Open Universiteit

Test Informed Learning with Examples (TILE)

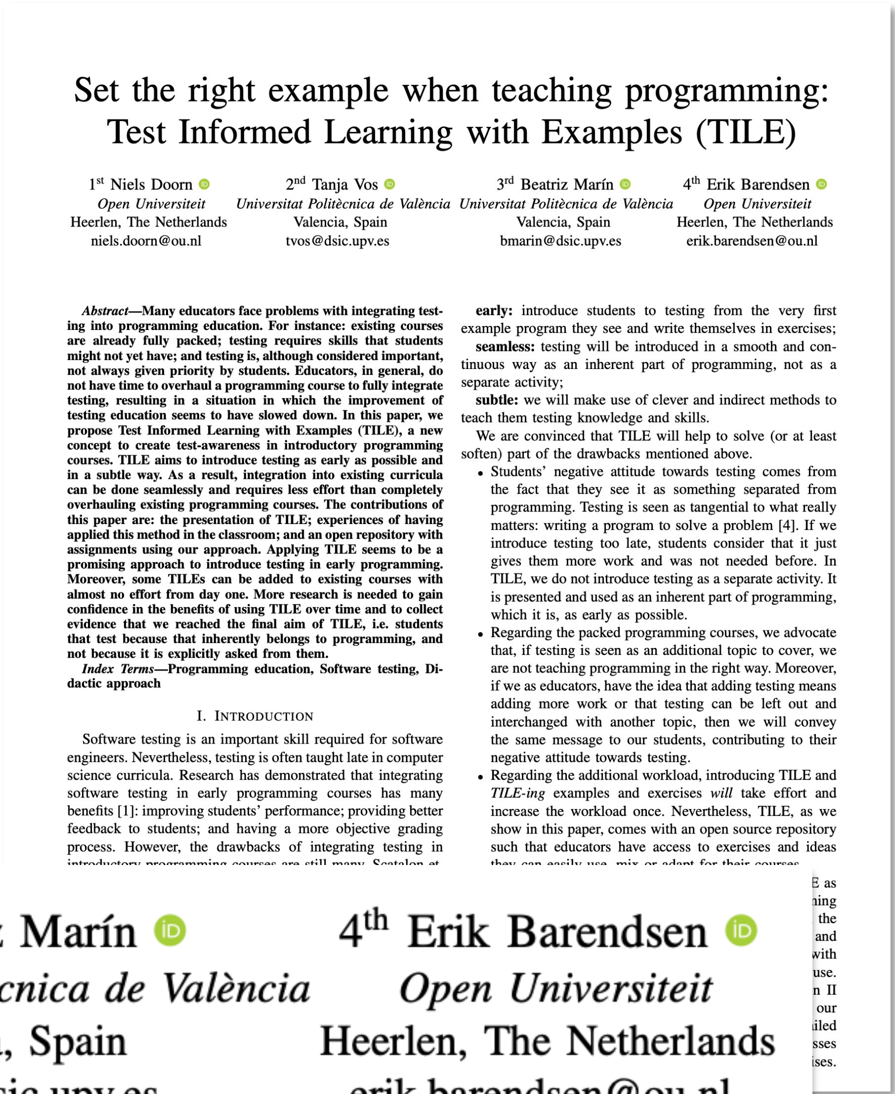
Set the right example when teaching programming

Niels Doorn

Open Universiteit

NHL STENDEN university of applied sciences

QPED_



Set the right example when teaching programming: Test Informed Learning with Examples (TILE)

1st Niels Doorn
Open Universiteit
Heerlen, The Netherlands
niels.doorn@ou.nl

2nd Tanja Vos
Universitat Politècnica de València
Valencia, Spain
tvos@dsic.upv.es

3rd Beatriz Marín
Universitat Politècnica de València
Valencia, Spain
bmarin@dsic.upv.es

4th Erik Barendsen
Open Universiteit
Heerlen, The Netherlands
erik.barendsen@ou.nl

***Abstract*—Many educators face problems with integrating testing into programming education. For instance: existing courses are already fully packed; testing requires skills that students might not yet have; and testing is, although considered important, not always given priority by students. Educators, in general, do not have time to overhaul a programming course to fully integrate testing, resulting in a situation in which the improvement of testing education seems to have slowed down. In this paper, we propose Test Informed Learning with Examples (TILE), a new concept to create test-awareness in introductory programming courses. TILE aims to introduce testing as early as possible and in a subtle way. As a result, integration into existing curricula can be done seamlessly and requires less effort than completely overhauling existing programming courses. The contributions of this paper are: the presentation of TILE; experiences of having applied this method in the classroom; and an open repository with assignments using our approach. Applying TILE seems to be a promising approach to introduce testing in early programming. Moreover, some TILEs can be added to existing courses with almost no effort from day one. More research is needed to gain confidence in the benefits of using TILE over time and to collect evidence that we reached the final aim of TILE, i.e. students that test because that inherently belongs to programming, and not because it is explicitly asked from them.**

***Index Terms*—Programming education, Software testing, Didactic approach**

I. Introduction

Software testing is an important skill required for software engineers. Nevertheless, testing is often taught late in computer science curricula. Research has demonstrated that integrating software testing in early programming courses has many benefits [1]: improving students' performance; providing better feedback to students; and having a more objective grading process. However, the drawbacks of integrating testing in

