4-26-2017

Comparison of visual programming and hybrid programming environments in transferring programming skills

Hussein Alrubaye

R·I·T

Comparison of visual programming and hybrid programming environments in transferring programming skills

by

Hussein Alrubaye

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Software Engineering

Department of Software Engineering
Golisano College of Computing and Information Sciences

Rochester Institute of Technology
Rochester, NY

Date (April 26, 2017)

Committee Approval:

Dr. Stephanie Ludi **Date**
Professor

J Scott Hawker **Date**
SE Graduate Program Director

Mohamed Wiem Mkaouer **Date**
Assistant Professor

Abstract

Teaching students programming skills at an early age is one of the most important aspects for researchers in recent decades. It may seem more practical to leverage an existing reservoir of knowledge by extending the block-based environment; which uses blocks to build apps towards text-based. Text-based is text code only, rather than starting to learn a whole new programming language [7]. To simplify the learning process, there is a new coding environment that's been introduced named block-based environment (Pencil Code, Scratch, App inventor) that is used by millions of students.

There are some challenges teachers are facing to bring the text-based environment to the classroom. One is block-based tools do not allow students to write real-world programs [1], which limit the student's abilities to writing only simple programs. Also, there is a big gap between the block and text-based environments. When students want to transfer from block-based to text-based, they feel that they are in a totally new environment [1]. Since block-code transition involves movement between different code styles and code representations with different syntax [7]. They move from commands with nice shapes and colors to new environments with only commands, also they have to memorize all the commands and learn the programming syntax.

We want to bridge the gap between the block-based and text-based by developing a new environment named hybrid-based, that allows the student to drag and drop block code and see real code instead of seeing it blocks only.

The study was done on 18 students, by dividing them into two groups. One group used block-based, and another group used hybrid-based, then both groups learned to write code with text-based. We found that hybrid-based environments are better than block-based environments in transferring programming skills to text-based because hybrid-based enhances the students' abilities to learn programming foundations, code modification, memorizing commands, and syntax error comparison with block-based. Also hybrid-based reduces the learning shock [37] that students feel when they learn programming with block-based then they move to text-based because students who are using hybrid-based development environment are interacting directly with code, while block-based groups have never seen the code before.

Acknowledgements

One of my most important goals when conducting research is to make it as easy as possible to teach programming to students. I am very grateful for the support of my mother and my wife Rana during this whole process. They have been extremely helpful getting me to this point. Lastly, I also want to thank my daughters, Jena and Laya, who can sometimes be a distraction with their constant crying and noise and at the same time be an inspiration for me to succeed. They are constant reminders that I need to be the best that I can be and provide for their future.

Table of contents

List of Figures

List of Tables

Chapter 1

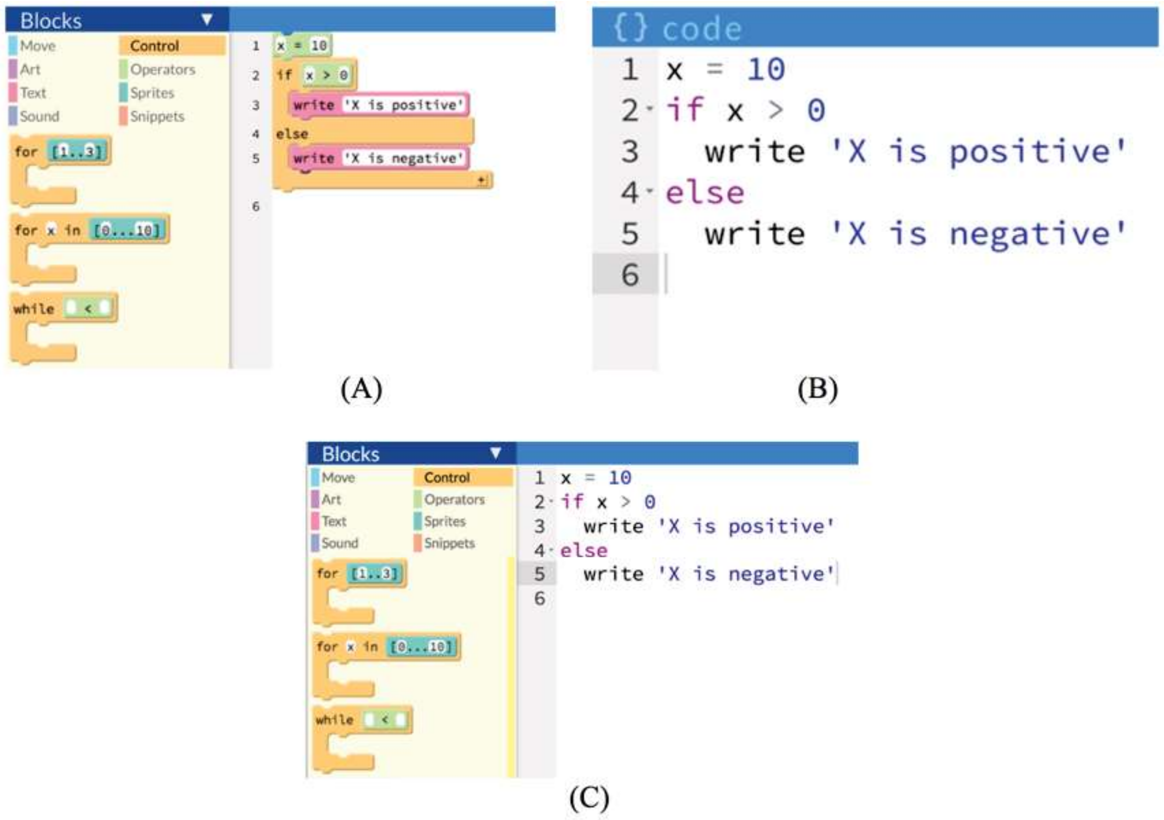
Introduction

I. Introduction

Teaching students programming in early age is one of most important aspect of the researchers and governments with new technologies that make the computer used in every sphere of our life such as Home, School, Shopping, and Medical. It is really very important for everyone to start learning how to code. There are many countries start teaching programming for the students in early age and encourage them to start coding using different tools like (Pencil Code, Scratch, Patch, Block, app inventor) and programs like Hour of code. And there program and tools which are being supported by big companies and organizations like pencil Code are supported by Google.

Since it is hard for people with no computer background experiences to start directly writing code and learn programming logic and syntax in writing code with the text-based environment like Java, C#, Python, Ruby on Rails, etc. We have to find a way that will help to make learning programming easy and fun for students, and find a way that prepares and engages students with no- programming experiences to start to learn programming and write code.

It might seem more practical to leverage that existing reservoir of knowledge by extending block-based (Figure (1.1a)) towards text-based, rather than start learning a wholly new programming language [7]. So there is new coding environment which has to be introduced namely block-based environment like (Pencil Code, Scratch, Patch, Block, and app inventor). This environment is welcomed by millions of new students to program. App inventor is used by 2.7 million users [4]. Scratches have more than 16 million users [5]. Block tutorial on code.org has been reaching over 359 million users [6].This environment produces new way in writing code that will reduce learning shock [37] to the students may feel when the start direct to learn to program with text-based. Block-based environment use blocks that represent programming commands with nice color and shape to make it easy to accessible, memorize and use to engage students to start programming with this environment and writing a successful program by drag-and-drop blocks [2]. So, there are many tools that have been published to make this process easy like Block-based programming. Block-based programming helps students to avoid programming syntax errors, and reduce the memorize programming command. With the text-based environment, students just need to drag-and-drop programming commands like (If, Loop, Variables) to build and run his app. So there is no need to memorize the commands. But these tools do not enable users to work directly with mainstream programming languages [15], they could use blocks only to write the program.



```
x = 10
if x > 0
  write 'X is positive'
else
  write 'X is negative'
```

```
x = 10
if x > 0
  write 'X is positive'
else
  write 'X is negative'
```

```
x = 10
if x > 0
  write 'X is positive'
else
  write 'X is negative'
```

Figure 1.1. Comparable blocks-based (A), text-based (B), and hybrid-based (C) programs

The tools that we use, in our case, are the development environments have an impact on how and what we think. Dijkstra said that “The tools we use have a profound (and devious!) Influence on our thinking habits, and, therefore, on our thinking abilities.” [26]. How the use of particular symbols system effect on students’ conceptualization? has been studied in different environments. Sherin [34] studied the learning of physics fundamental using either programming language or algebraic notation. He found that students who learned in different environments have different affordances in learning physics fundamental. He said that “Algebra physics trains students to seek out equilibrium in the world. Programming encourages students to look for time-varying phenomena and supports certain types of causal explanations, as well as the segmenting of the world into processes” [34]. Similar works have investigated on the relationships between different environments representations and affordances such as [35, 36] and they found that different representations have different affordances on people.

Weintrop [1] compared the effects of block-based; hybrid- based and text-based on transfer programming skills using Pencil Code [10]. The study was done by dividing the classroom into three sections. The first section used block-based, the second section used text-based, and third used hybrid-based (block /text) All students were learning same curriculum (ex, variables, loops, and conditions) for 5 weeks. In week 6 all students moved to text-based with Java. He found that (92%) of students said that block-based is easier to learn to program than text-based. Because it is easy to drag-drop and fewer memories commands also students had a better attitude to understand the loops, variables than text-based students. However, the students find drawback in block-based that is difficult to build a large complex program. The drawback of this paper is that the hybrid group did not have their own tool; they were switching between code and block which make it difficult for students to track the representation of blocks in the code when the program grows, so they did not get fair learning comparison. They have to have their own tool. In our study; we will implement a specific tool for a hybrid team with specific features that have block and text-based features to make it as an easy bridge to transfer to text-based.

Robinson [7] did study on Scratch that is one of a most popular block-based development environment powered by MIT. One of an interesting feature in Scratch that is simple to understand and use and avoid the learner the syntax errors. While Scratch focuses on to learn the programming logic, not programming languages, so the student does not learn how to build a program, instead, he learns how to think logically [7]. The drawback of this paper is that when student transfer from Scratch to the text-based environment, he does not have any programming background. In our study; we will implement hybrid-based with abilities to use real programming commands like Loop, Variables, and Control structure.

There were some challenges for teachers that were faced to bring the text-based environment to the classroom. One was block-based tools cannot allow students to write the real-world program [1] which limit the student's abilities and thinking to write only simple programs. Also, there is a big gap between the block and the text-based environment. When students want to transfer block-based to text-based, they feel that they are totally in a new environment [1]. Since block-code transition involves movement between different code style and code representation with different syntax [7]. They move from commands with nice shapes and color to a new environment with only commands and they have to memorize all commands and learn to program with syntax.

We want to bridge the gap between the Block-Based and Text-based by developing a new environment named Hybrid-Based (Figure (1.1c)) and study the effect of Hybrid and block-based environment on transfer programming skills when a student starts with Text-based. By implementing these two environments and studying their effect on transfer student skills, we will be able to learn which tool is better for us.

We selected Pencil Code as the Block-based environment, and we download the open source project for Pencil Code from Github, then we update the project to produce new Hybrid-Based environment. In Hybrid-Based when students drag-and-drop the commands, they will see it as code in the editor. While in block-based when students drag-and-drop the command they will see it as a block. Figure1.1 show how the final code will be written in block-based and hybrid-based environments.

We have selected to work with Pencil Code, because "The goal of Pencil Code is to build enough confidence in beginning programmers so that they can create programs without Pencil Code" [3]. Also, it allows a beginner to achieve satisfaction results quickly with minimizing the frustration. Furthermore; it introduces the beginners directly to programming system that is used by professional by allowing them to toggle between text-based and block-based [3]. Pencil Code allows students to write real CoffeeScript program with blocks only. As discussions regarding pseudo - code [8] are helpful in distinguishing between structural and functional elements of language. It will be good for our study because it allows students to write real CoffeeScript program using blocks. In addition; CoffeeScript is similar to Python, so it will open the way to learn a simple and powerful language like python easily. Also, the project is open source.

However, allowing students to see text translation for their blocks did not help enough to understand what every block is represented in text-based. Think about converting ten blocks to code, it is hard for students to keep tracking what every block mean in code. For that reason, we come up with the hybrid-based environment.

II. Research Questions

Finally, we want to answer this research question:

Q1: Is Hybrid-based allows the students to understand and identify programming concept (Loop, Variables, Control structure) easier than Block-based when they start with Text-based programming?

Q2: Will the Hybrid-based environment reduce the misunderstanding (learning shock) that the student may feel when transferring from block-based to text-based?

Chapter 2

Literature review

Literature review

A large number of researchers and organizations are involved in educating the programming and focus on challenges that face the beginner's when they start to learn programming concepts. Most of the study focused on the design of languages and the programming environment that learner will use when he involves in learning programming and the relationships between the representations of the languages and the students' abilities to understanding and his motivation to learn to code.

In this chapter; we will discuss the relationship between the computer and learning. We will start by giving you a brief review on the computer and programming languages and why this area is interested. Then we will discuss different types of programming environments, and the challenges that are faced by the students when they start to learn with a specific environment. While we go through every environment, we will continue with a review of the literature on the relationship between the graphical representation and learning. We will discuss the benefits and drawbacks of every development environment and how that could effect on the beginners when they start to learn to program.

We will review the historical design of the programming languages, and how researcher focused on language representations to enhance the learning and make it easy to understand, engagement, and easy to remember. We will focus a lot on visual programming, and how graphical representation in programming languages had a high effect on motivating students to start to learn to program.

There are various studies that have to be done to evaluate visual programming effects on learning programming, and most of the researchers agree that visual programming helps a lot to motivate the beginners to start to write the first app. However visual programming tools have some limitations today that have a drawback on the students in building a complete app and write correct code syntax that we will discuss in details.

