Opleiding Informatica

Creating a Framework for
Sublevels in Hedy

Thera Smit

Supervisors:
Felienne Hermans & Sabiha Yeni

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

June 19, 2021

Abstract

Learning to program can be tough, as there is a lot of syntax and new concepts to learn. The gradual programming language Hedy gives novice programmers a way to learn programming by introducing new concepts in a gradual way. This thesis attempts to expand Hedy by adding a framework for sublevels to even out some of the bigger steps between levels. The thesis focusses on the introduction of the for loop. It describes the creation of a sublevel to help introduce the for loop in a simpler way and shows the reasoning behind the creation of the sublevel framework.

Contents

1 Introduction

When first learing to program, people are confronted with a huge amount of information. Taking Python as an example, one needs to learn about keywords, indentation, semicolons and more. Syntax can be quite overwhelming when first confronted with it. The problem with learning a programming language is that all syntax has to be learned at once, otherwise the computer won't understand it. Forget the semicolon, syntax error. Leave out the indentation, indentation error. For novice programmers this makes it hard to get started, as memorising everything at once is almost impossible. Programming courses have tried to solve this problem by introducing the language in small steps, but those small steps usually still involve a lot of new concepts at once. Take the for loop for example. In Python the for loop consists of no less than 9 different parts which all have to be understood and used at once, see figure 1. Leave one out, the program breaks. Section 2.1 will address why having to learn this many parts at once can be a problem. In normal Python there is no way to introduce concepts without immediately having to introduce every element of that concept, including syntax. This is where Hedy comes in.

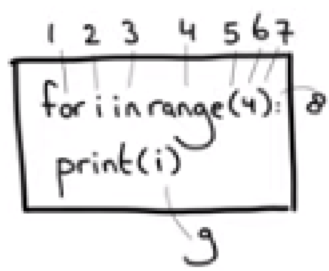

Figure 1: For loop in Python showing the different elements. *Credit: Hedy Introduction Talk by Felienne Hermans*

1.1 Hedy

Hedy [Her20] is what is called a "gradual programming language". It starts of simple, then grows more complex as it introduces new concepts. Hedy is split up in different levels [Hed], each with slightly different, more complex semantics and syntax. Its aim is to teach Python, not by introducing full concepts one at the time, but by using a simpler language to introduce a concept gradually. Take the print statement for example. In the first level of Hedy, there exists a print statement where everything after writing `print` is written straight to the output. So typing `print Hello World` will write `Hello World` on the screen. This level doesn't have much practical use for its programs, but it is a nice way of introducing the print statement as a concept. The next level introduces variables, but keeps them simple by simply letting the user type `name is Thera` to get a variable `name` which equates to `Thera`. By typing the following program, `Hello Thera` will be printed on screen:

```
name is Thera
print Hello name
```

Observant readers may have noticed that typing `print My name is name` will not result in the expected `My name is Thera`, instead outputting `My Thera is Thera`. This problem can be used to explain why Python uses quotation marks to differentiate between strings and variables. Coincidentally, next level this concept of using quotation marks to tell the computer which part to print as string and which part to use as variables is introduced. And so on and so on. By introducing new concepts in this gradual way, Hedy aims to teach Python in a way similar to how natural languages are taught. This way learning Python can be made easier and more effective. Hedy is still a work in progress project and this thesis aims to improve a small part of that project.

1.2 Thesis overview

This thesis will focus on two subjects: the implementation of the for loop in Hedy and the creation of a framework for sublevels in Hedy. Chapter 2 will explain some definitions regarding Hedy's philosophy and the subject of this thesis. Chapter 3 talks about other projects within Hedy, as well as projects surrounding programming language education. Chapter 4 will go into detail about the implementation of the for loop and the motive behind the creation of the sublevel framework. Lastly, chapter 5 will talk about the conclusions reached and future research.