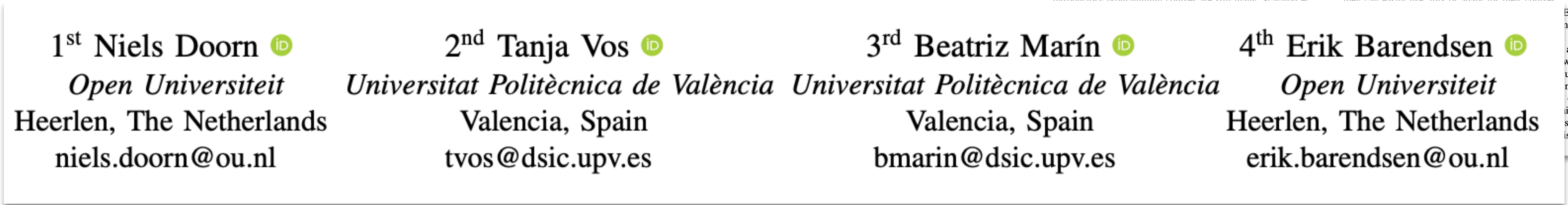
early: introduce students to testing from the very first example program they see and write themselves in exercises;

seamless: testing will be introduced in a smooth and continuous way as an inherent part of programming, not as a separate activity;

subtle: we will make use of clever and indirect methods to teach them testing knowledge and skills.

We are convinced that TILE will help to solve (or at least soften) part of the drawbacks mentioned above.

- Students' negative attitude towards testing comes from the fact that they see it as something separated from programming. Testing is seen as tangential to what really matters: writing a program to solve a problem [4]. If we introduce testing too late, students consider that it just gives them more work and was not needed before. In TILE, we do not introduce testing as a separate activity. It is presented and used as an inherent part of programming, which it is, as early as possible.
- Regarding the packed programming courses, we advocate that, if testing is seen as an additional topic to cover, we are not teaching programming in the right way. Moreover, if we as educators, have the idea that adding testing means adding more work or that testing can be left out and interchanged with another topic, then we will convey the same message to our students, contributing to their negative attitude towards testing.
- Regarding the additional workload, introducing TILE and *TILE-ing* examples and exercises *will* take effort and increase the workload once. Nevertheless, TILE, as we show in this paper, comes with an open source repository such that educators have access to exercises and ideas



1st Niels Doorn
Open Universiteit
Heerlen, The Netherlands
niels.doorn@ou.nl

2nd Tanja Vos
Universitat Politècnica de València
Valencia, Spain
tvos@dsic.upv.es

3rd Beatriz Marín
Universitat Politècnica de València
Valencia, Spain
bmarin@dsic.upv.es

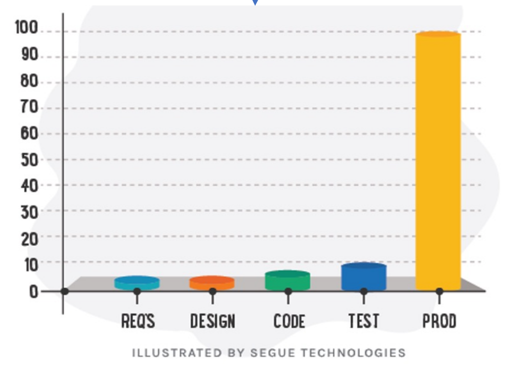
4th Erik Barendsen
Open Universiteit
Heerlen, The Netherlands
erik.barendsen@ou.nl

WHY??

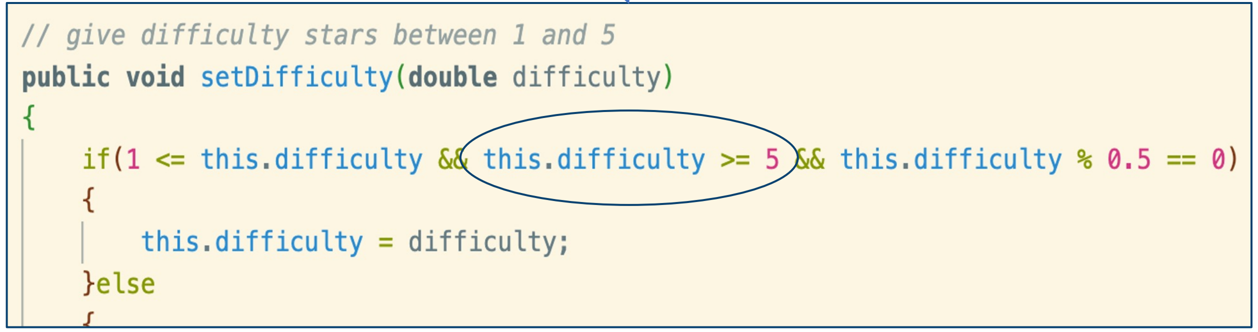
Software testing is very important…

…but also problematic in education

Testing early is very effective to measure software quality and avoid high costs

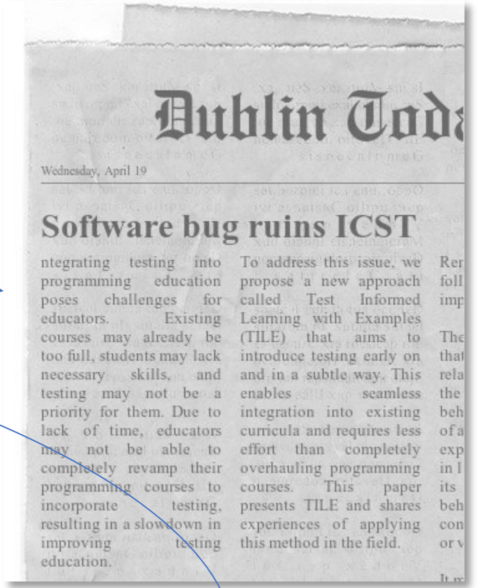


Students don't test their code very well



```
// give difficulty stars between 1 and 5
public void setDifficulty(double difficulty)
{
    if(1 <= this.difficulty && this.difficulty >= 5 && this.difficulty % 0.5 == 0)
    {
        this.difficulty = difficulty;
    }else
    {
```

