive answer

```
# To save space
p = create_phone_book()
p = add(p, "Aardvark", 1234567890)
p = add(p, "Zziiz", 1234567891)
assert get_phone_number( \
  p, "Aardvark") \
  == 1234567890
```

This will be a *tour de force*!

9.3 Better answer

Use a dictionary!

```
# To save space
p = { \
  "Aardvark": 1234567890, \
  "Zziiz": 1234567891 \
  }
assert p["Aardvark"] == 1234567890
p["Zziiz"] = 9876543210
assert p["Zziiz"] == 9876543210
```

9.4 Questions about dictionaries?

Write down in the HackMD!

9.5 Exercise

Do

- `Day 3 -> IMDb exercise - Day 3`
- `Day 3 -> PyQuiz 3.1`

10 Using system arguments

10.1 Problem

Imagine you have written a useful Python script. You want to run the script in two different ways:

- Quiet: no output
- Verbose: with many prints to the screen

How to do this?

Write down in the HackMD!

10.2 Answer

Call the script differently, for example:

```
python3 my_script.py --verbose
python3 my_script.py --quiet
```

10.3 Exercise, little help

Write a Python script that …

- when run with `--verbose` states it is running verbosely
- when run with `--quiet` (ironically) states it is running quietly

Either Google or read the next slide for hints

Write down in the HackMD!

10.4 Exercise, more help

```
import sys

print(sys.argv)
```

`sys.argv` is a list of strings.

10.5 Answer

```
import sys

assert len(sys.argv) == 2
if sys.argv[1] == "--quiet":
  print("Quiet")
elif sys.argv[1] == "--verbose":
  print("Verbose!")
else
  assert not "Unknown argument"
```

10.6 Questions about command line arguments?

Write down in the HackMD!

11 Conclusion

To be able to express your ideas in correct and readable code in an elegant way.

- Express assumptions with `asserts`
- Use functions to make code readable
- Use tests to verify the code is correct
- Use sets or dictionaries when this fits naturally

11.1 Questions about today's theory?

Write down in the HackMD!

If not, apply this in The Project:

- Use `assert`
- Split code into functions
- Test functions

12 Done!

Go home and rest :-)

13 Breaks

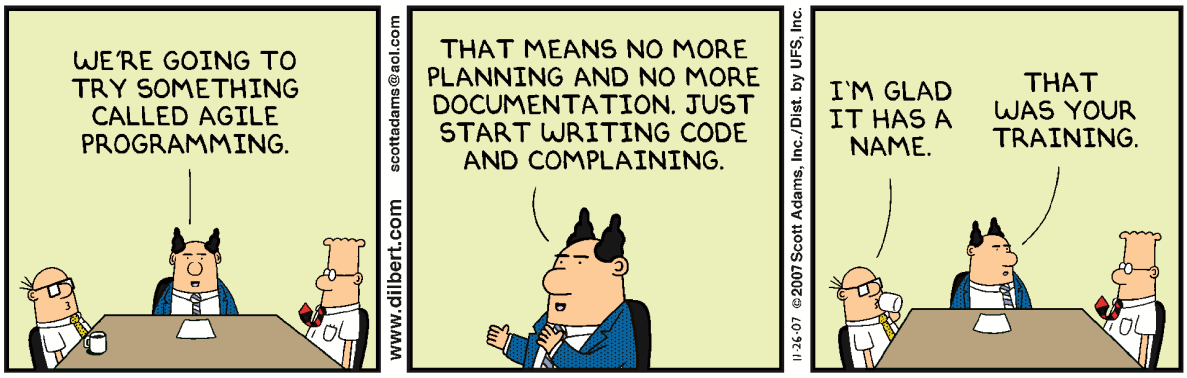
Are important. Please rest!

13.1 Break 1: 10:00-10:15

13.2 Break 2: 11:00-11:15

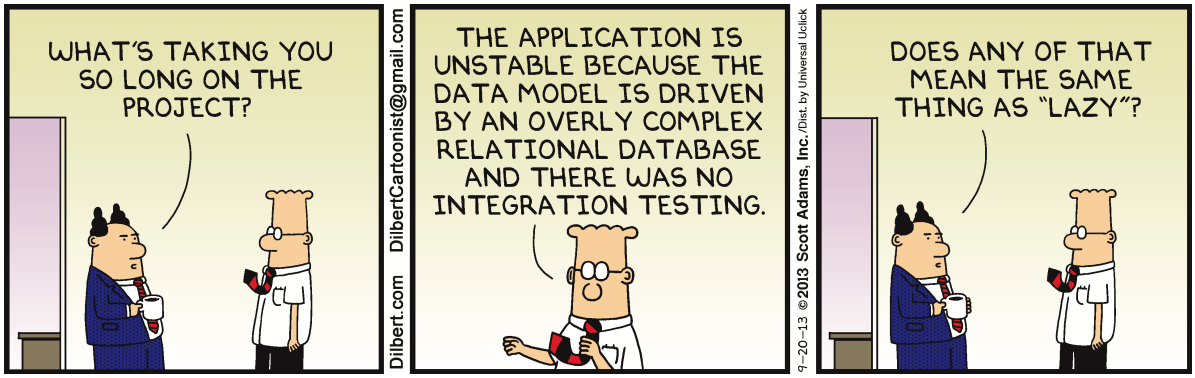
13.3 Break 3: 12:00-13:00

13.4 Break 4: 14:00-14:15



13.5 Break 5: 15:00-15:15

13.6 Break 6: 16:00-16:15



13.7 Done

References

1. Newport C. Deep work: Rules for focused success in a distracted world. Hachette UK; 2016.
2. Sutter H, Alexandrescu A. C++ coding standards: 101 rules, guidelines, and best practices. Pearson Education; 2004.
3. Stroustrup B. The c++ programming language. Pearson Education; 2013.
4. McConnell S. Code complete. Pearson Education; 2004.
5. Liberty J. Sams teach yourself c++ in 24 hours. Sams publishing; 2002.
6. Lakos J. Large-scale c++ software design. Reading, MA. 1996;173:217–71.
7. Stroustrup B, Sutter H, et al. C++ core guidelines. Web Last accessed February. 2018;
8. Meyers S. Effective c++: 55 specific ways to improve your programs and designs. Pearson Education; 2005.
9. Martin L. Joint strike fighter, air vehicle, c++ coding standard. Lockheed Martin, December; 2005.
10. Van Rossum G, Warsaw B, Coghlan N. PEP 8–style guide for Python code. Python org. 2001;1565.
11. Erdogmus H, Morisio M, Torchiano M. On the effectiveness of the test-first approach to programming. IEEE Transactions on software Engineering. 2005;31(3):226–37.
12. Alkaoud H, Walcott KR. Quality metrics of test suites in test-driven designed applications. International Journal of Software Engineering Applications (IJSEA). 2018;2018.
13. Janzen DS, Saiedian H. Test-driven learning: Intrinsic integration of testing into the CS/SE curriculum. Acm Sigcse Bulletin. 2006;38(1):254–8.
14. Mayr H. Projekt engineering: Ingenieurmäßige softwareentwicklung in projektgruppen. Hanser Verlag; 2005.
15. Madeyski L, información G de sistemas de. Test-driven development: An empirical evaluation of agile practice. Springer; 2010.