2 Definitions

2.1 Cognitive Load

When learning about new concepts like a new language or, in the case of Hedy, new syntax ones brain can only hold so much new information at once. When trying to learn too many items at once, the brain becomes overloaded with new information and won't be able to process it anymore. This is called cognitive load. When learning a programming language like Python, cognitive load can become a problem as there are many new concepts that must be learned at once in order to create a program. [Her21] As said in section 1, there is no way to simplify normal Python code without breaking the program. Syntax has to be done correctly or the computer won't understand it. The reason Hedy is designed as a gradual programming language, is to avoid cognitive load by only introducing a small amount of new concepts at the same time and by building off of earlier learned concepts.

2.2 For loops

The for loop in Python consists of a bunch of different elements which all have to be learned and understood simultaneously in order to successfully implement it. There are no less than three keywords, one of which is actually a function with multiple possible parameters, a variable which only exists inside the for loop, a semicolon and the indentation. Learning about the for loop for the first time can be a daunting task for novice programmers because of cognitive load as mentioned in the previous section. Hedy attempts to take away the cognitive load in level 8 of the program by reducing the amount of new concepts to just the key words and a specified range. `for i in range(4):` becomes `for i in range 1 to 4`. By taking out the brackets and semicolon students don't need to worry about the complicated looking syntax yet and can just focus on the concept of the for loop itself. One of the subproblems of this thesis was attempting to simplify this for loop from Hedy even more. The reason for this simplification is that the for loop as Hedy presents it still introduces many new concepts. There are still three new keywords plus the range specification and the variable `i`. How this loop was simplified further is explained in section 4.

2.3 Sublevels

The level structure in Hedy prior to this thesis was linear. When a level is completed, one goes to the next level where new concepts are introduced. This thesis introduces the concept of sublevels to Hedy. A sublevel is a small level that can be used to ease users into a new subject even more gradually than using the normal levels. This concept came to be because the way Hedy previously introduced the for loop seemed a little too steep of a step. Instead of writing a whole level and throwing off the previously established flow, a framework to support sublevels was added to insert these types of extra levels. Currently only one exists, level 7-1, which was made for this thesis and helps introduce the for loop in a simpler way than done in level 8, see section 4. As a framework exists now for sublevels, new sublevels can be added later.

3 Related Work

Hedy is an open source project. As such, multiple projects are being worked on at the same time. New levels have been added, error messages have been improved and syntax highlighting has been added to name a few. One recent addition to Hedy are assignments. These assignments guide the user through the new concepts by giving them an idea to work with and a challenge to complete. These assignments span multiple levels, each building on the previous level like Hedy's levels build on each other. This way users get used to new concepts while learning to get creative with their programming in a fun way.

There are also several other educational oriented programming languages. One of the most popular ones is Scratch [Scr]. This language uses visuals instead of lines of code to make programming easier. By using blocks like the one in figure 2 Scratch is able to completely get rid of the complicated looking text based code. This makes it much easier for novice programmers to mess around with programming without having to deal with constant error messages. Thanks to this fact Scratch is used by people of all ages. Scratch's simple structure has no gradual learning to it in the way Hedy does, as every block is immediately available to the user, but it does have tutorials which introduce the blocks in a playful way to the user. The philosophy is similar to that of Hedy: avoid cognitive load (section 2.1) by simplifying abstract concepts like syntax and unknown keywords.