I. **Computers and Learning**

"The early history of the computer, the utility for teaching programming tools was the faculty member who had experiences with programming. The teacher was using their experiences to teach programming class. At that time programming was new technology in the area that needs more studies and evaluation to find the best way to teach the students programming. Teacher cannot start direct by teach the beginners programming, because they do not have any background on how program and process works, and communicate that make it hard for teacher to teach students and identify the prerequisite for programming class, and how they could find easy and simple way to teach students with motivation and engagement.

In the early 1960s, Alan Perlis, the director of the Computation Center at what was then called the Carnegie Institute of Technology, argued for the inclusion of programming in the curriculum for all learners as part of as working with computers would help develop well-rounded students who would be ready for whatever challenges the world put before them [12]. Perlis envisioned students writing programs to solve real world problems, in doing so, he argued they would improve their abilities to abstract, organize, plan, and use information from diffuse and abstract environments. While Perlis was thinking about programming as a pedagogical strategy at the undergraduate level, others saw the potential of learning as a powerful learning strategy for younger learners. Papert and his colleagues at MIT and BBN Labs developed the Logo programming language as a way to allow younger learners to engage in programming and the various met cognitive practices that accompany it [13].

In foreground the act of authorship and construction of programming, this type of learning experience, which Papert called "Constructionism", also granted autonomy to learners, using the computer as a medium for creativity and personal expression. The view of the computer as an expressive and powerful medium for learning has shaped decades of designs of computational learning environments [18]. The type of learning enabled by computers and advocated by Papert and colleagues went beyond just the act of programming to include other aspects of the potential of a computer for learning, which will be returned to below" [2].

II. **Learning Context**

“As technology pervades our daily lives, there are growing opportunities to have learners engage with ideas from computer science across diverse contexts both in formal and informal contexts. A growing body of work is looking at ways to teach computing in informal settings, be they at home or in shared communal spaces like libraries or museums. At the same time, new initiatives to bringing computer science into formal spaces are also underway. In many ways, these two approaches complement each other and have different goals and contain a different set of features that can draw diverse populations of learners into the field.

In the formal space, long-standing courses like AP Computer Science are being supplemented with new curricula like the AP Computer Science Principles course [20], and other curricula leveraging new technologies and programming environments like the Exploring Computer Science [21] project and a Taste of Computing
course [22], Formal learning opportunities provide infrastructure, access to teachers and extended amounts of time for learners to engage with core ideas in computing, but are often limited by resources available to schools and other constraints that come from being situated inside existing educational infrastructure. Informal learning environments provide a much more open canvas upon which to create learning opportunities and allow for types of engagement that are not possible with the constraints of schools. Projects like the Computer Clubhouse initiative [23] give learners an open space for learners to pursue projects of their own interest, grant greater flexibility to learners in terms of the types of activities they engage in, and are not subject to the same constraints or expectations that accompany school-based learning. Other after school projects offer more structure, but still take advantage of the freedom that accompanies informal learning, such as the FUSE project [24]” [2].

III. Programming Languages and Environments

The decision of selecting program language to use and when environment learners will program in has a long been a subject of vigorous debate in the computing education research Community [25]. The design of programming languages environment and how the design of development environment will affect on students in learning this is closely related to the question that is being pursued in this thesis.

We are really interested to find an environment that makes the learning of programming easy and simple, so we need to look back and see what people did before us to make the learning of programming simple and easy. We will discuss in details in next sections. In this chapter on some types of programming environments that used to leverage the learning of programming such as the graphical environments and how these environments engage the students to learn and what is the drawback that we identify to fix. This chapter includes discussion of textual versus a graphical approach to programming interfaces as well as the design of the languages and environment.

IV. Representations and Learning

“The tools that we use have a profound (and devious!) Influence on our thinking habits, and, therefore, on our thinking abilities [26]. The tools that we used in our case the development environment and programming languages have profound and unexpected effects on how and what we think. These symbolic systems provide a representational infrastructure upon on our knowledge is built and communicated [27]. By adopting these perspectives, we will understand the relationships between growing the family of programming representation and understandings and the practices they publish. Trying to write C++ program with Notepad++ editor is much different from writing C++ program in Eclipse editor. The second one provides auto-complete and gives hints on syntax error these guides will help a lot to guide the developer to write a clear program. So when we build our own tool we have to think of these points and how to make the environment development easy and attractive.

V. **Text-based programming**

Text-based programming languages allow a developer to build apps using text code by typing code in the editor then run the code to build an app that does a specific function. Along with different programming languages and tools, there are also bigger picture differences around types of languages and programming paradigms. There are too many languages and frameworks have been introduced. With these languages; it is really hard for a learner to select which language or frameworks to start to learn with, and which languages is easy to start coding. In this section, we will discuss some of these languages.

1- C++

In 1979 Bjarne Stroustrup, a Danish computer scientist begins work with C languages with classes to produce C++. The motivation for creating new language was the works that he has to do for his Ph.D. thesis. Programming languages like C++ are a really hard language for beginners to start with. It has hard syntax with semicolon that make it hard for beginner to run the first app, and C++ use pointers to save data in memory location for temporarily which make it really hard for beginner to start to direct with C++, because the developer need to manage the use of memory and object lifetime, otherwise he will cause to memory leak that leads to delay in the program.

The question is how the beginner could manage the memory while he still did not know programming syntax and program workflow that make this language as hard chose to start with.

2- Java

In 1991, James Gosling, Mike Sheridan, and Patrick Naughton initiated the Java language project. When Java is introduced to make the life easy for the developer to manage and work with memory and object life. Java tried to take the management of using the memory from the developer and move it to the compiler jobs by using garbage collection to kill any object that no longer used. However add garbage collection in Java is great, but they still did not target the beginners in this update, because the beginner has to learn how to code before learning and how to manage the memory.

3- Python

In 1989, Python implemented by Guido van Rossum at Centrum Wiskunde & Informatics. Python tried to simplify the learning process by making the programming syntax easy with spaces and adding functions that do powerful works with limits line of code. These features allow Python to be on the top of programming languages in few years because it shows to be really easy for a beginner to start their first app with Python.

4- PHP and Laravel

The framework is a new feature that is used to minimize the developer works and make the development process much easy if we still need the language and we want to make it simple. For example, PHP is one of most powerful language that used in web development. This language needs to much-repeated works from the developers that lead to building framework to make this process easy and simple like Laravel framework that is open source PHP framework created by Taylor Otwell and intended for the development of web applications using the pattern of Model-View-Controller (MVC). This framework helps a lot to manage the web development with PHP and make it much simple that before, with Laravel single line of could use to generate work that needs a day for a developer to do it by coding with normal PHP.

As you see all text programming and frameworks focus on making the development process easy but they never target the total beginner. I cannot start directly with Laravel without having a background in PHP. So even Laravel made the development process easy, but still, the learner has to start with PHP first.

"For that reason, the researchers though continuing to continue with developing the programming languages and frameworks is not enough for totally beginners to start to learn. They have to use tools that much simple and easy to learn like visual programming tool. For that reason, visual programming comes to the area and many studied start learning with visual programming tools because they see it easy and simple for startups, we will discuss these tools in next section.

Along with recognizing that features of the language can support or hinder learnability, it was realized that software used to author programs (called Integrated Development

Environments or IDEs) themselves could provide a large number of supports the help the novice overcome challenges including syntax errors, deciphering compilation errors, and problematic sections of programs. This recognition coincided with a larger shift in the computing space that was advocating for a shift away from users conforming to computer requirements towards the world where the computer confirmed to the user [28]. These efforts
initially focused on supporting experts, but educational technology designers soon realized that what is good for the expert is often not the same as what is best for the novice and when designing educational tools, you should proceed with the learner in mind [29]. As such, a growing number of IDEs have been developed explicitly with novice programmers in mind

An early and influential development environment designed to facility novices specifically, through a reduction in potential syntax errors was the Cornell Program Synthesizer, which built on the fact that programs are not flat text, but instead "hierarchical compositions of computational structures and should be edited, executed, and debugged in an environment that consistently acknowledges and reinforces this viewpoint" [30].
The Cornell Program Synthesizer provided users with templates that followed the syntactic structure of the language and thus when filled in, would result in invalid statements that could be added to the program. These templates were constructed by following the grammar defined by the language's abstract syntax tree (AST). While a number of different projects and research groups followed the lead set by the Cornell Program Synthesizer, Carnegie Mellon University emerged as a leader in the development of programming tools that used features of the language to support novices" [2].

VI. Visual Programming

As the development of programming languages and environments evolved, it was found that shifting from an all-text representation of programs to an approach that leverages spatial and graphical features could be productive for learning and understanding [31]. Collected under the label “Visual Programming”, these environments are broadly defined as “any system that allows the user to specify a program in a two (or more) dimensional fashion” [32]. While this definition is intentionally broad, it excludes text-based
programming (which is considered to constrain composition of programs to a single, horizontal dimension), software used to produce visualizations (like animation and drawing programs), and tools that visually depict the execution of programs (like environments that visually render memory contents or algorithmic flow of a running program). To provide a framework for evaluating visual programming environments, Green and Petre [33] developed a cognitive dimension’s framework that characterizes features of these environments and maps out the tradeoffs that exist between different visual design choices. These cognitive dimensions include Abstraction Gradient (various granularities of abstraction supported), Consistency (how formulaic is the language), Progressive Evaluation (what feedback is available from partially complete programs), and Visibility (how easy is it see and read the code) among others. This framework proved to be productive and is widely used to evaluate visual programming environments. Based on the promise of easier to learn, easier to use, programming systems, many visual programming languages have been designed and evaluated [2].

In this thesis, we are fundamentally investing in the relationship between the programming representation and learning. We begin our literature review by exploring the representation infrastructure of most common systems that used for teaching the beginners programming (Pencil Code, Scratch, and App Inventor). We selected to study these systems because many students are preferred to learn with these environments, and in few years the users of these platforms will grow to millions which make it is really an interested area to start investing on.

VII. **Blocks-based Programming**

The block-based approach of visual programming has become widespread in recent years and there are millions of students who started learning to program with different reasons: learner found that it is easier to learn to program than text-based. Because it is easy to drag-drop and fewer memories commands also they have a better attitude to understand the loops, variables than text-based students. However, the student finds the drawback for block-based that is difficult to build large complex program [1].

There are a number of development environments that use block-based which is Pencil Code, Scratch, Patch, Block, and app inventor. This environment is welcomed by millions of new students to program. App inventor used by 2.7 million users [4]. Scratch has more than 16 million users [5]. Block tutorial on code.org has been reaching over 359 millions [6]. This environment produces new way in writing code that will reduce learning shock that students may feel when they start directly to learn to program with text-based. Block-based environment use blocks that represent programming commands with nice color and shape to make it easy to accessible, memorize and use. To engage students to start programming with this environment and writing a successful program by drag-and-drop blocks [2]. Therefore, many tools have been published to make this process easy like Block-based programming. Block-based helps students to avoid programming syntax error, and reduce the memorize of programming commands. With the text-based environment, students just need to drag-and-drop programming commands like (If, Loop, Variables) to build and run his app. So there is no need to memorize the commands. But these tools do not enable users to work directly with mainstream programming languages [15], they could use blocks only to write the program. We will discuss the features and drawbacks of block-based tools.

1- Scratch

Scratch that can be seen in Figure 2.1 is one of the most popular block-based development environment that powered by MIT. Now Scratch has more than 16 million users [5]. One of an interesting feature in Scratch that is simple to understand and use and Scratch avoids the learner syntax error. While Scratch focuses on to learn the programming logic, not programming languages so the students will not learn how to build a program, instead he will learn how to think logically [7]. So when he goes to text-based environment he does not have any programming background

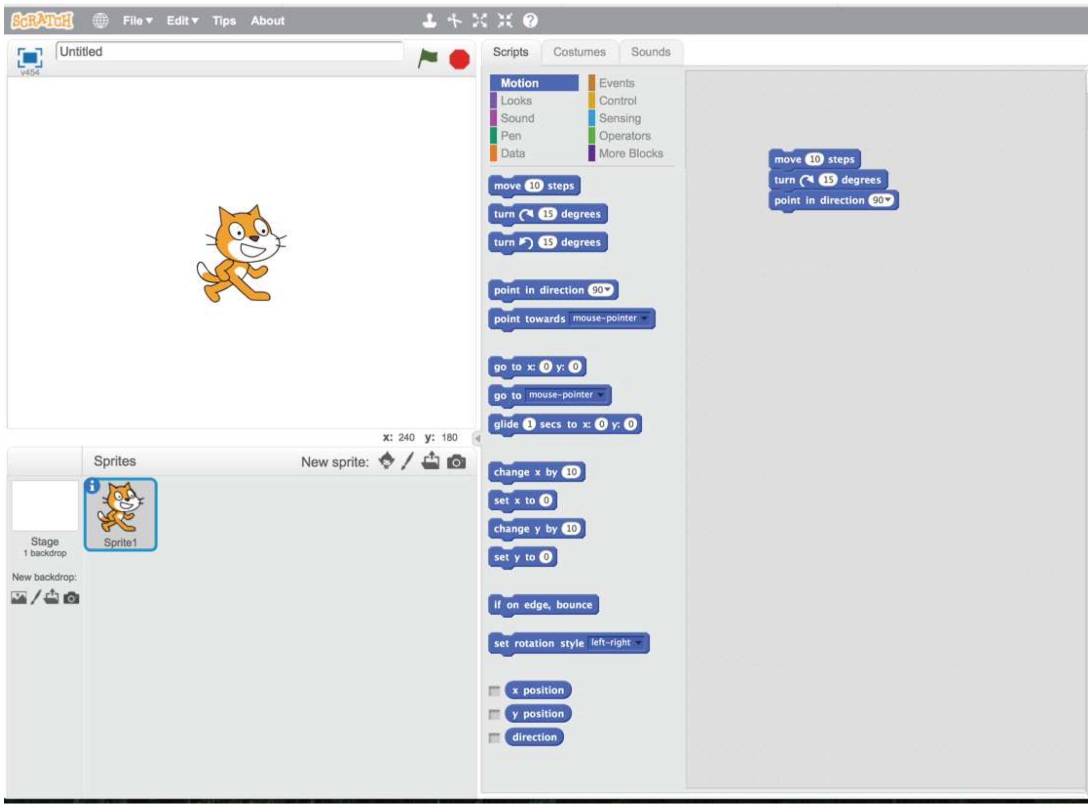
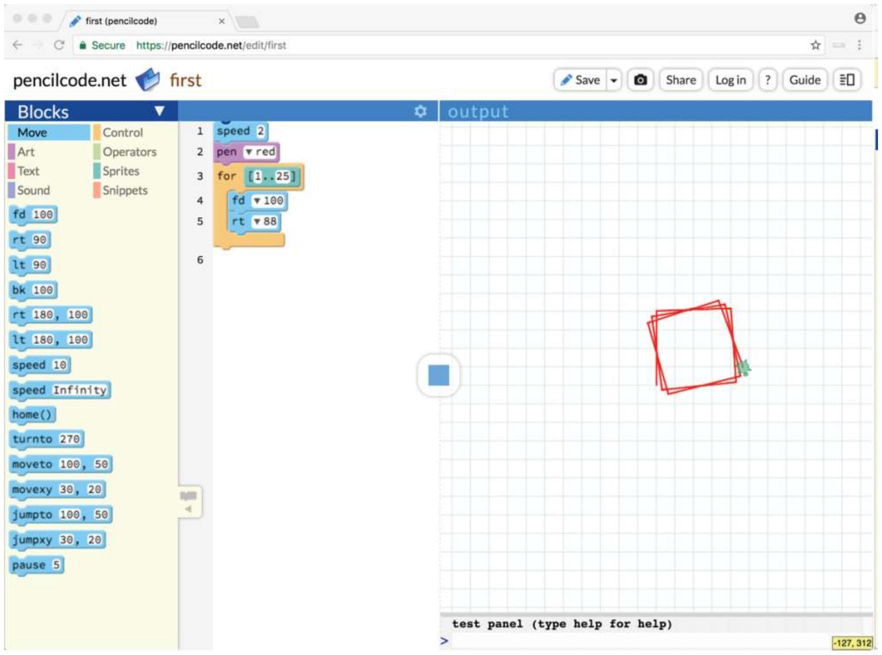


