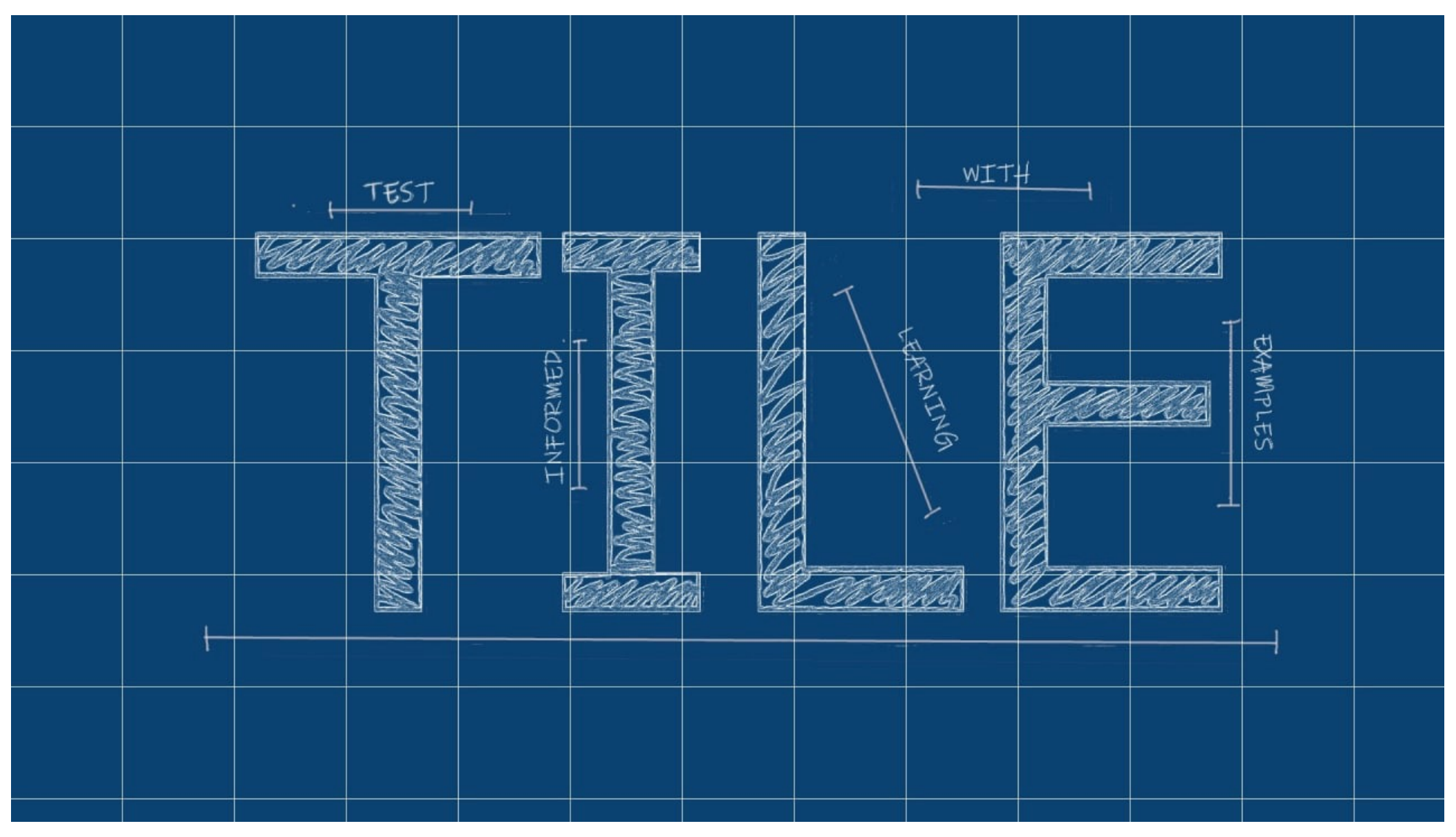


Test Informed Learning with Examples (TILE)