Figure 2: A repeat block used in Scratch

4 Implementation

4.1 Constructing the for loop

As stated in section 2, the for loop in Hedy still feels too complicated despite its simple design compared to the for loop in Python. This was the original reason for this thesis. By making a level in between the current levels 7 and 8 with a simpler version of the for loop, it could be introduced in a more gradual way. To achieve this, multiple different (educational) programming languages were explored to see how they introduced the for loop.
A lot of programming languages use a `repeat` statement instead of a for loop. `repeat 5` will repeat the loop body 5 times. Scratch [Scr], Logo [Log] and Quorum [Quo] are examples of languages which use repeat statements. Then there is Sonic Pi [Son], which uses a `times do` statement. `3.times do` repeats the body 3 times. Hedy also uses a repeat statement in its earlier levels before moving on to the actual for loop. Three languages which use a for loop with simple syntax that became the basis for the for loop designed for this thesis are Lua [Lua], which uses a statement of the form `for var = 1, 3 do`, PencilCode [Pen], which uses `for 1..3` and Snap! [Sna], which uses `for i = 1 to 3`.
The syntax of Snap!'s for loop is very interesting as it is very close to the Hedy for loop. But where Hedy uses `in range`, Snap! just uses an equal sign. At level 8 in Hedy equal signs have not been introduced yet, as syntax has been pushed as far back as possible. What Hedy users are familiar with though, is the variable assignment through the word `is`. As seen before in section 1.1, statements like `name is Thera` have been used since level 2. Using Snap!'s version of the for loop and changing the equal sign to the keyword `is` users are familiar with, the complicated for loop with four different new elements suddenly changes to a new concept with just two small changes. A new keyword `for` and the range specified by the keyword `is` and new keyword `to`. By also changing the variable from the abstract `i` to something more easily understood like `counter`, the for loop can be simplified even further. This became the basis for the new level which is now called level 7-1. Why this level is called 7-1 is explained in the next section.

4.2 Sublevel framework

While designing the simpler for loop explained in the last section, work on Hedy continued. Level 8 and up were being implemented and it became very hard to just insert a level into the structure, especially a level that had not been tested on its importance, see section 5.2 for further explanation on why this has not been done. Instead, the idea of sublevels was born. As explained in section 2.3, sublevels are small levels in bewteen the actual levels of Hedy which can be used to introduce new concepts even more gradually than Hedy already does. Originally, the idea was to use sublevels to test out where smaller steps might be necessary, but this idea has slowly evolved into a concept where students can opt to use a sublevel when they need more explanation on a concept or need more time to learn concepts. This way students who are faster can skip the small steps and immediately move on to the full concept, while students who need more explanations can use the sublevels for a more gradual approach. Sublevel 7-1, the simple for loop explained last section, is currently the only sublevel. As the sublevel framework is still new, currently the only way to access it is by knowing the specific URL [Lev]. See also section 5.3 on future developments.

4.3 Grammars

Every level in Hedy has its own grammar and syntax. These grammars are stored in Hedy's backend and called upon whenever a user wants to run their code. A transpiler, in Hedy's case Lark [Lar], which is a program that can parse context free grammars turns the code into a tree which can then be converted into Python code. In order to implement new (sub)levels, a new grammar has to be written as well as a translation from that grammar to Python. As levels build off of each other, not every function has to be implemented for every level. Hedy is built in such a way that functions from previous levels that remain unchanged this particular level, like the print statement after level 3, use the grammar and translation to Python from the previous level. To implement level 7-1 this practice was used as well. Level 7-1 only introduces the for loop and removes the repeat statement, so those were the only two changes made for the grammar of level 7-1. The grammar for the for loop looks something like this:

```
for_loop: "for " (NAME | var) " is " (NUMBER | var) " to " (NUMBER | var) _EOL
(" "+ command) (_EOL " "+ command)*
```

`for_loop` is the name of the statement. Everything between quotation marks is taken by the transpiler as strings. When parsing a line of code, the transpiler looks for an exact match to these strings. `(NAME | var)` and `(NUMBER | var)` are variables. The transpiler checks if the line of code to be parsed has a variable or a number in those spots. These are also the arguments that help Hedy turn Hedy level code into Python code. The arguments from this for loop are put into a Python for loop while the quotation mark parts are ignored. `_EOL` is a keyword for the end of a line and `command` is a statement which means everything classified as a "command" can be at that spot. In simple terms, the `for_loop` statement tells the transpiler to look for a statement looking like:

`for` *variable* `is` *number or variable* `to` *number or variable*
    some command
    possibly more commands

As the only difference between level 7-1 and level 8 is the way the for loop is written, the grammar and translation to Python are very similar. Level 7-1 even uses the same Python translation as level 8, as the arguments fall in the same spots.