Software failures are to be avoided



Dublin Tod

Wednesday, April 19

Software bug ruins ICST

ntegrating testing into programming education poses challenges for educators. Existing courses may already be too full, students may lack necessary skills, and testing may not be a priority for them. Due to lack of time, educators may not be able to completely revamp their programming courses to incorporate testing, resulting in a slowdown in improving testing education.

To address this issue, we propose a new approach called Test Informed Learning with Examples (TILE) that aims to introduce testing early on and in a subtle way. This enables seamless integration into existing curricula and requires less effort than completely overhauling programming courses. This paper presents TILE and shares experiences of applying this method in the field.

Rer
foll
imp

The
that
rela
the
beh
of a
exp
in l
its
beh
con
or v

It m

There are not many evidence based didactical approaches

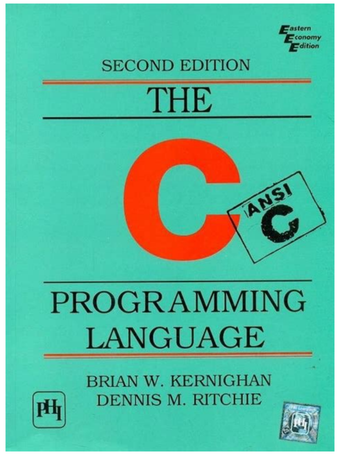
Educators struggle with teaching software testing

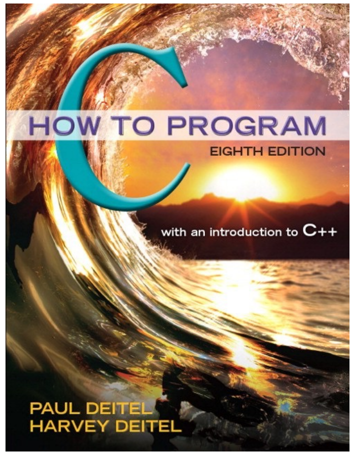
TESTING IS INTRODUCED LATE! JUST LOOK AT THE BOOKS

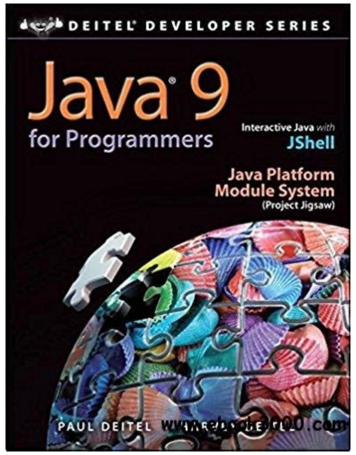
→ Ten commonly used books on C, Java, Python

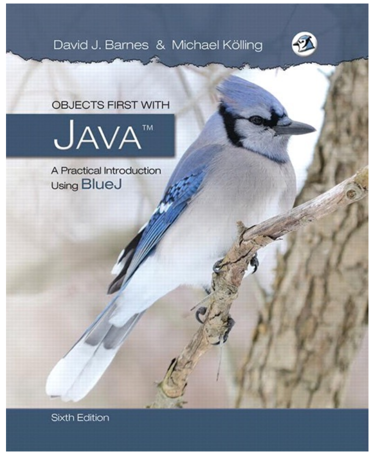
- Use of TILE constructs in exercises
- When is testing introduced
- When is assert introduced

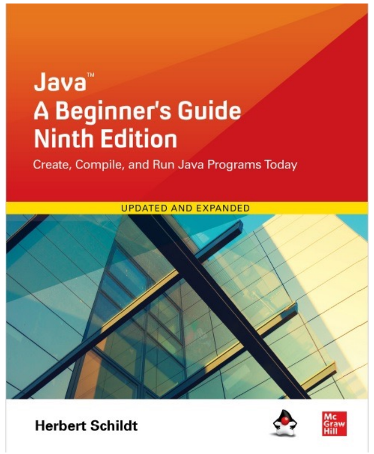
TESTING IS INTRODUCED LATE! JUST LOOK AT THE BOOKS



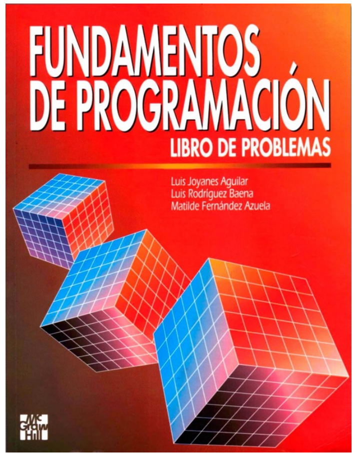


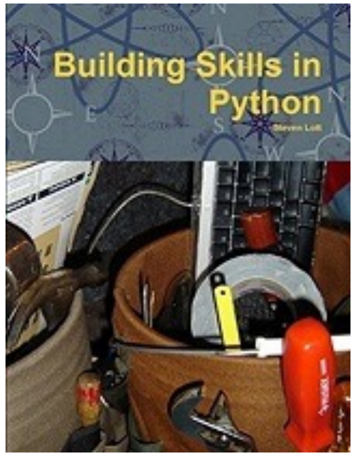

DEITEL DEVELOPER SERIES
Java 9 for Programmers
Interactive Java with JShell
Java Platform Module System (Project Jigsaw)
PAUL DEITEL