Figure 2.1. Scratch

2- Pencil Code:

Pencil Code that can be seen in Figure 2.2 is another tool that is used for the block-based development environment. “The goal of Pencil Code is to build enough confidence in beginning programmers so that they can create programs without Pencil Code” [3]. Also, it allows a beginner to achieve satisfying results quickly with minimizing frustration. Furthermore; it introduces the beginners directly to programming system that is used by professional by allowing them to toggle between text-based and block-based [3]. Pencil Code allows students to write real CoffeeScript program with blocks only. As discussions regarding pseudo - code [8] are helpful in distinguishing between structural and functional elements of language. It will be good for our study because it allows students to write real CoffeeScript program using blocks. In addition; CoffeeScript is similar to Python, so it will open the way to learn a simple and powerful language like python easily. Also, the project has an open source.



```
speed 2
pen red
for [1..25]
  fd 100
  rt 88
```

```
test panel (type help for help)
>
```

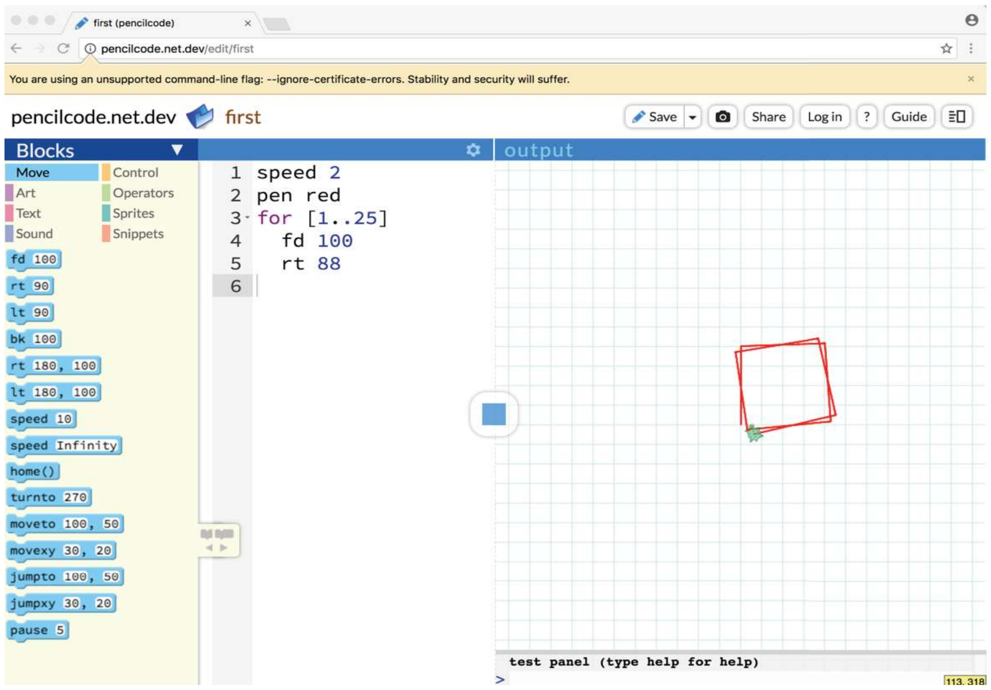
Figure 2.2. Pencil Code

VIII- From Blocks-based to Text-based Programming

Block-based develops environment like Pencil Code allows switching between the code and block to enable the developer to see how the block could be implemented in the code. However, allowing students to see text translation for their blocks did not help enough to understand what every block is represented in text-based. Think about converting blocks to code, it is hard for students to keep tracking what every block mean in code. For that reason, we come up with the hybrid-based environment.

IX- Hybrid-based Programming

To cover the gap between the Block-Based and Text-based we develop a new environment named Hybrid-Based that can be seen in Figure 2.3. Hybrid-based is a new development environment that allows the student to see the organic CoffeeScript code for any block after they drag-drop the block to the development environment. We will discuss this tool in details in next chapter.



You are using an unsupported command-line flag: --ignore-certificate-errors. Stability and security will suffer.

```
speed 2
pen red
for [1..25]
  fd 100
  rt 88
```

```
test panel (type help for help)
>
```

Figure 2.3. Hybrid-based, code example of move tortoise

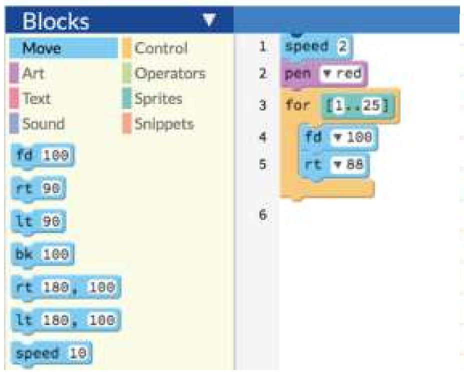
Chapter 3

Design and Implementation

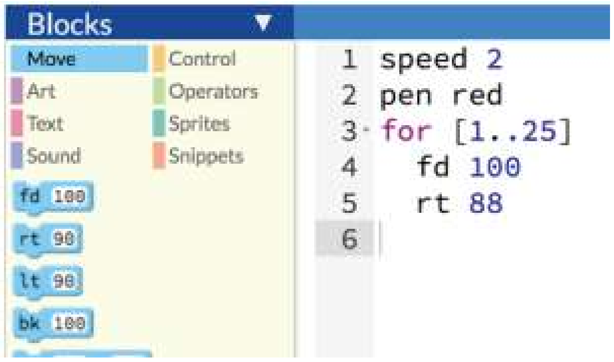
Design and Implementation

I. Introduction

In this chapter, we will discuss the design and implementation of a new development environment that is hybrid-based [11]. The hybrid-based environment is a coding development tool that we develop to allow students with no- programming experiences to start writing their apps in CoffeeScript programming language easily, by getting benefits from drag-drop blocks, see Figure(3.1b).



```
speed 2
pen red
for [1..25]
  fd 100
  rt 88
```



```
speed 2
pen red
for [1..25]
  fd 100
  rt 88
```

Figure(A) Figure(B)

Figure 3.1. Block-based (A) vs. hybrid-based (B) environment design

Our final system design should change Pencil Code design from Figure (3.1a) to Figure (3.1b). We have selected Pencil Code as the Block-based environment, and we download the open source project for Pencil Code from Github [16], then we updated the project to produce new Hybrid-Based environment. In Hybrid-Based Programming when students drag-and-drop the commands, he will see it as code in the editor. While in block-based when students drag-and-drop the command he will see it as a block.

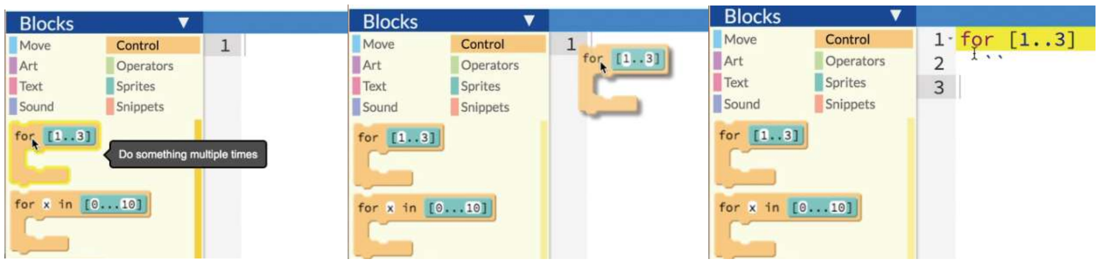
We select to work with Pencil Code, because "The goal of Pencil Code is to build enough confidence in beginning programmers so that they can create programs without Pencil Code" [3]. Also, it allows a beginner to achieve satisfaction results quickly with minimizing the frustration. Furthermore; it introduces the beginners directly to programming system that is used by professional by allowing them to toggle between text-based and block-based [3].

Pencil Code allows students to write real CoffeeScript program with blocks only. As discussions regarding pseudo - code [8] are helpful in distinguishing between structural and functional elements of language. It will be good for our study because it allows students to write real CoffeeScript program using blocks. In addition; CoffeeScript is similar to Python, so it will open the way to learn a simple and powerful language like python easily. Also, the project is an open source.

However, allowing students to see text translation for their blocks did not help enough to understand what every block is represented in text-based. Think about converting blocks to code, it is hard for students to keep tracking what every block mean in code. For that reason, we come up with the hybrid-based environment.

In hybrid-based student will be able to see the organic CoffeeScript code for any block, after they drag-drop the block to the development environment. The development process is from three steps:

1. The student selects the block to see Figure (3.2a).
2. Student drags the block see Figure (3.2b).
3. Student drop it in the development environment see Figure (3.2c). As we see how the For-loop block is converted to the Code.



```
for [1..3]
```

Figure(A): Select block Figure(B): Drag block Figure(C): Drop block

Figure 3.2. Drag-drop command process

We will discuss in next sections all details about design and implementation of the hybrid-based environment from Pencil Code.

II. System design

The system design that is shown in Figure 3.3 is represented hybrid-based system architecture. The idea behind hybrid-based is covering the gap between the block and text-based, So the learner will be able to use the blocks and see the code to have simple and easy user experiences while he learns to program.

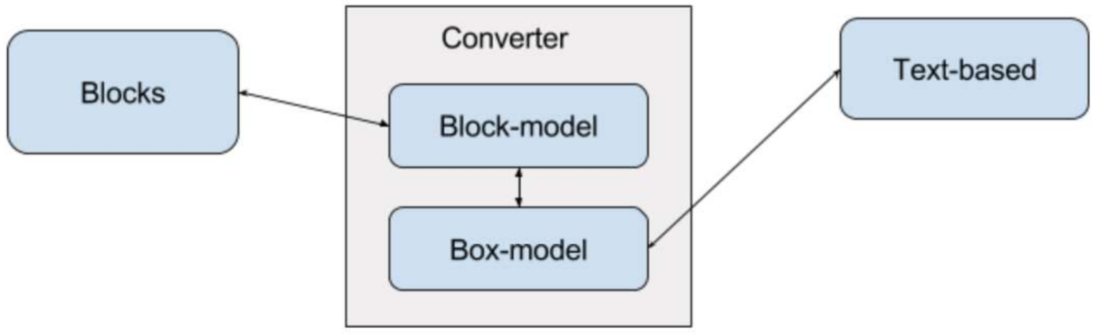


Figure 3.3. Hybrid-based System Design

By looking at Figure 3.3, we see that the system design is really simple. We have three components; the First component is the blocks that represents every programming reserved command (For, IF, variables, etc) that will be available in the toolbox for a user to be able to drag-drop to build his application. We are using Google Blocky [17] as block-based in our system that already shipped with Pencil Code [10].

The second component is the converter which is droplet [15] that we used to convert blocks to code and code to block. As we know by drag-drop blocks from the toolbox to the workspace will lead to block-model which represent the program as a number of connected blocks. We have to convert the blocks to box-model which is pure code when user drag-drop blocks. To do that we used Droplet [15] as to convert between text-based and block-based.

The third component is the text-based that place to see the code and enable the user to drop blocks in this environment from the toolbox and edit the code using the keyboard.

III. Pencil Code

Pencil Code allows students to write real CoffeeScript program with blocks only. By drag-drop blocks from the toolbox, the student could build his own applications without taking care about the syntax in a programming language. Also, Pencil Code enables students to switch between the code and blocks to see how every block will be represented in the code. Figure 2.2 shows how we could build a simple app that allows the player to draw squares in a different angle.

Pencil Code is a web application. It is being built using web technology like JavaScript, CSS, HTML, and Python. There is a main file located under “content/editor.html” which does all the works, also the JavaScript files behind the editor.html located in “src/” folder. Also, there is some internal library this tool is used which is two main tools.