5 Conclusions and Further Research

5.1 Conclusions

In conclusion, the sublevel framework will be useful for future Hedy projects. Level 7-1, though not yet tested, can now be used to teach novice programmers who need a little extra help the for loop in an even simpler way that Hedy could. The sublevel framework can be used to create new sublevels where necessary, though currently they can only be accessed by knowing the specific URL. Hedy is still in development and new features are added often. The creation of sublevels was yet another step forwards in Hedy's development.

5.2 Discussion

While designing the sublevels the original plan was to make two different sublevels, one which ended up becoming 7-1 and another called 7-2 which was the same as level 8. A new level 8 would be designed as a continuation of either one of the sublevels. This ended up forming a problem however, as the sublevels were not readily available. This meant users would go from level 7 to the new level 8 without going through the sublevels first, leaving them confused. This prompted the removal of sublevel 7-2.

The problem of whether or not users have gone through a sublevel or not extends to the current version of Hedy too. Hedy introduces its levels with an explanation of the new concept. When a user has gone through a sublevel however, they may already be familiar with the concept so the intro text would end up explaining the wrong thing to them. Level 7-1 and level 8 struggle with this very concept, see appendix A.

Because of the current state of the world actually doing experiments with the new for loop and sublevels was sadly impossible. There is however support for A-B testing in Hedy, so these tests could still be done at a later date.

5.3 Further Research

In the future new sublevels could be added to give students more options to get extra help with new concepts. As sublevels are currently not accessible without knowing the specific URL, another future development would be to implement a button to access the level (and any other sublevels) from Hedy itself. This is currently being worked on.

As the importance of level 7-1 has not been tested yet, A-B tests could be performed where one group learns about for loops using the sublevel and another by just following Hedy normally. By testing to see how many mistakes are made in later levels when students use the for loop again, or by measuring the dropout rate during level 8, the impact of level 7-1 can be tested. When more sublevels are added, the same can be done for those levels.

Appendix A highlights another problem that could and should be solved at some point in the future, the flow of the introduction texts. These texts should build off of earlier acquired knowledge, but when sublevels come into play that knowledge may be different depending on the user.

References

[Hed] Hedy website. hedycode.com.

[Her20] Felienne Hermans. Hedy: A gradual language for programming education. 2020.

[Her21] Felienne Hermans. *The Programmer's Brain: What every programmer needs to know about cognition*. 2021.

[Lar] Lark. https://lark-parser.readthedocs.io/en/latest/index.html.

[Lev] Hedy level 7-1. https://hedycode.com/hedy/7-1.

[Log] Logo. https://el.media.mit.edu/logo-foundation/what_is_logo/logo_programming.html.

[Lua] Lua. http://www.lua.org/.

[Pen] Pencilcode. https://pencilcode.net/.

[Quo] Quorum. https://quorumlanguage.com/.

[Scr] Scratch. https://scratch.mit.edu/.

[Sna] Snap! https://snap.berkeley.edu/.

[Son] Sonicpi. https://sonic-pi.net/.

A Sublevels and problems with flow

Level 7-1, figure 3, introduces the concept of the for loop to Hedy users using the way described in section 4. In the introduction text, figure 4, it tells the user the keyword `repeat` they learned about before has been changed to a for loop. This way the user can take their knowledge about the repeat statement and apply it to this new concept of a for loop. When they complete level 7-1 and move on to level 8 however, they are greeted by the same introductory message, figures 5 and 6, just slightly changed to include the differences between the two levels. For users going normally from level 7 to level 8 the intro text makes sense, but when coming from a sublevel one would expect the text to continue with the knowledge that the sublevel has been completed. Users may even think level 8 is a duplicate of level 7-1 and skip it all together, missing out on the new syntax and subsequently getting confused in later levels when their learned syntax doesn't work. This problem will likely also exist when new sublevels are added.