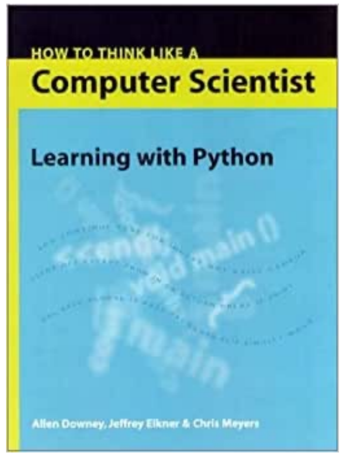

David J. Barnes & Michael Kölling
OBJECTS FIRST WITH JAVA
A Practical Introduction Using BlueJ
Sixth Edition

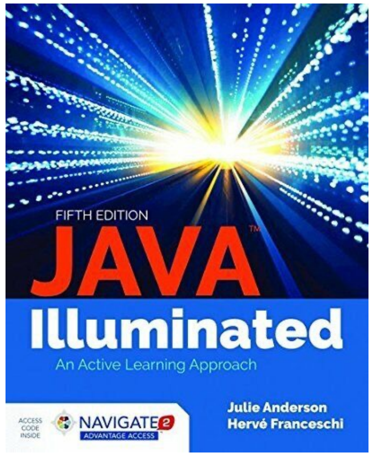

Java A Beginner's Guide Ninth Edition
Create, Compile, and Run Java Programs Today
UPDATED AND EXPANDED
Herbert Schildt

Ignace Van de Woestyne & Jan Vanthienen
SOFTWAREONTWERP EN PROGRAMMEREN MET PYTHON 3
acco

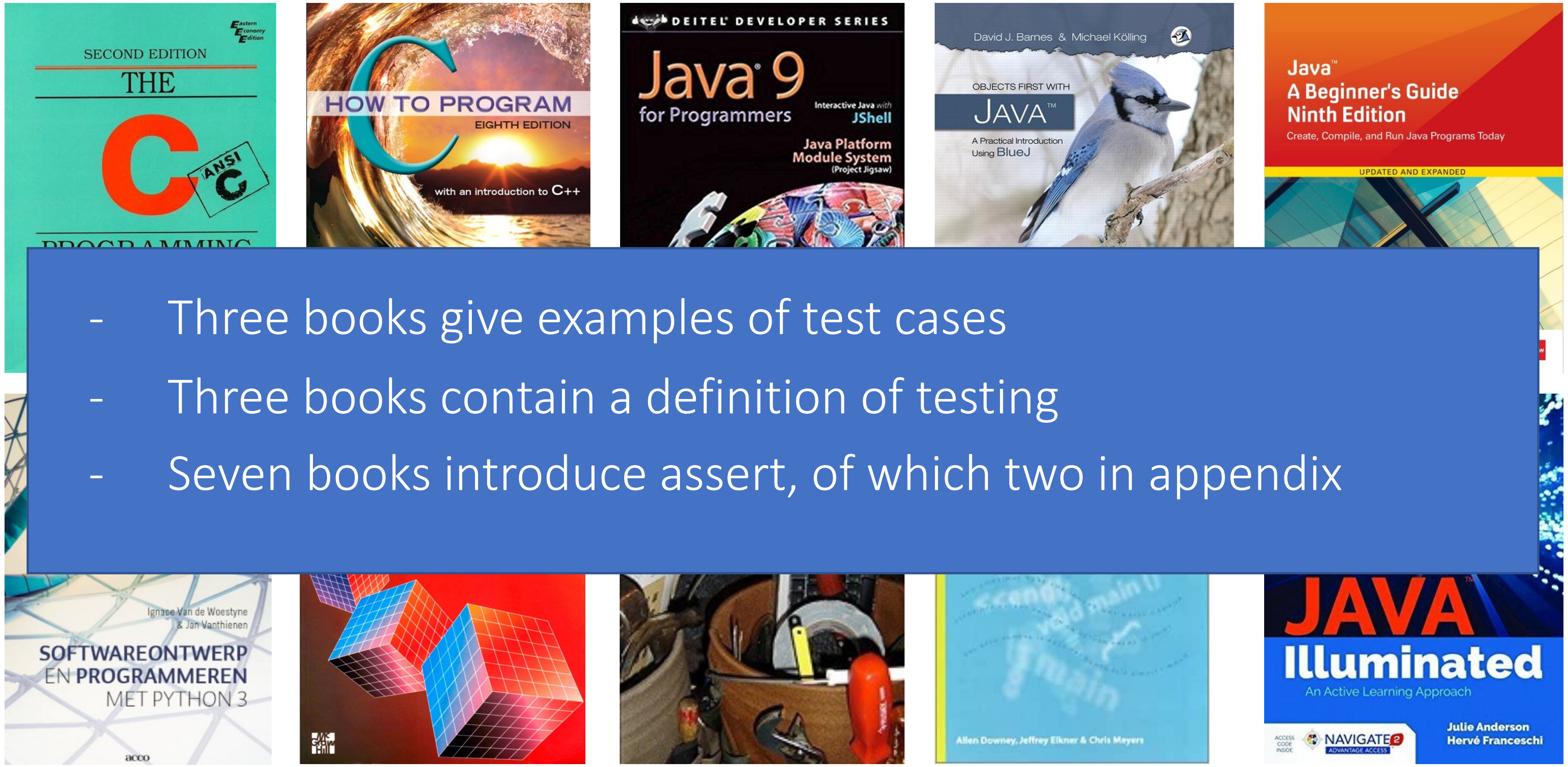





HOW TO THINK LIKE A Computer Scientist
Learning with Python
Allen Downey, Jeffrey Elkner & Chris Meyers