1. **Droplet:** Droplet editor [15] seeks to re-envision block programming as text editing and re-envision text editing as block programming.

2. **Blocky:** Blocky is a library that enables users to add visual code editor to the web. The Blocky editor uses interlocking, graphical blocks to represent code concepts like loops, condition, and variables add more [17]. We will discuss both in details.

1- Droplet

Droplet editor [15] seeks to re-envision block programming as text editing and re-envision text editing as block programming. The droplet is used as a part of pencil Code [10] to switch between the block-based to text-based to enable users to see the code. The problem in using droplet within pencil Code that used to convert a complete program that builds using blocks to text-based which confuses the students about which every block will be represented in the code when he clicks on the switch to code. Especially when the programs grow for 10 lines and more.

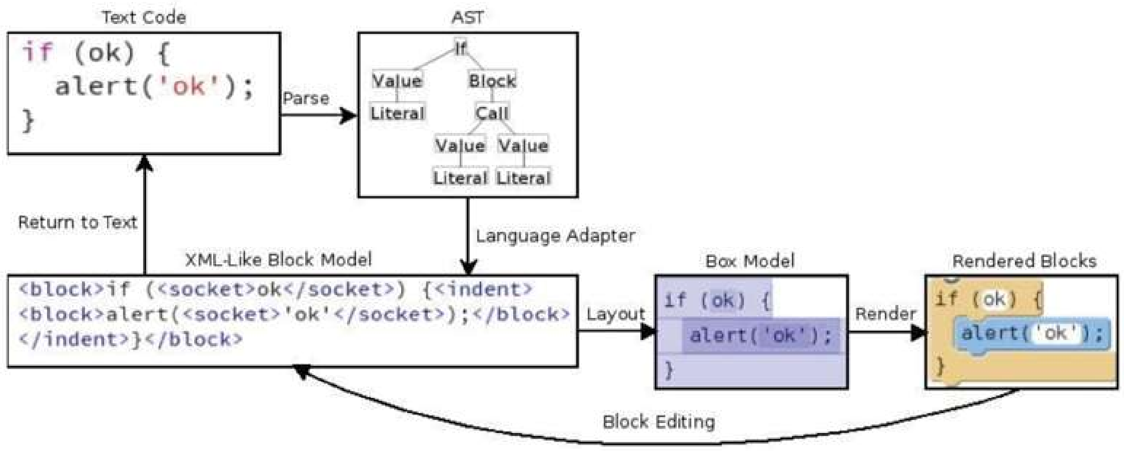


Figure 3.4. Droplet architecture [15]

The droplet can edit JavaScript and CoffeeScript. In Pencil Code, they used CoffeeScript because it was easy to understand with no semicolon and less syntax error. Also, it provides easy extensibility to any other programming languages.

Droplet's data model is text stream marked up with XML-like tokens such as **<block>,</block>,<socket>,</socket>**, where every block that dropped to droplet editor will be represented as a token with starting and ending. The example that can be seen in Figure 3.4 shows when someone tries to type condition statement in droplet editor to display a specific alert when the condition is ok.

To convert text code to blocks, first, the text code will be parsed to AST tree to find the parent and children. Then this tree will be converted to XML-like blocks model using language adapter that will be responsible for defining the language that this code will be represented. In Pencil Code, we used CoffeeScript. We could extend the adapter to another programming language, by writing new language adapter. The adapter will represent the IF condition as a block because we know that conditions could have many lines of code as children, so these lines have to be surrounded by block. While the logic condition is if statement and the alert command are represented as a token because they represent a single line with no children. Also, this adapter has the ability to do type-checking or to deal with orders of operations; the language adapter receives a callback. Before a new blocks dropped into droplet editor, the adapter may prohibit the drop process if the type is incomplete.

After having the XML, the droplet needs to render the XML to blocks. The droplet will read the code line and consider every line. For each line, it will surround all the text by block, sometimes spacing will be inserted between the lines in each box model to make the block clear when the path is drawn. The droplet will draw the path that is surrounded all its rectangles but avoids the unintended area.

When we convert all text to block droplet will show animation. We disabled this animation when we updated the code to support hybrid-based environment because in hybrid-based we need to convert every block to code which leads too many animations every second that confuses the developer while he is working on writing his program.

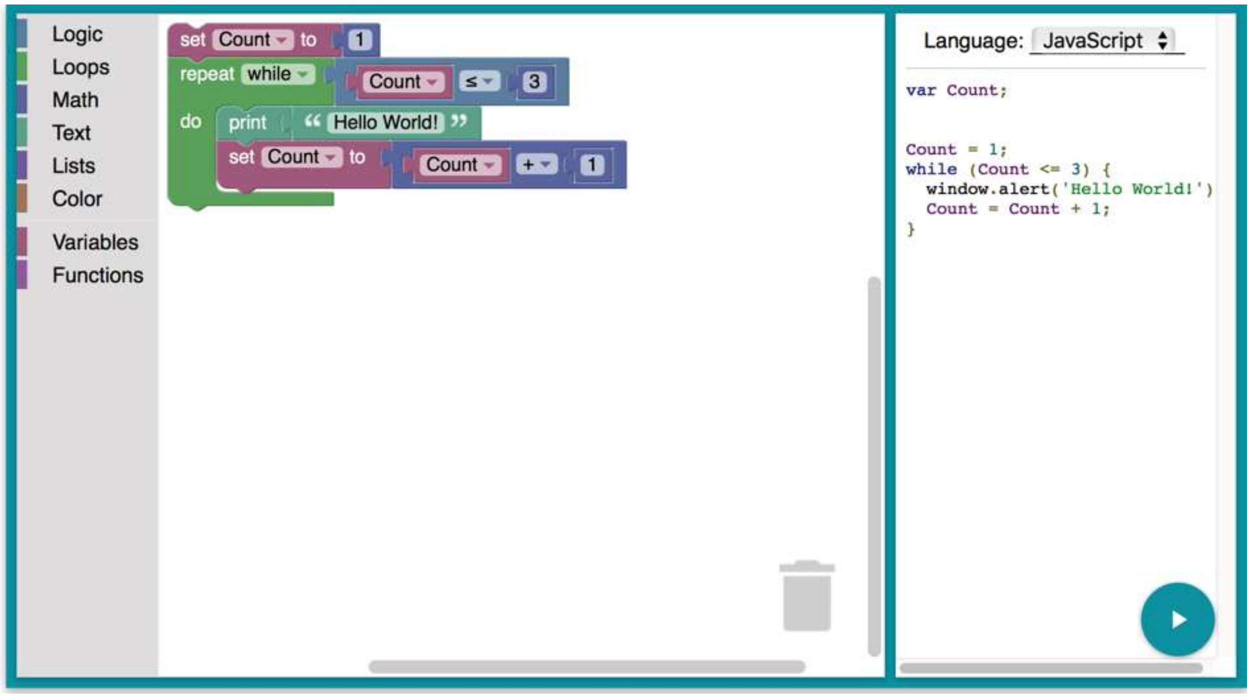
To convert blocks to text code, the droplet will parse the blocks to XML-Like block model in same previous parsing process then return XML back to text code without using a tree because blocks already defined by color using Blocky.

The idea that we come with is converting every drop block to code directly, so the students see how every block will be represented in the code, while he writes his programming. Instead of sending all program to droplet model to convert it to code, we send every block to droplet to convert it to code, also we enable users to edit and write code in droplet editor and we send the code back to droplet model to build the blocks. While students see program only as code, we hide the program as blocks.

2- Blocky

Blocky is a library that enables users to add visual code editor to the web. The Blocky editor uses interlocking, graphical blocks to represent code concepts like loops, conditions, and variables. See Figure 3.5 that shows Blocky editor.

The block editor is pure JavaScript library work on the client side 100%. It does not need for server-side calls when students drag-drop which make it working faster and use less network bandwidth. In addition, Blocky is compatible with all major browsers like Chrome, Firefox, Safari, Opera, and IE. Blocky build to be high extensible, so it is really easy to be integrated with any tool. For that reasons, there are many block-based tools published that using Blocky like Pencil Code, Oz Blocky, Code bug, Code.org, Micro: bit, Wonder workshop, and App Inventors. These different libraries customized Blocky to match their needs.



```
var Count;


Count = 1;
while (Count <= 3) {
  window.alert('Hello World!')
  Count = Count + 1;
}
```

Figure 3.5. Block editor [17]

Pencil Code is one of the block-based tools that used Blocky to build an app from blocks. By looking back to Figure 3.4, you will find that block is used to represent loops, conditionals, variables, functions, and game movement commands. Pencil Code developers customize Blocky to build droplet that has the distinguishing feature that is the ability to convert from code to block and block from code.

IV. Implementation

The system is designed for hybrid-based that is shown in Figure 3.3 is structured in the same way that Pencil Code system design is built. In Pencil Code, we could switch between the text-based and block-based one-time when the user clicks on “show code” or “show block”. What is happening when the user clicks on “show code?” The blocks will be passed from Blocky to Droplet model to show the code, while users click on “show block” the code is passed from droplet to Blocky model to build blocks. That means that there are two different separate models in Pencil Code which is droplet and Blocky.

To implement hybrid-based environment, we will not change Pencil Code system design instead of adding and updating some functions. By looking at Table (3-1), you will find that all details about every updated file that are described by updated file name, updated file path, number of lines that updated, process type which may be either add new line of code, delete line or update existing lines, and update ID that we give unique identification for every update to keep track our work. We will discuss in detail all updates that we did for folder Hybrid Pencil Code [11] to produce new hybrid-based environment.

JavaScript

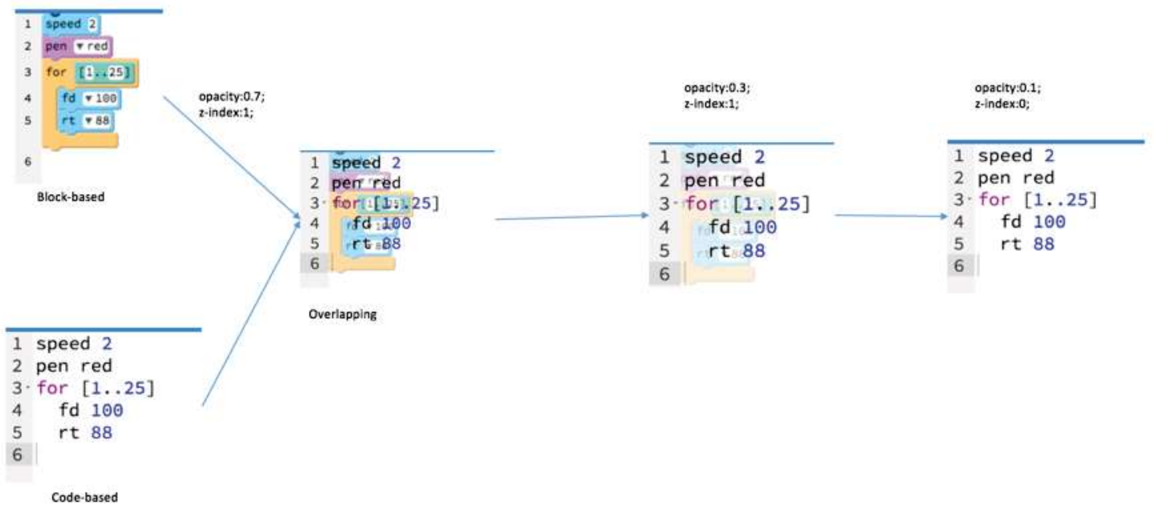
```
//added by Hussein alrubaye
$('.pane').on('change key up paste', 'text area', function(e){
  var pane = $(this).closest('.pane').prop('id');
  e.preventDefault();
  setPaneEditorBlockModeUpdate(pane, true,$(this).attr('droplet-editor'));
});
```

Snapped Code 3.1. Change key up paste method

For update id (8), Instead of switching between code-block by click on “show code” or “show block”, we need to do this process automatically behind the scenes without letting the student know about that. To do that, we added a new function that fire. Whenever any block is dropped from toolbox to the development environment or whenever the developer updates the text-based, see Snapped Code 3.1. When we drag-drop block, this function will fire and pass the block directory to droplet to generate the code in front of the student. So when students drop any block he will see it as a code. Same things when students change the line of text-based by keyboard this event will fire and build the blocks from the code.

For update id (7), In hybrid-based both text-based and block-based are shown in front of user screen instead of one by time with come style sheets that hide the program as blocks and show the program as code only. So we just stopped that line of code that prevents the toggle blocks and generates the code. So whenever any block is dropped, the droplet will build generate the text-based and when any line of code is modified, the droplet will build the blocks.

For update IDs (2, 3, 4, 5, 6), these updates are dealing with user interface design. As we know Pencil Code allows converting between two environments block-code, so we need to change the layout from switching between two environments to one environment that overlaps the blocks and code in such way the developer could see hybrid-based.



```
speed 2
pen red
for [1..25]
  fd 100
  rt 88
```

Figure 3.6. Update UI

See Figure 3.6 that shows the process of overlapping the blocks and text-based layout. First, we have two different environment block and code, and then we overlap these two environments by changing the place where they are located using update IDs (3, 5, and 6) that allows us to place both environments in the same location on the screen.

Now we need to hide the block-based, so we used update ID (2) to reduce the opacity. We change the opacity of blocks to 0.1%, so we cannot see blocks. Then we need to hide block-based after the text-based, so the developer will be able to write code in text-based. We used z-index to move block-based behind the text-based.

There are many processes happens behind the scenes for converting any drop block to code and any written code to block which lead to many animations. We need to update the animation to zero using update ID (3), so the developer will not become confused. Finally, we had the hybrid-based Pencil Code development environment.