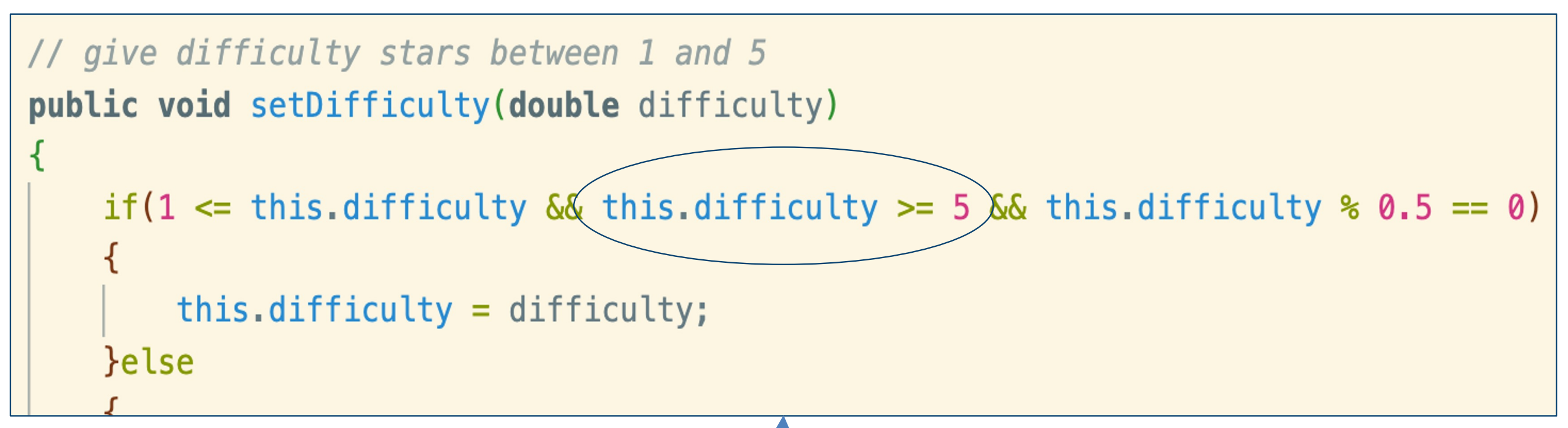
Setting the right example when teaching programming

Software testing is very important…

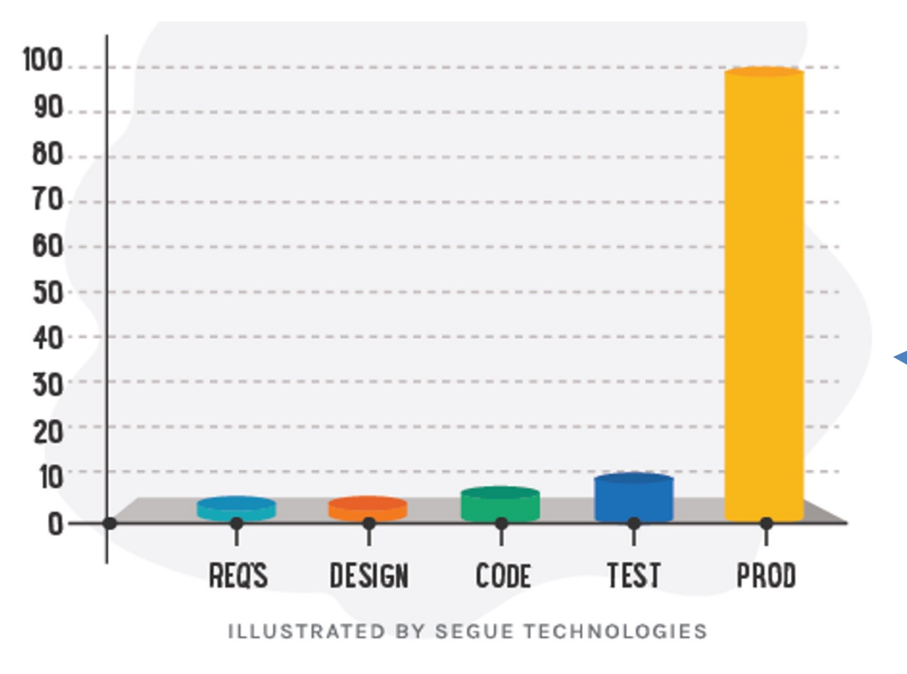
…but also problematic in both industry and education

Software failures are to be avoided

Educators struggle with teaching test case design



```
// 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
    {
```



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

NWO Daily

20-4-2023

Software bug ruins ICT.OPI

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; %with a controlled ex

Students don't test their code very well

There are no evidence based didactical approaches

TILE creates test aware programming courses

A new approach to introduce software testing:

1. *Early* – start learning to test from the very first exercise
2. *Seamless* – testing will be introduced in a smooth and continuous way as an inherent part of programming, and not as a separate activity
3. *Subtle* – using clever methods to teach testing knowledge and skills

There are four different ways to create TILEs from existing exercises:

Test run TILEs: we can ask the students to *test* the program instead of asking them to *run* the program.