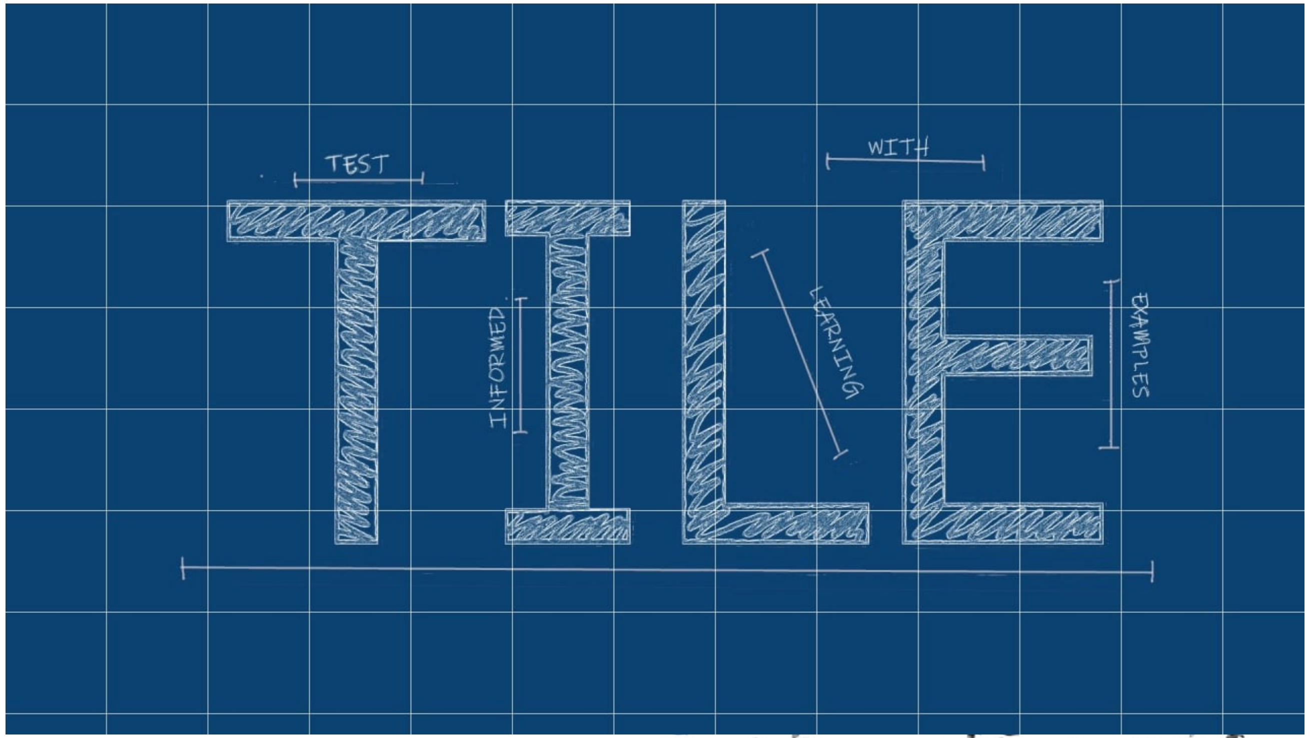


TESTING IS INTRODUCED LATE! JUST LOOK AT THE BOOKS


SECOND EDITION
THE C PROGRAMMING
ANSI C
C HOW TO PROGRAM
EIGHTH EDITION
with an introduction to C++
DEITEL DEVELOPER SERIES
Java 9 for Programmers
Interactive Java with JShell
Java Platform Module System (Project Jigsaw)
David J. Barnes & Michael Kölling
OBJECTS FIRST WITH JAVA
A Practical Introduction Using BlueJ
Java A Beginner's Guide Ninth Edition
Create, Compile, and Run Java Programs Today
UPDATED AND EXPANDED
Ignace Van de Woestyne & Jan Vanthienen
SOFTWAREONTWERP EN PROGRAMMEREN MET PYTHON 3
acco
Allen Downey, Jeffrey Elkner & Chris Meyers
JAVA Illuminated
An Active Learning Approach
Julie Anderson
Hervé Franceschi
NAVIGATE 2

- Three books give examples of test cases
- Three books contain a definition of testing
- Seven books introduce assert, of which two in appendix

Open Universiteit



Test Informed Learning with Examples

What is TILE and how does it help?

WHAT IS TILE?