File Path	File name	Line number	Process type	Update ID
content/	**editor.html**	**13-15**	**Update**	**1**
content/lib/	**droplet.css**	**49-51**	**Add**	**2**
content/lib/	**droplet.js**	**65308-56310**	**Update**	**3**
content/lib/	**droplet.js**	**65479-65480**	**Update**	**4**
content/lib/	**droplet.js**	**65826-65829**	**Update**	**5**
content/lib/	**droplet.js**	**65308-65310**	**Remove**	**6**
content/src/	**view.js**	**2841-2842**	**Remove**	**7**
content/src/	**view.js**	**1933-1943**	**Add**	**8**

Table 3.1. Edited files

V- Deployment

After we completed the developing the hybrid-based, now we have to make it public for students to start work with it. Because it needs specifics frameworks that are not available in most of hosting companies. We rent virtual machine on Microsoft Azure with public IP, then we installed the needed framework which is (python 2.7, node.js), and we host the hybrid-based Pencil Code on the virtual machine to be accessible from the public.

Chapter 4

Methodology and Results

Methodology and Results

I- Methodology overview

The study makes up to compare the effect of two different teaching programming environment block-based and hybrid-based on transfer programming skills when students move forward to write code in the text-based environment. For this case, we need to study the effect of block-based and hybrid-based on transfer programming skills separately. By teaching every environment to a different group of students then evaluating the effect of every environment on transfer programming skills. Then find out which tool is better.

To do this experiment, we sent flyer message to students in the department of civil engineering and environmental science in Rochester Institute of Technology. We sent these two departments because they did not teach programming course, so students did not have any programming experiences.

We selected 18 students for participating in the study for one week, four hours' sessions, and every session takes 2 hours. We tried to follow the same procedure that was done by David Weintrop [9]. We are dealing with the human subject, so if we have an error, it will cost us a lot of money. We decide to divide 18 students into two Groups, See Figure 4.1.

1- Block-group: This group will learn programming concepts using Block-based.

2- Hybrid-group: This group will learn programming concepts using Hybrid-based.

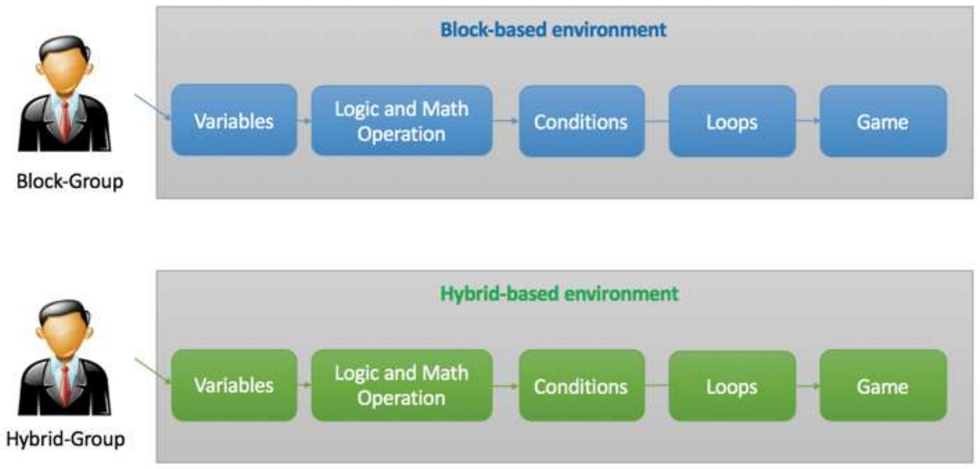
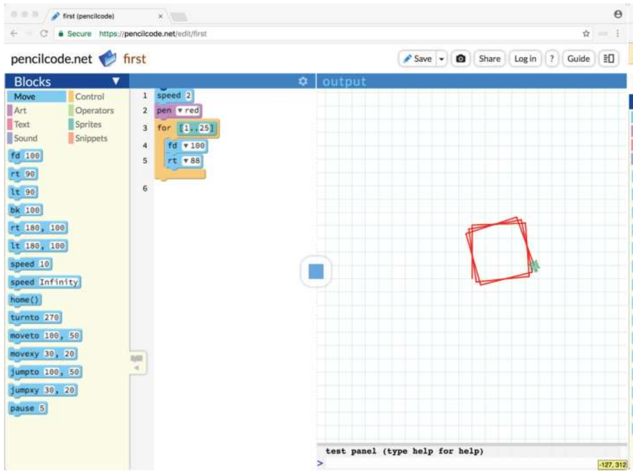


Figure 4.1 Learning path

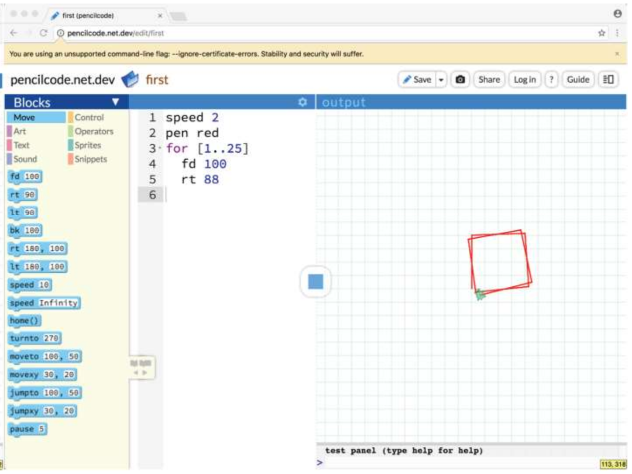
A- Class schedule and team's environment:

By dividing the students into two groups, we scheduled class time for every group to meet and learn programming foundation with their environment at (Monday 2/6/2017, Wednesday 2/8/2017). Block-group will learn to program with the Block-based environment, see Figure (4.2a). This Team will learn to program using Pencil Code [10] as a block-based learning environment. While hybrid-group will learn to program with the Hybrid-based environment, see Figure (4.2b). This Team will learn programming using Hybrid-based [11] as a learning environment.



```
speed 2
pen red
for [1..25]
  fd 100
  rt 88
```

Figure (A)



```
speed 2
pen red
for [1..25]
  fd 100
  rt 88
```

Figure (B)

Figure 4.2. View of Block-based (A) vs. hybrid-based (B) with Pencil Code

We scheduled to teach same topics for two teams to make sure we could evaluate the effects of both block-based and hybrid-based on transfer programming skills. We booked one of Rochester institute of technology software engineering classes for specific time slots. We used the data show to teach the students the programming topics with visualizing the objects and demonstrating the execution of every program that we teach during the sessions.

For teaching time, we scheduled two classes on different time for every team. That means every team will meet two times, and the time frame for every meeting is two hours. Total class duration was four hours that was divided into two hours for every team separately.

B- Class agenda:

First class on Monday 2/6/2017, in this class we teach every team separately on same topics. We started by Pre-Test by testing student logic, programming, and problem-solving to make sure that students who participate do not have any programming experiences.

Then we started class teaching (see Appendix A) with variables and how we could use variable to hold different data type. We teach students the strings in details, and how they concatenating two string or string with different data type like an integer. Also, we teach the students about input data to the program from screen to make the program usable by the users, and how the program runs step by step. Then we moved forward to teaching the logic statements, by giving them different examples of logic. We focus on this activity on a comparison using logic operation ($3<2$ is false while $3>2$ is true). Then we teach the students the condition statement (IF-Else), and how we could execute a specific block of code under a specific condition. We teach different examples like change student degree to level (A, B, C). Then we moved forward, and we teach the student's loops commands (For, While), and how we could run a block of code more than one time. All materials on Github repository [13], lecture 1, 2.

Second class at Wednesday 2/8/2017, in this class we teach every team separately same topics. We start by teaching the draw game [11] under activities folder (lecture 4). This activity helps students to execute what they learn in the first lecture on real-world application by moving the player to different places using (Loops, conditions), so the students will feel fun and engagement when they learn to program [14].

Then we did the Post-Test (see Appendix B) for 20 minutes only. The test was using CoffeeScript as a text-based language because both environment block and hybrid-based build on CoffeeScript, so students will not feel shocked when they see the code.

II- Results

After teaching every group how to write a program in their environment hybrid or block-based, we did the post-test to evaluate the effects of two environmental on transfer programming skills. Finally; we come up with a number of surprising results about Hybrid-based environment effects on transfer programming skills to text-based.

To compare the effect of hybrid and block-based tools on learning programming, we find that the hybrid-based environment enhances the learning of programming for the students in a number of features:

1- **Easy to learn: The** Hybrid-based environment is easy to learn than the block-based environment.

2- **Code Modify:** Students who are using hybrid-based environment have high ability to modify than the block-based environment when they move to text-based.

3- **Memorize Commands: The** Hybrid-based environment is easy to memorize commands than the block-based environment.

4- **Syntax Errors:** In the hybrid-based environment, students have less probability of producing syntax errors than the block-based environment, when they move to text-based.

5- **Learning Shock:** Students who were using hybrid-based environment have less ration of having environment sock than the block-based environment when they move to text-based.

Finally; we answered research questions by:

Q1: Figure (4.3, 4.4, 4.5, 4.6, 4.7) that talk about enhanced the easy to learn, code modifies, memorize commands, and syntax error answered research question that “Is hybrid-based allows the student to understand and Identify programming concept (Loop, Variables, Control structure) easily than block-based when they start with Text-based programming?”

Q2: Figure (4.8) that talk about reduced the learning shock for students when they moved to text-based answered research question that “Will hybrid-based environment reduce the misunderstanding (learning shock) that student may feel when transfer from block-based to text-based?”

We will discuss every enhancement point in details:

1- **Easy to learn**: Students who were learning program with hybrid-based have high grades than students who were using block-based development environment, See Figure 4.3.

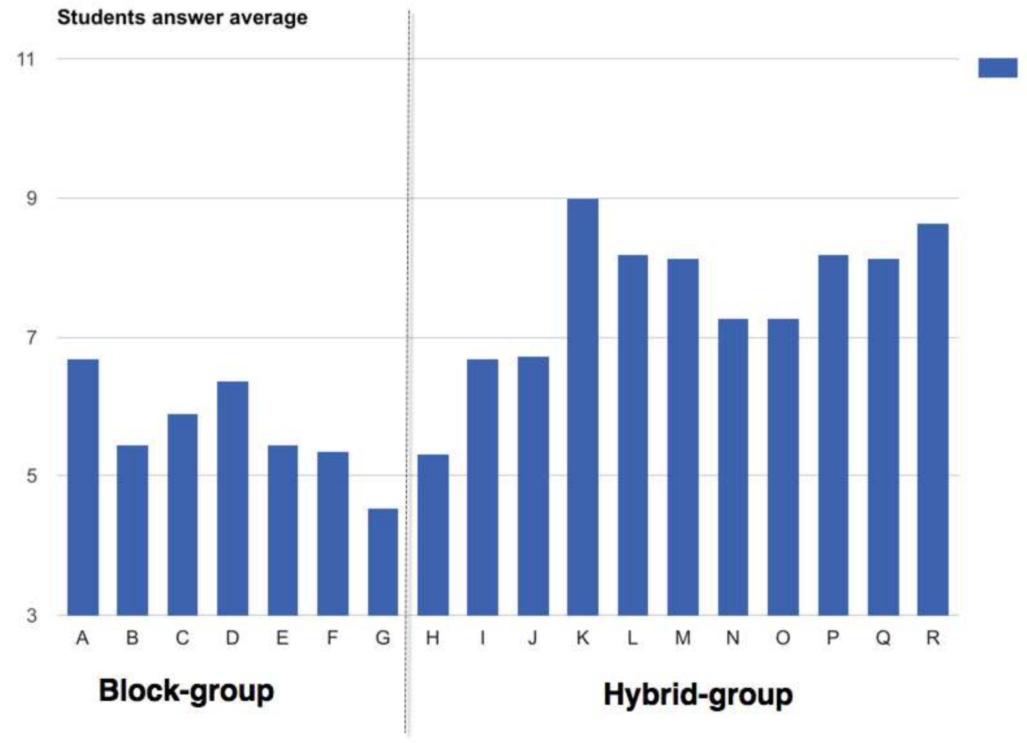


Figure 4.3. Student grades level

This figure shows that students who were using hybrid-based are advanced on students who were using block-based in their grades. This is agreed with what we think when we did the experiment because hybrid-based students were able to see the code during the learning.

Hybrid-based helps students to understand the program flow, variables, loops, and syntax better than students who were using block-based, because the hybrid group was seen the code during the learning, in addition to getting the advantages of drag-drop blocks.

In hybrid-based students were able to drag-drop blocks of code without need it to memorize the commands. In same time students see how the block convert to code when they drag-drop blocks from the toolbox to development area. While block-group could see only block when they drag-drop blocks from the toolbox to the development area. As results; when hybrid-group see the text-based questions they did better than block-group. For that reason, hybrid-group got high grades than block-group.

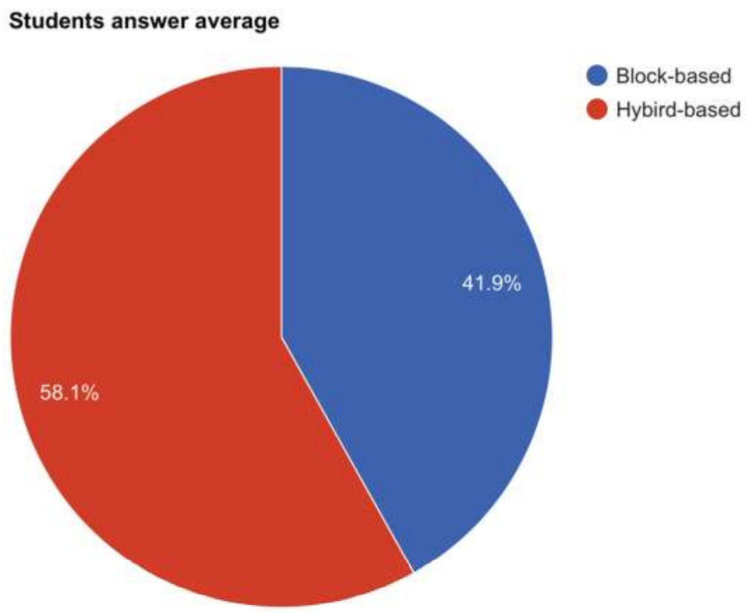


Figure 4.4 Compare student grades

By looking at Figure 4.4 that compares the answers of two groups. We see that students who were using hybrid-based have 58.1% while students who were using block-based have 41.9% only. That means that hybrid -based enhanced students learning by an increase in students' grades 16.2% in comparing with the block-based.