Example of a test run TILE

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

Example of a test message TILE

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: add examples of, or ideas about, possible *test cases*

Example of a test case TILE: presenting 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	test inputs			expected
id	operator	value1	value2	output
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

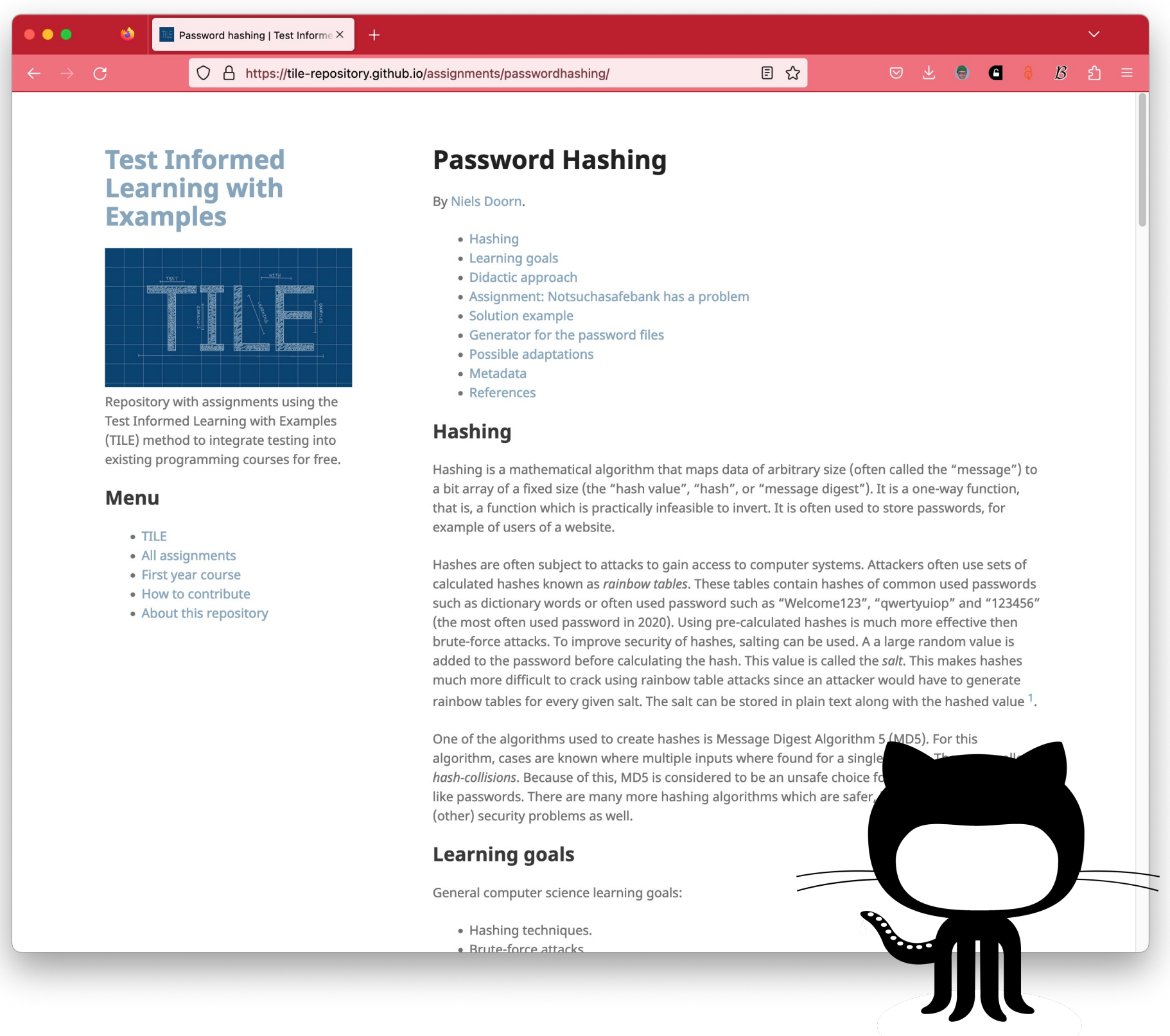
Test domain TILEs: replace the domain of an exercise with examples from the testing domain.

Example of a domain TILE

Exercise: Imagine Ana wrote a program that sorts a list of numbers. Evidently, this needs to be tested. Create a program that guides her through the process of deciding whether she has tested sufficiently as shown below. Try to extend the program with possible cases that test other important things (at least two are useful to add).

```
>>> %Run
  Hello! I'm gonna help you improve your
  sorting program!

  Did you check a basic case like:
  [3, 1, 8] is sorted into [1, 3, 8]? (y/n) y

  Excellent!

  Did you check what happens when the list
  is empty? (y/n) y

  Nice.

  Did you check what happens for a list with a
  single element, like [3]? (y/n) y

  Well done!

  Did you verify it also works with negative
  numbers, like [4, -8, 10]? (y/n) n

  You'd better try that right now!

  That was my last question!
  You took care of 75% of the cases. Well done!
```

We have created an open repository containing TILEd exercises usable in existing 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 *hash-collisions*. Because of this, MD5 is considered to be an unsafe choice fo like passwords. There are many more hashing algorithms which are safer, (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

edu.nl/utgcw

Open Universiteit

QPED_

NHL STENDEN university of applied sciences

Read more about TILE at research.nielsdoorn.nl