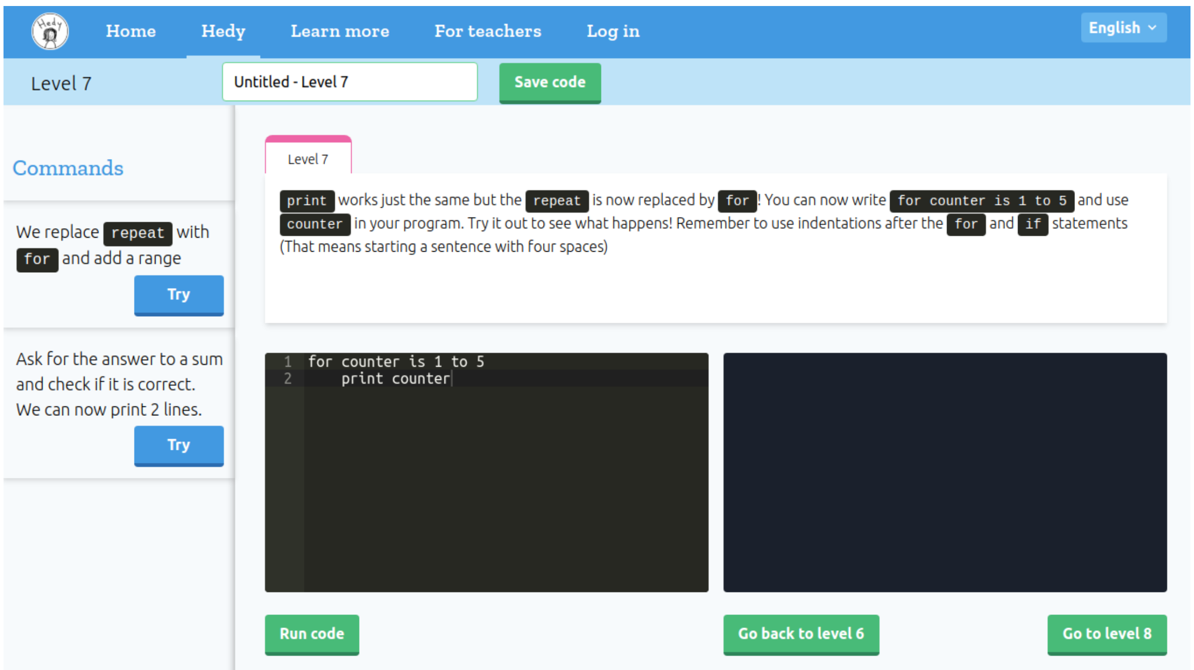

Figure 3: Hedy level 7-1. Currently the layout still says level 7. This will be changed at a later date