2- **Code Modify:** We wanted to validate the study ability to modify existing code. So we asked the student two questions (see Appendix B). In question number (6) that asks to update the move of the tortoise on specific. Also in question (13-IV), we asked students to update the program of comparing two numbers to catch the case of zero input.

Students who were using hybrid-based development environment have high ability to modify existing code than students who were using block-based development environment. As Figure 4.5; we see that hybrid students have 64.3% to modify code while block-based have 35.7%. That means hybrid students are in advance about 28.6% of students who were using block-based on code modify.

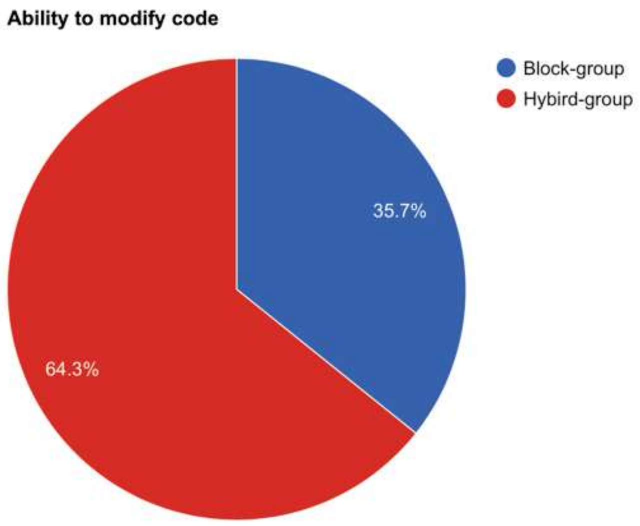


Figure 4.5. Ability to modify code

Their results come as we expected because block-based tools do not enable users to work directly with mainstream programming languages [15], they could use blocks only to write the program, which makes really hard for these students to modify existing code that they do not have any experiences in code modification. While hybrid -based enable users to work directly with mainstream programming languages. So, students who were using hybrid-based, they had direct interaction with the text-based, so when a student wants to assign a value to variables or write additional code, he uses the keyboard instead of the drag-drop blocks.

As a result; being in touch with text-based, lead to experiences with code modification, and ability to add or update existing code. This tool is different from Droplet [15], with hybrid-based students do not need to switch between block-based and text-based, they drag-drop the block and it converts to code. While in droplet students need to switch block to code to see the real code, which is difficult for students to track the representation of blocks in the code when the program grows for more than 10 lines.

3- **Memorize Commands:** To measure which environment is better in memorizes the programming commands. We did a survey at the end of the study by asking students from both teams to review their abilities to memorize commands from one to five. Where one is bad and five is good, by answering these questions (See Appendix B, question 19. And 20):

Q19- Did you had hard time to understand loop, condition, and functions in text-based environment (5 stars is good 1 star is bad)?
Q20- Are you able to memorize the command of the loop, condition, and functions, when you start with in text-based environment (5 stars is good 1 star is bad)?

Students who were using hybrid-based development environment find it easy to memorize, and understand programming commands than students who were using block-based development environment. As Figure 4.6; we see that hybrid students have 54.5% to memorize commands while block-based have 45.5%. That means hybrid students are advanced 9% on students who were using block-based in memorizing the commands.

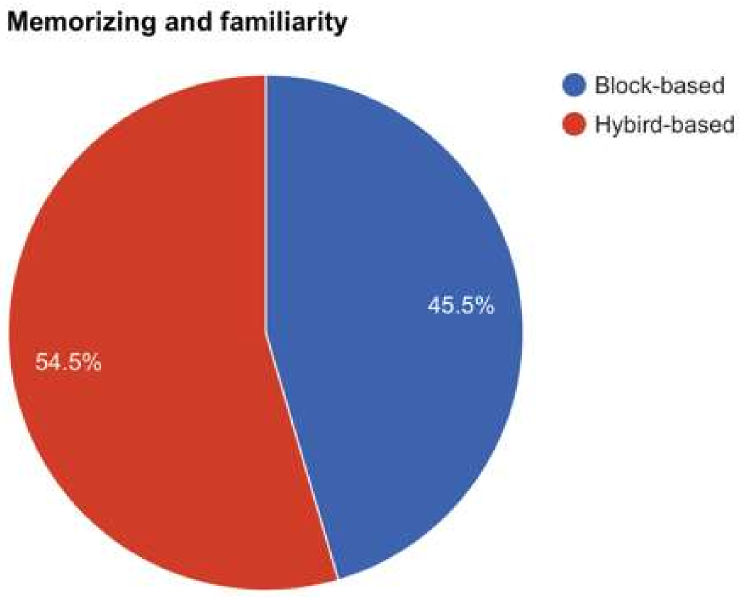


Figure 4.6. Memorize commands

Memorize programming commands (IF, For, and Variables) in the block-based environment was one of the most interesting positive point for a block-based development environment because students who use block-based are able to see every programming commands as blocks, so they will be able to memorize the commands [1] as many researchers agree on.

However, we find that students who were using hybrid-based they have high ability to memorize the commands than block-based students, because they are able to play with command in the code, and rewrite command again using the keyboard, in addition, to see every programming commands as blocks. As results; See command as a block and write it, lead to strong ability to memorize the commands.

4- **Syntax Errors:** We wanted to validate the study ability to find syntax error and writing code with correct syntax. So we asked the student three questions (see Appendix B). In question number (5) we added syntax error in the condition statement by making the donation and the condition block of code start from same line space which does not work in a CoffeeScript, because it is space sensitive language, we have to add space before adding the condition blocks of code. Also in question (13-IV), we asked students to write a program to check their code syntax. Third question (17) to write a program that compares two numbers and prints the greater one.

We find that students who were using block-based have a high ration of produce syntax errors than hybrid-based. Only 20.5% of Students who were using hybrid-based development environment have syntax errors when they start with text-based. While 87.5% of students who were using block-based had syntax errors. See Figure 4.7.

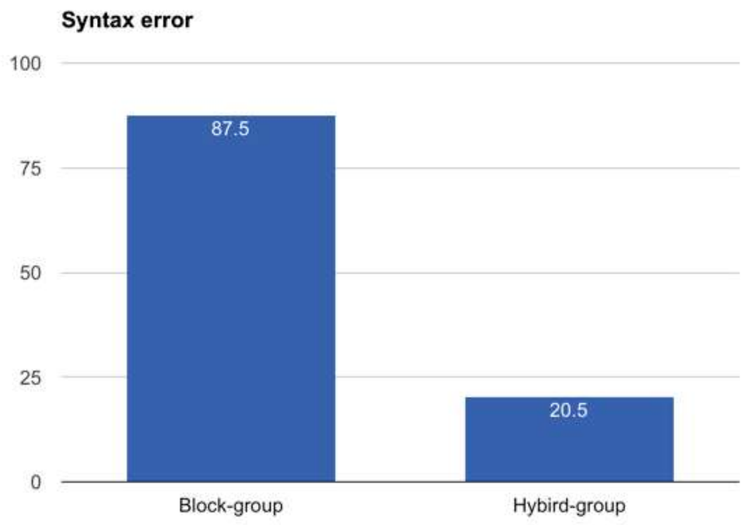


Figure 4.7. Syntax Errors

The hybrid group is supported learning programming syntax, so students learned what is programming syntax during their lessons and what spacing mean in CoffeeScript, while they were writing their programs. So this result is totally meaningful. The students in the hybrid group are being in touch with the code, while they were learning to program. So they have the ability to write a program with correct syntax than block-group. Because of block-group work with block only. So there is no chance for them to learn programming syntax with blocks. As results; they do not have a chance to learn programming syntax.

Writing a program with correct syntax is one of the most interesting contributions that hybrid-based tool comes up with because it allows the students to learn the program syntax in addition to getting benefits of learning with blocks. That help these students, when they move to text-based, they will understand every programming language which have their own syntax which leads to less learning shock and less syntax error. Even block-based avoid program syntax to make learning easy [1]. In hybrid-based, we help students to understand the syntax by giving hints about where next line should be placed to avoid a syntax error.

5- **Learning Shock:** A feeling that students feel when they move from block-based to text-based environment. "Educator had various difficulty transitioning learners from this graphical environment to conventional text-based program environment" Weintrop, D. (2015, October). Moving from the environment that allows building program using blocks only which is block-based to a new environment that builds a program using code only which is text-based. This change makes students feel that they are in the new environment and they need to learn new things which lead to learning shock.

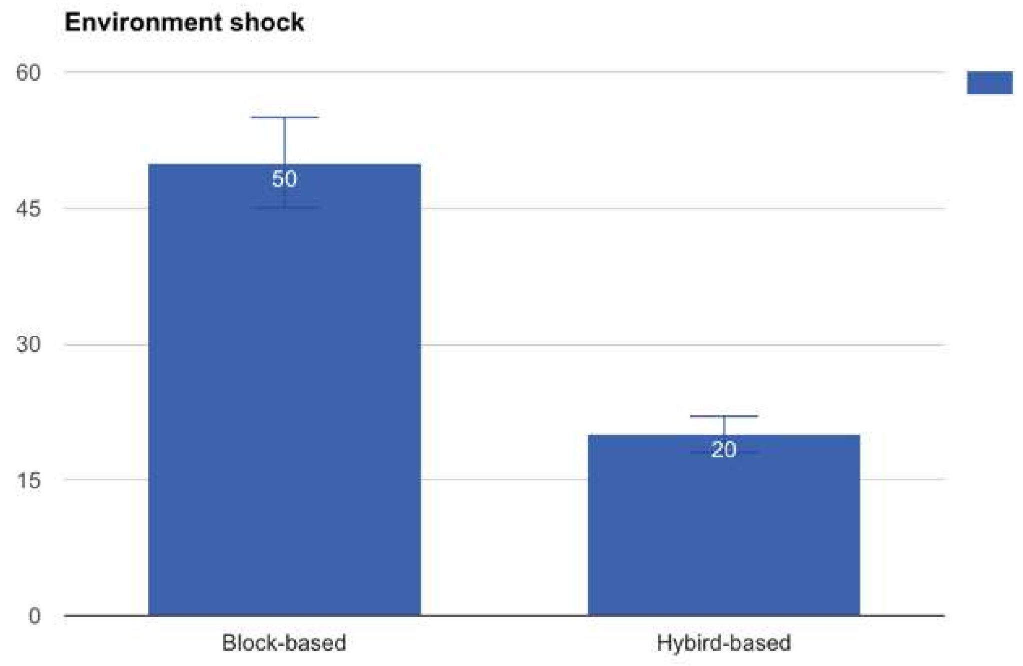


Figure 4.8. Learning Shock

To measure the Learning shock that a student may feel when he moves to text-based, we did a survey at the end of the study by asking students from both teams this question (See Appendix B, question 21):

Q21: When you started writing code in the text-based environment, did you feel you are in a new development environment, or the commands for loops, conditions, and functions were familiar to you?

A- Yes, it looks new

B- No, it looks familiar

We find that students who were using hybrid-based development environment had less Learning shock than students who were using block-based development environment. Only 20% of Students who were using hybrid-based development environment have a learning shock when they started with text-based and they answer this survey question by “yes, it looks new”, while other students answered this question by “No, it looks familiar”. While 50% of students who were using block-based answered the survey question by “yes, it looks new”, because they had Learning Shock, and they feel that they are in the new environment. See Figure4.8.

This result is totally meaningful because the hybrid group is being in touch with the code while they were learning to program, so they will not feel shocked with the text-based environment. While block-group never saw the code before, because they work with block only. As a result; they feel that the code is something new to them.

III- Study observations

During the learning sessions, while students were listening and executing the activities with block-based and hybrid-based. It seems very clear to me that students who were using block-based were engaging and comfortable while students who were using the hybrid-based seem to be more serious. We believe that this feeling happen because block-based minimize the complexity of writing code and make it more attractive and fun for beginners with nice shapes and color than text code only. In the same time, this design of block-based has a high impact on having learning shock for students when they move to text-based.