A new approach to introduce software testing:

Early - from the first programming exercise
Seamless – as an inherent part of **programming** education
Subtle - clever and indirect

THREE TYPES OF TILES

Test **run** TILEs
Test **cases** TILEs
Test **message** TILEs

Open Universiteit

1: Test run TILEs

TEST RUN TILES

We can ask the students to **test** the program instead of asking them to **run** the program

TEST RUN TILES

We can ask the students to **test** the program instead of asking them to **run** the program

Consider the following program:

```
n = int(input("Enter a number: "))
square = n * n
print("The square is: ", square)
```

Compare the wording of the following two ways:

1. Now let us **run** this program, the user can give input through the keyboard and the results will be shown on the screen
2. Now let us **test** this program by running it and **entering test input data** through the keyboard and **checking the resulting output** on the screen

Open Universiteit

2: Test cases TILEs

TEST CASES TILES

Students often only test **happy path** execution

We can add **add more concrete examples of possible test cases** to create awareness of other useful test cases

TEST CASES TILES

Students often only test **happy path** execution
We can add **add more concrete examples of possible test cases** to create awareness of other useful test cases

Test case TILEs come in different shape and form:

1. We can add **example test executions,**
2. or add **example test cases,**
3. make students think about **combinations and boundary values,**
4. and we can point students to a **parallel oracle**.

TEST CASES TILES: PRESENTING TEST CASES

Students often only test **happy path** execution
We can add **add more concrete examples of possible test cases** to create awareness of other useful test cases

Exercise: *Implement a program that asks the user for a comparison operator:* `<`, `<=`, `>`, `>=`, `==`, `!=` *and 2 values. Your program has to display on screen the result* (`True` or `False`) *of the given operation applied to the two values.*

test id	test inputs			expected output
	operator	value1	value2	
1	<	12	4	`False`
2	>	100	40	`True`
3	==	`"Hello!"`	40	`False`
4	!=	100	`"Python"`	`True`
5	>=	98.67	0.45	`True`
6	<=	-100	40	`True`
7	<	24	`"24K"`	`True`
8	>=	`"email"`	`"correo"`	`True`

Open Universiteit

3: Test message TILEs

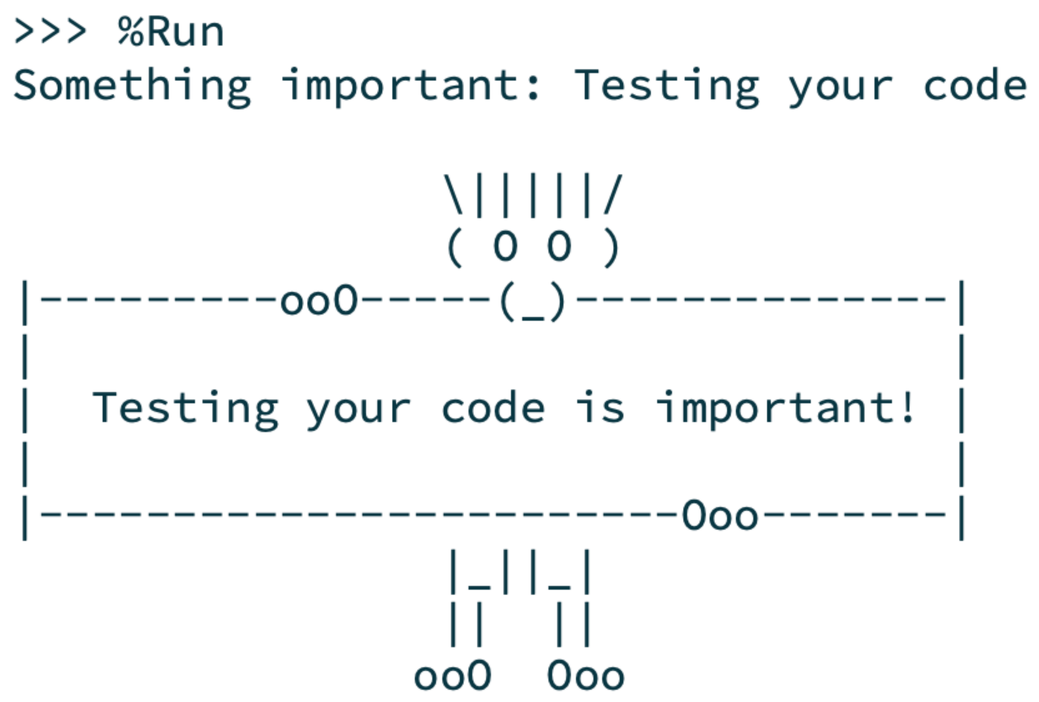
TEST MESSAGE TILES

TILEs of this type hide **a subliminal message** about the importance of testing.

TILEs of this type hide a **subliminal message** about the importance of testing.

TEST MESSAGE TILES

Exercise: Write a program that asks the user for something important and returns a billboard ASCII art.



```
>>> %Run
Something important: Testing your code

                \|||||/
                ( O O )
|---------ooO-----(_)--------------|
|                                  |
|  Testing your code is important! |
|                                  |
|------------------------Ooo-------|
                |_||_|
                ||  ||
               ooO  Ooo
```