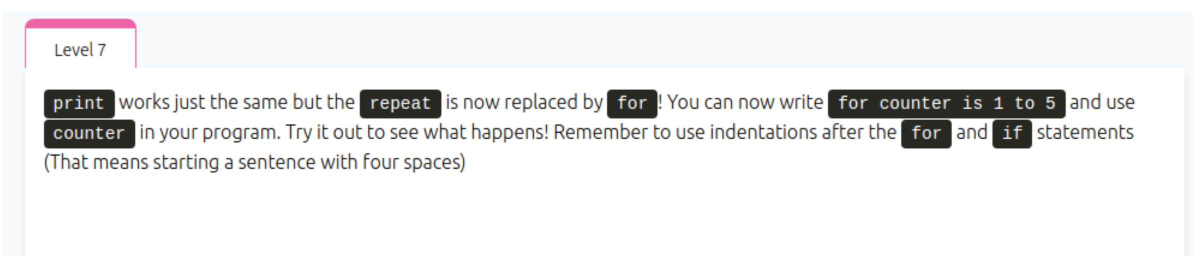

Figure 4: Closeup of the intro text of level 7-1

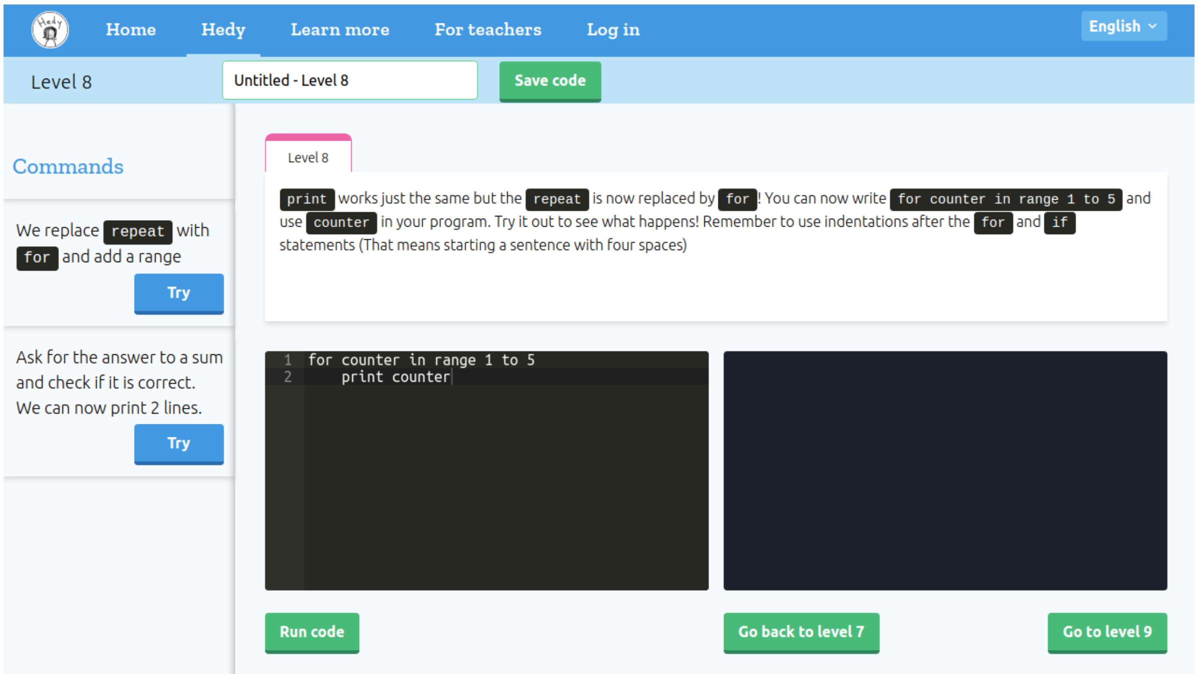

Figure 5: Hedy level 8

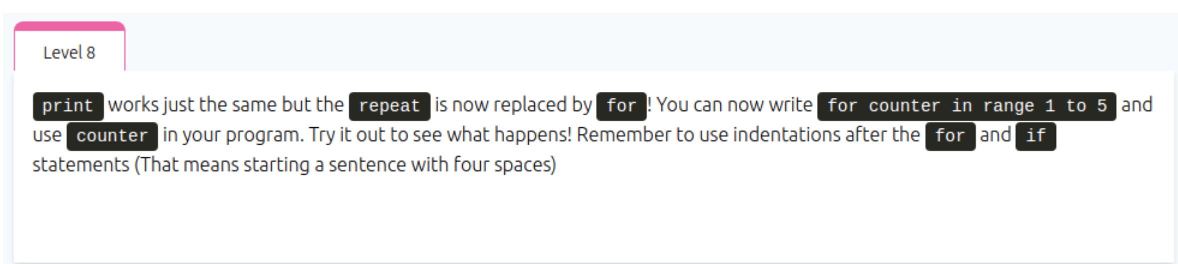

Figure 6: Closeup of the intro text of level 8