IV- Results Discussion

According to the results that we discussed in results section that compare the effect of hybrid-based and block-based on how easily student will learn to program. We find that hybrid-based environment enhanced the student's ability to learn and identify the programming (Loop, Variables, Control structure) easily than block-based because hybrid-based enhanced the easy to learn, code modifies, memorize commands, and syntax error for students in comparing with block-based. So here we answered by yes for first research question that is:

- ***Is hybrid-based allows the student to understand and Identify programming concept (Loop, Variables, Control structure) easily than block-based when he starts with Text-based programming?***

Also hybrid-based reduced the learning shock that students were feeling when they were learning to program with block-based then they moved to text-based because students who were using hybrid-based development environment find less learning shock than students who were using block-based development environment. So here we answered by yes for second research question that is:

- ***Will hybrid-based environment reduce the misunderstanding (learning shock) that student may feel when transfer from block-based to text-based?***

Chapter 5

Conclusion and future works

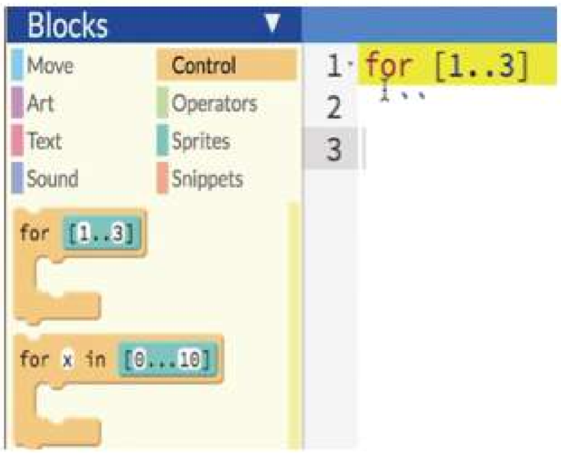
Conclusion and Future Works

I- Conclusion

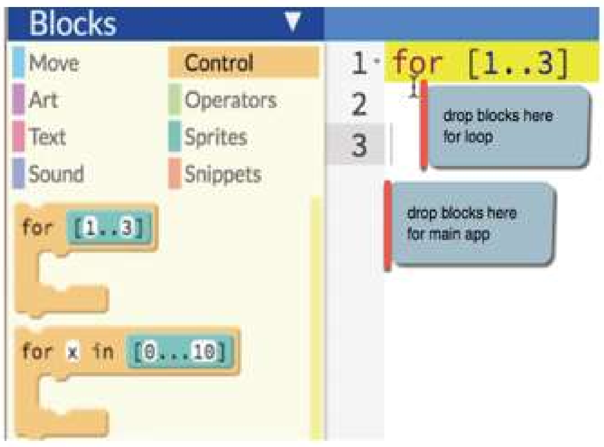
This thesis compares the effects of hybrid-based and block-based environment on transfer programming skills to text-based. According to the results that we found after teaching two groups, programming with different environment block-hybrid-based then move both teams to write code with text-based. We found that hybrid-based environment is better than the block-based environment in transfer programming skills because hybrid-based enhanced the students' abilities to learn programming foundations, code modifies, memorize commands, and syntax error comparing with block-based. Also hybrid-based reduced the learning shock that students were feeling when they were learning to program with block-based then they moved to text-based because students who were using hybrid-based development environment are interacting directly with code, while block-based group never saw the code before.

II-Future works

Adding hints and guidelines in hybrid-based will make the development environment more interactive especially while the students are writing the program. We believe that the student needs more hints and guidelines, while he is writing his program with hybrid-based to assist him to learn program syntax easily, also help him to identify the next required step. We think our tool need improvement by adding more hints for students while they drag-drop blocks to write the program. So in future, we want to enhance the learning with the hybrid-based environment and make the learning easier. By adding additional comments and hint for students when they drag-drop blocks. For example; in Figure (4.1b) shows that when student drag-drop "loop" block, he will ask to add the block content while in current version Figure(4.1a), he will see only single quotation that does not mean anything.



Figure(A)



Figure(B)

Figure 4.1. Current (A) vs. future Hybrid-based (B)

References

References:

[1] Weintrop, D. (2015, October). Blocks, text, and space between The role of representations in novice programming environments. In *Visual Languages and Human-Centric Computing (VL/HCC), 2015 IEEE Symposium on* (pp. 301-302). IEEE.

[2] Brown, N. C., Mönig, J., Bau, A., & Weintrop, D. (2016, February). Panel: Future Directions of Block-based Programming. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (pp. 315-316). ACM.

[3] Bau, D., Bau, D. A., Dawson, M., & Pickens, C. (2015, June). Pencil code: block code for a text world. In *Proceedings of the 14th International Conference on Interaction Design and Children* (pp. 445-448). ACM.

[4] App Inventor reaches 2.7 million registered users. URL: http://appinventor.pevest.com/?p=512 Accessed February 2017.

[5] Scratch community statistics. URL: https://scratch.mit.edu/statistics/. Accessed February 2017.

[6] Hour of code. URL: http://hourofcode.com/us. Accessed March 2015.

[7] Robinson, W. (2016, October). From Scratch to Patch: Easing the Blocks-Text Transition. In *Proceedings of the 11th Workshop in Primary and Secondary Computing Education* (pp. 96-99). ACM.

[8] Cutts, Q., Connor, R., Michaelson, G., & Donaldson, P. (2014, November). Code or (not code): separating formal and natural language in CS education. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education* (pp. 20-28). ACM.

[9] David Weintrop.Modality Matters: Understanding the Effects of Programming Language Representation in High School Computer Science Classrooms,(2016, September)

[10] Pencil Code portal. URL: https://pencilcode.net. Accessed February 2017.

[11] Hybrid Pencil Code. URL: https://github.com/hussien89aa/HybridPencilCode. Accessed February 2017.

[12] Perlis, A. J. (1962). The computer in the university. MiT Press.

[13] Feurzeig, W., Papert, S., Bloom, M., Grant, R., & Solomon, C. (1969). Programming languages as a conceptual framework for teaching mathematics (BBN Report No. 1889). Cambridge, MA: Bolt, Beranek, and Newman.

[14] Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... & Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, *52*(11), 60-67.

[15] Bau, D. (2015). Droplet, a blocks-based editor for text code. *Journal of Computing Sciences in Colleges*, *30*(6), 138-144.

[16] Pencil Code. URL: https://github.com/PencilCode/pencilcode. Accessed February 2017.

[17] Blockly. URL https://developers.google.com/blockly/ . Accessed February 2017.

[18] diSessa, A. A., & Abelson, H. (1986). Boxer: A reconstructible computational medium. *Communications of the ACM*, *29*(9), 859-868.

[19] Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35.

[20] Astrachan, O., & Briggs, A. (2012). The CS principles project. *ACM Inroads*, *3*(2), 38-42.

[21] Goode, J., Chapman, G., & Margolis, J. (2012). Beyond curriculum: the exploring computer science program. *ACM Inroads*, *3*(2), 47-53.

[22] Reed, D., Wilkerson, B., Yanek, D., Dettori, L., & Solin, J. (2015). How exploring computer science (ECS) came to Chicago. *ACM Inroads*, *6*(3), 75-77.

[23] Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... & Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, *52*(11), 60-67.

[24] Jona, K., Penney, L., & Stevens, R. (2015). Remediating Learning. In Proceedings of the 11th Annual Conference on Computer Supported Collaborative Learning (CSCL. Gothenburg, Sweden).

[25] Stefik, A., & Hanenberg, S. (2014, October). The programming language wars: Questions and responsibilities for the programming language community. In *Proceedings of the 2014 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software* (pp. 283-299). ACM.

[26] Dijkstra, E. W. (1982). How do we tell truths that might hurt?. In *Selected Writings on Computing: A Personal Perspective* (pp. 129-131). Springer New York.

[27] Kaput, J., Noss, R., & Hoyles, C. (2002). Developing new notations for a learnable mathematics in the computational era. *Handbook of international research in mathematics education*, 51-75.

[28] Norman, D. A. (1993). *Things that make us smart: Defending human attributes in the age of the machine*. Basic Books.

[29] Soloway, E., Guzdial, M., & Hay, K. E. (1994). Learner-centered design: The challenge for HCI in the 21st century. *interactions*, *1*(2), 36-48.

[30] Teitelbaum, T., & Reps, T. (1981). The Cornell program synthesizer: a syntax-directed programming environment. *Communications of the ACM*, *24*(9), 563-573.

[31] Smith, D. C. (1977). Pygmalion: A Computer Program to Model and Stimulate Creative. *Thought. Basel, Stuttgart: Birkhauser*.

[32] Myers, Brad A. "Taxonomies of visual programming and program visualization." *Journal of Visual Languages & Computing* 1, no. 1 (1990): 97-123,p.98.

[33] Green, T. R. G., & Petre, M. (1996). Usability analysis of visual programming environments: a 'cognitive dimensions' framework. *Journal of Visual Languages & Computing*, *7*(2), 131-174.

[34] Sherin, B. L. (2001). A comparison of programming languages and algebraic notation as expressive languages for physics. *International journal of computers for mathematical learning*, *6*(1), 1-61.

[35] Luria, A. R. (1982). Language and cognition, ed. *J. Wertsch. Wiley.[aRAM]*.

[36] Boroditsky, L. (2001). Does language shape thought?: Mandarin and English speakers' conceptions of time. *Cognitive psychology*, *43*(1), 1-22.

[37] Griffiths, D. S., Winstanley, D., & Gabriel, Y. (2005). Learning shock the trauma of return to formal learning. *Management Learning*, *36*(3), 275-297.

Appendix A

This Appendix contains all materials that used in the experiment for teaching student programming.

Lecture 1: variables and Plan of study

1- Plan of study:

A- Students will be divided into two groups:

- Team1: Study programming concept with Block-based
- Team2: Study programming concept with Hybrid-based.
- Both teams will execute same tasks, and then they work text-based and answer number of questions about programming fundamental.

B- Both team start program with CoffeeScript

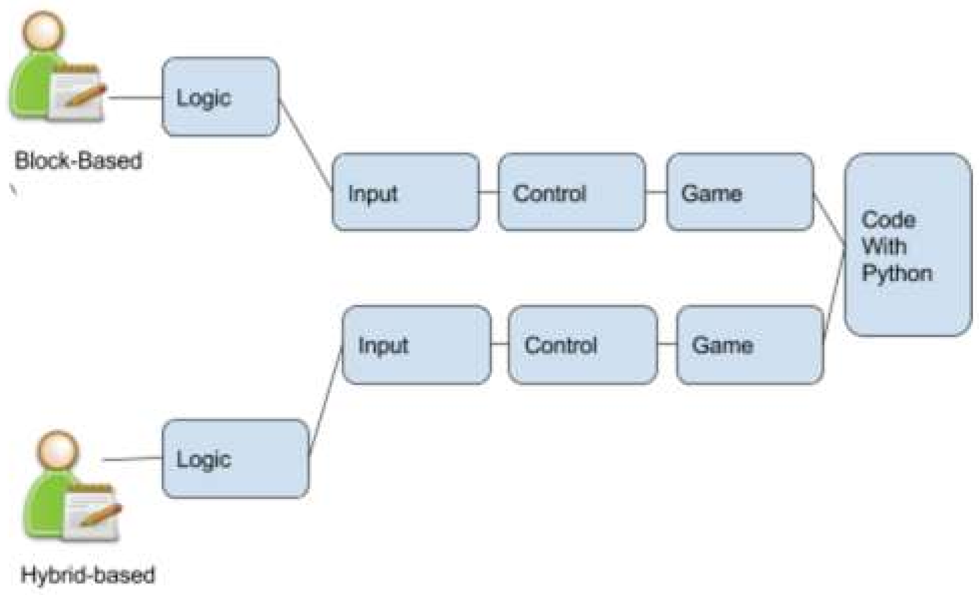


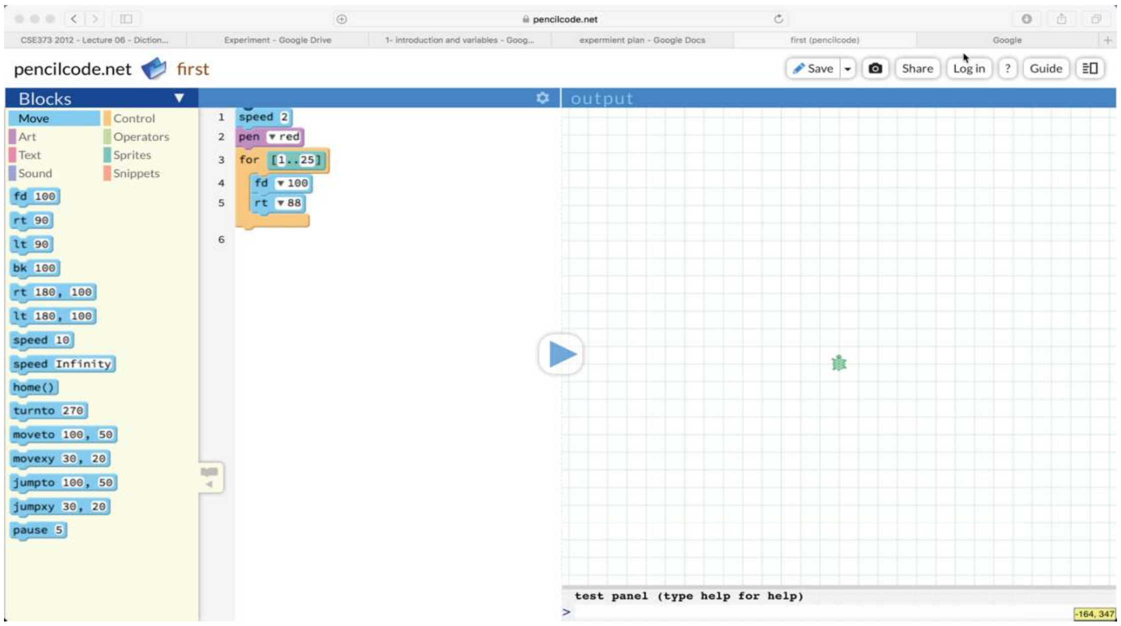
Figure1: Activities plan

Start Pre-Test to evaluate the students

https://goo.gl/LhqJ0q

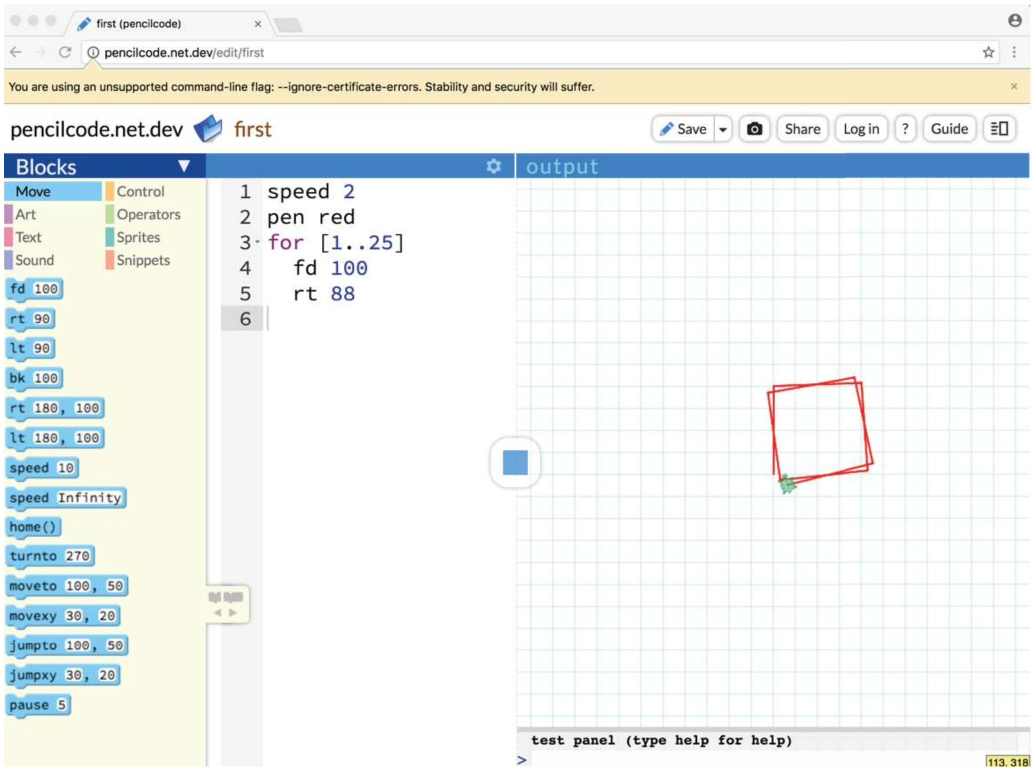
Working Portal

https://goo.gl/OD23jD



```
speed 2
pen red
for [1..25]
  fd 100
  rt 88
```

Block-based



```
speed 2
pen red
for [1..25]
  fd 100
  rt 88
```

Hybrid-based

2- Introduction and variables: Variables are used to store information to be referenced and manipulated in a computer program.