Open Universiteit

Applying TILE in an existing course

Our experiences

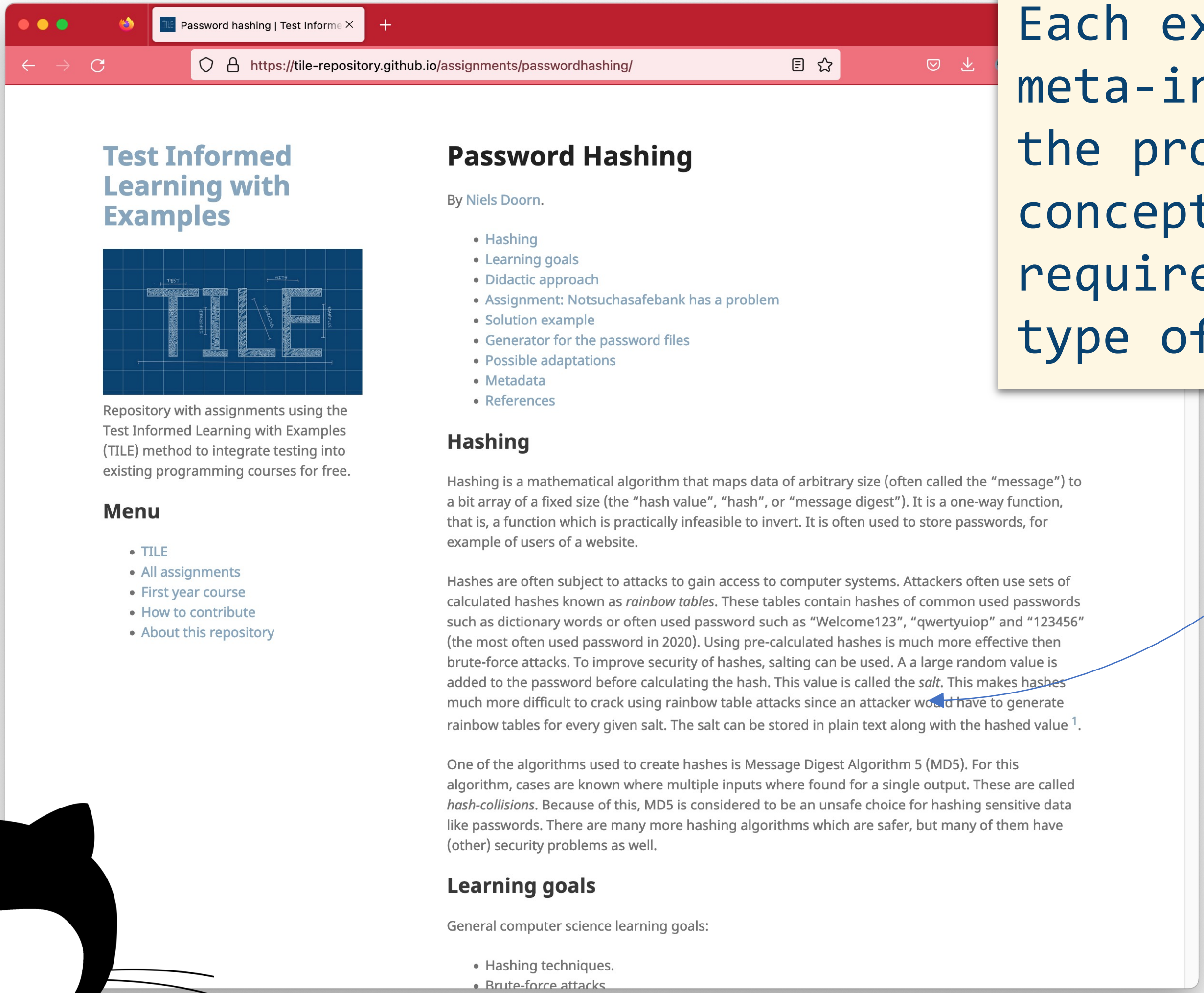
APPLYING TILE IN AN EXISTING COURSE

→ First year Bachelor Python course
- All exercises have been TILEd
- Test run TILEs require little effort
- Test cases TILEs increases the size of the workbook
- Students started to think more like testers
- Exercises were better understood
- Students became enthousiatic about testing
- It is challenging to get colleagues involved

Open Repository

OPEN REPOSITORY

We created an open repository containing TILEd exercises usable in excisting courses


Password hashing | Test Informe
https://tile-repository.github.io/assignments/passwordhashing/

Test Informed Learning with Examples

Repository with assignments using the Test Informed Learning with Examples (TILE) method to integrate testing into existing programming courses for free.

Menu

- TILE
- All assignments
- First year course
- How to contribute
- About this repository

Password Hashing

By Niels Doorn.

- Hashing
- Learning goals
- Didactic approach
- Assignment: Notsuchasafebank has a problem
- Solution example
- Generator for the password files
- Possible adaptations
- Metadata
- References

Hashing

Hashing is a mathematical algorithm that maps data of arbitrary size (often called the "message") to a bit array of a fixed size (the "hash value", "hash", or "message digest"). It is a one-way function, that is, a function which is practically infeasible to invert. It is often used to store passwords, for example of users of a website.

Hashes are often subject to attacks to gain access to computer systems. Attackers often use sets of calculated hashes known as *rainbow tables*. These tables contain hashes of common used passwords such as dictionary words or often used password such as "Welcome123", "qwertyuiop" and "123456" (the most often used password in 2020). Using pre-calculated hashes is much more effective then brute-force attacks. To improve security of hashes, salting can be used. A a large random value is added to the password before calculating the hash. This value is called the *salt*. This makes hashes much more difficult to crack using rainbow table attacks since an attacker would have to generate rainbow tables for every given salt. The salt can be stored in plain text along with the hashed value [1].

One of the algorithms used to create hashes is Message Digest Algorithm 5 (MD5). For this algorithm, cases are known where multiple inputs where found for a single output. These are called *hash-collisions*. Because of this, MD5 is considered to be an unsafe choice for hashing sensitive data like passwords. There are many more hashing algorithms which are safer, but many of them have (other) security problems as well.

Learning goals

General computer science learning goals:

- Hashing techniques.
- Brute-force attacks

Each exercise contains meta-information about the programming concepts taught, required pre-knowledge, type of TILE et cetera

Everybody can contribute!

edu.nl/f9ptp

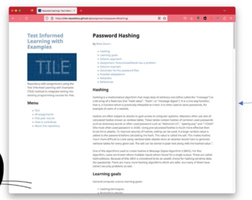
Test Informed Learning with Examples

Students don’t test their code very well

```
// give difficulty stars between 1 and 5
public void setDifficulty(double difficulty)
{
    if(1 <= this.difficulty && this.difficulty >= 5 && this.difficulty % 0.5 == 0)
    {
        this.difficulty = difficulty;
    }else
```

OPEN REPOSITORY

We have created an open repository containing TILEd exercises usable in excisting courses


Test Informed Learning with Examples

Repository with assignments using the Test Informed Learning with Examples (TILE) method to integrate testing into existing programming courses for free.

Menu

- TILE
- All assignments
- First year course
- How to contribute
- About this repository

Password Hashing

By Niels Doorn.

- Hashing
- Learning goals
- Didactic approach
- Assignment: Notsuchasafebank has a problem
- Solution example
- Generator for the password files
- Possible adaptations
- Metadata
- References

Hashing

[illegible]

Learning goals

General computer science learning goals:

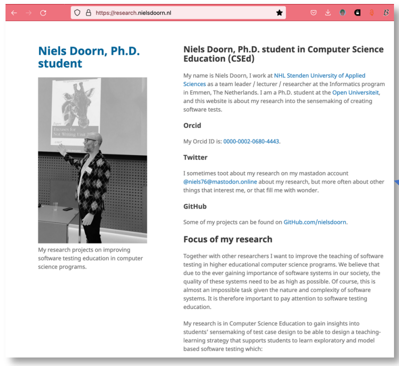
- Hashing techniques.
- Brute-force attacks.

Each exercise contains meta-information about the programming concepts taught, required pre-knowledge, type of TILE et cetera

edu.nl/f9ptp

Everybody can contribute!

Test Informed Learning with Examples


https://research.nielsdoorn.nl

Niels Doorn, Ph.D. student

My research projects on improving software testing education in computer science programs.

Niels Doorn, Ph.D. student in Computer Science Education (CSEd)

My name is Niels Doorn, I work at NHL Stenden University of Applied Sciences as a team leader / lecturer / researcher at the Informatics program in Emmen, The Netherlands. I am a Ph.D. student at the Open Universiteit, and this website is about my research into the sensemaking of creating software tests.

Orcid

My Orcid ID is: 0000-0002-0680-4443.

Twitter

I sometimes toot about my research on my mastadon account @niels76@mastodon.online about my research, but more often about other things that interest me, or that fill me with wonder.

GitHub

Some of my projects can be found on GitHub.com/nielsdoorn.

Focus of my research

Together with other researchers I want to improve the teaching of software testing in higher educational computer science programs. We believe that due to the ever gaining importance of software systems in our society, the quality of these systems need to be as high as possible. Of course, this is almost an impossible task given the nature and complexity of software systems. It is therefore important to pay attention to software testing education.

My research is in Computer Science Education to gain insights into students’ sensemaking of test case design to be able to design a teaching-learning strategy that supports students to learn exploratory and model based software testing which:

Read more about this research

edu.nl/utgcw

WHAT IS TILE?

A new approach to introduce software testing:

Early
Seamless
Subtle

- Education on Software testing needs to **improve**
- Educators **lack time** to overhaul existing courses
- TILE introduces testing from the **first exercise**

Please **join our community** and contribute to our repository

TEST RUN TILES

We can ask the students to **test** the program instead of asking them to **run** the program

Consider the following program:

```
n = int(input("Enter a number: "))
square = n * n
print("The square is: ", square)
```

Compare the wording of the following two ways:

1. Now let us **run** this program, the user can give input through the keyboard and the results will be shown on the screen
2. Now let us **test** this program by running it and **entering test input data** through the keyboard and **checking the resulting output** on the screen

TEST MESSAGE TILES

TILEs of this type hide a **subliminal message** about the importance of testing.

Exercise: Write a program that asks the user for something important and returns a billboard ASCII art.

```
>>> %Run
Something important: Testing your code

                   \|||||/
                   ( O O )
|---------ooO-----(_)--------------|
|                                  |
|  Testing your code is important! |
|                                  |
|----------------------Ooo-------- |
                   |_||_|
                   ||  ||
                  ooO  Ooo
```

TEST CASES TILES: PRESENTING TEST CASES

Students often only test **happy path execution**
We can add **add more concrete examples of possible test cases** to create awareness of other useful test cases

Exercise: *Implement a program that asks the user for a comparison operator:* `<`, `<=`, `>`, `>=`, `==`, `!=` *and 2 values. Your program has to display on screen the result* (`True` or `False`) *of the given operation applied to the two values.*

test id	test inputs			expected output
	operator	value1	value2	
1	<	12	4	False
2	>	100	40	True
3	==	"Hello!"	40	False
4	!=	100	"Python"	True
5	>=	98.67	0.45	True
6	<=	-100	40	True
7	<	24	"24K"	True
8	>=	"email"	"correo"	True

niels.doorn@ou.nl

QPED_ NHL STENDEN university of applied sciences Open Universiteit