X= 5

y=x

z=x+y

3- Math Operations: Operations in programming same as operation in math

X= 5

y=10

sum=x+y

mul=x*y

sub=x-y

Task: Develop Simple calculator

X= 5

y=10

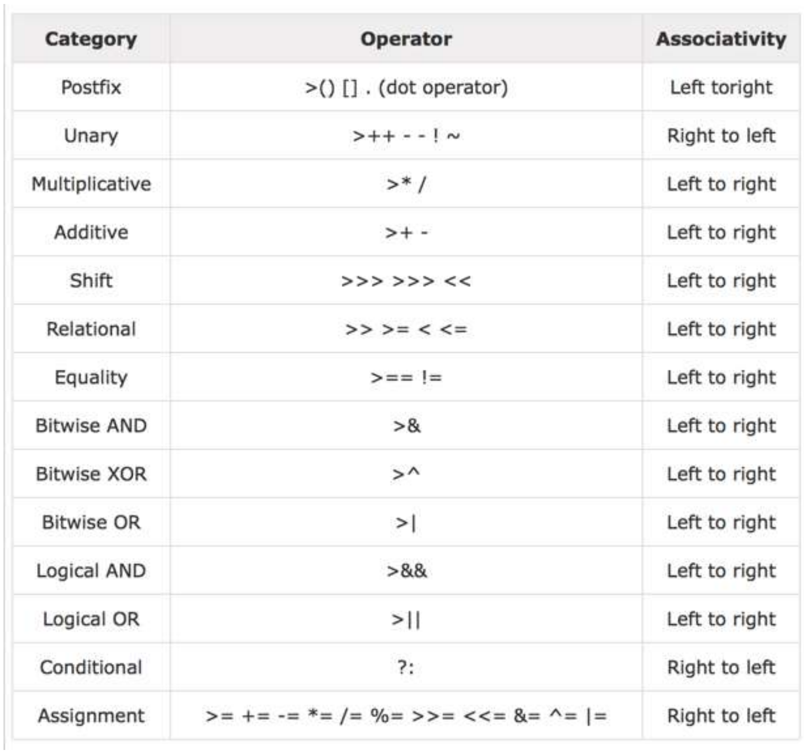
Sum=x+y

*Modify code to have other operations

4- Logic Operations

- (3>2)=true
- (4<5)=false
- (4>3)&& (6<8) =true
- (4>3)|| (6>8) =true
- !(true)=false

Priorities

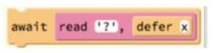


Category	Operator	Associativity
Postfix	>() [] . (dot operator)	Left toright
Unary	>++ - - ! ~	Right to left
Multiplicative	>* /	Left to right
Additive	>+ -	Left to right
Shift	>>> >>> <<	Left to right
Relational	>> >= < <=	Left to right
Equality	>== !=	Left to right
Bitwise AND	>&	Left to right
Bitwise XOR	>^	Left to right
Bitwise OR	>|	Left to right
Logical AND	>&&	Left to right
Logical OR	>||	Left to right
Conditional	?:	Right to left
Assignment	>= += -= *= /= %= >>= <<= &= ^= |=	Right to left

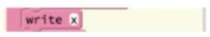
Example

- 3+10*5-10
- 5*2+¾

5- Input/output: read from input keyboard and print on screen



```
await read '?', defer x
```



```
write x
```

Task: Find your age in years

Await read '?', defer DOB

Age = 2017 - DOB

Write Age

Lecture 2: Loop and Condition

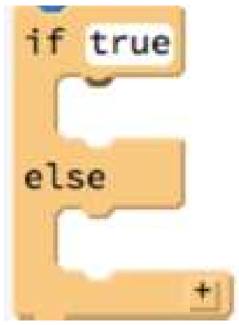
1- Condition

If (true)

// Run block 1 of code

Else

//Run block 2 of code

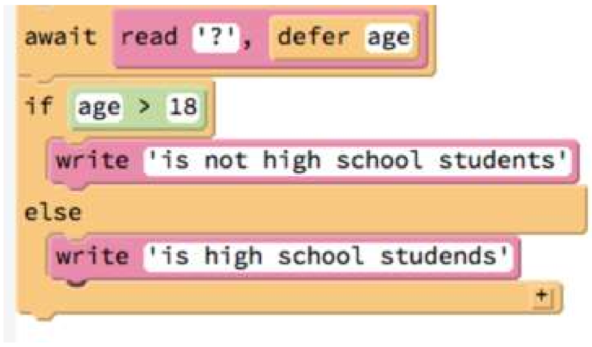


```
if true

else

```

Task: Find high school students



```
await read '?', defer age
if age > 18
  write 'is not high school students'
else
  write 'is high school studends'
```

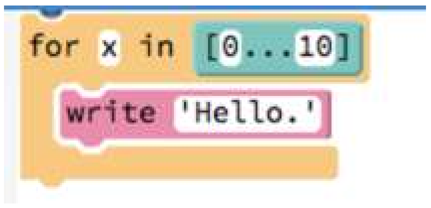
Task (TODO): Enter degree and show level

- >90 = A
- 80 -90= B
- 70-80= C
- <70 = fail

2- Loops

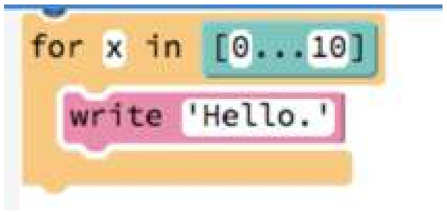
For x in [0...10]

// Run block of code number of times



```
for x in [0...10]
  write 'Hello.'
```

Task: print hello 10 times



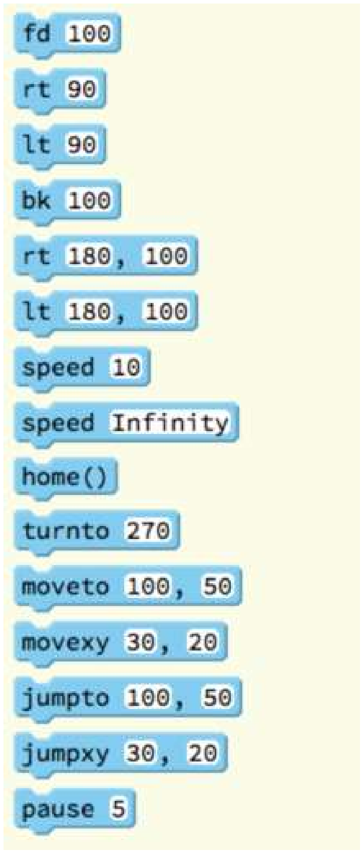
```
for x in [0...10]
  write 'Hello.'
```

Task: find x+x^x+x^3..

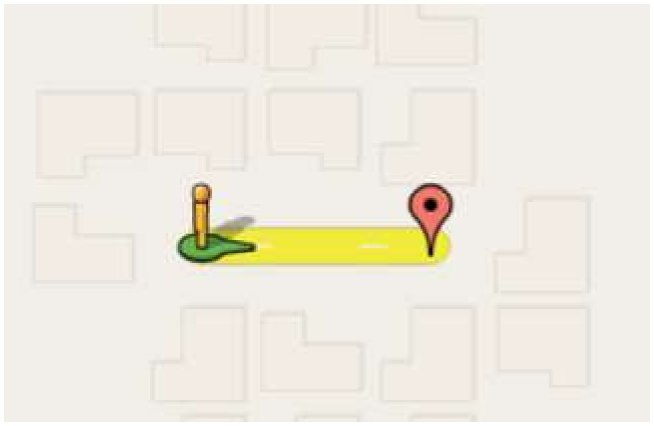
Task (TODO): Find class average

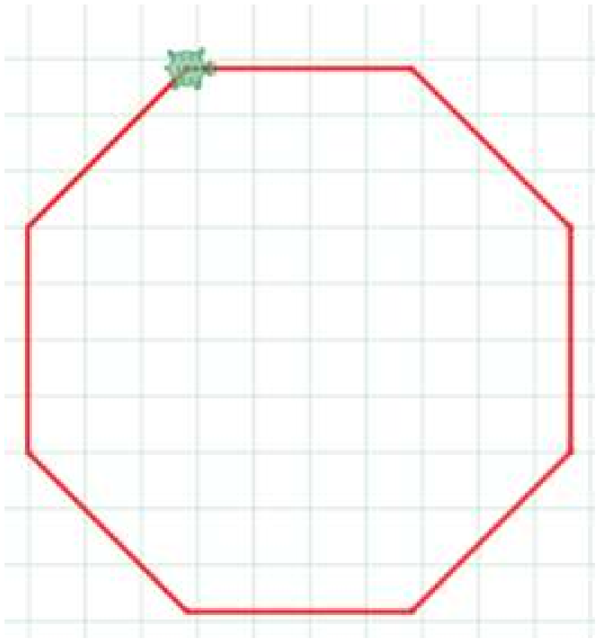
Lecture 3: Game

In this lecture, we will learn how to move Tortoise in specific paths



```
fd 100
rt 90
lt 90
bk 100
rt 180, 100
lt 180, 100
speed 10
speed Infinity
home()
turnto 270
moveto 100, 50
movexy 30, 20
jumpto 100, 50
jumpxy 30, 20
pause 5
```





Task 1: Simple moves

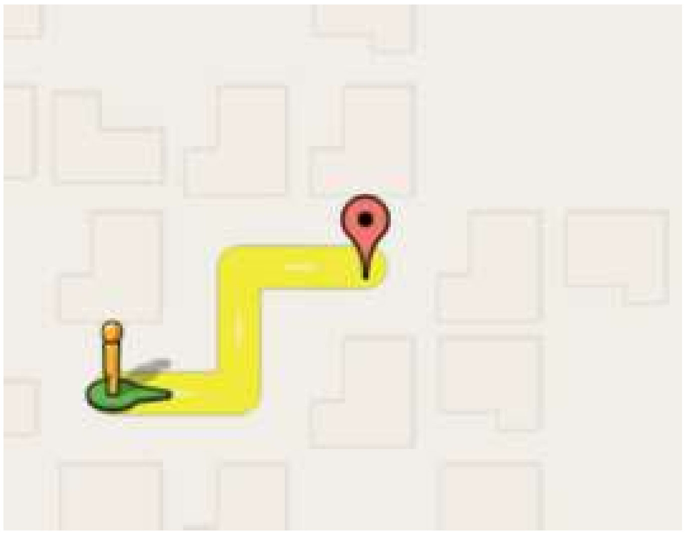
Speed 2

Pen red

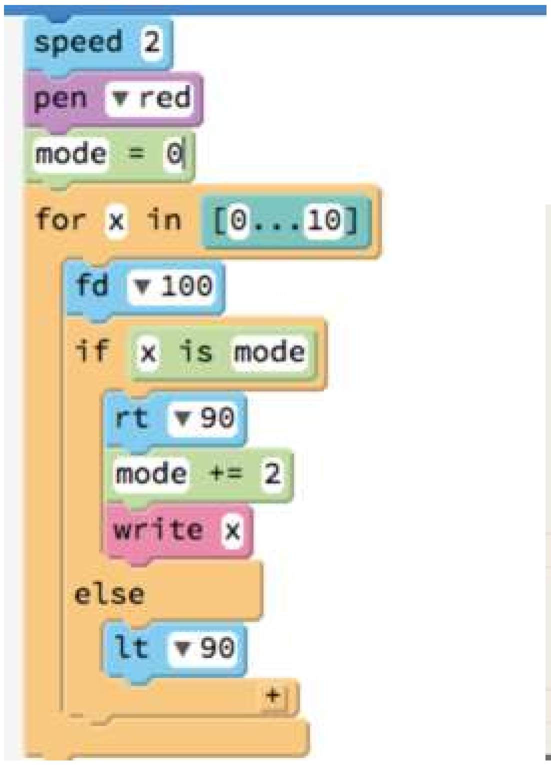
rt 90

fd 100

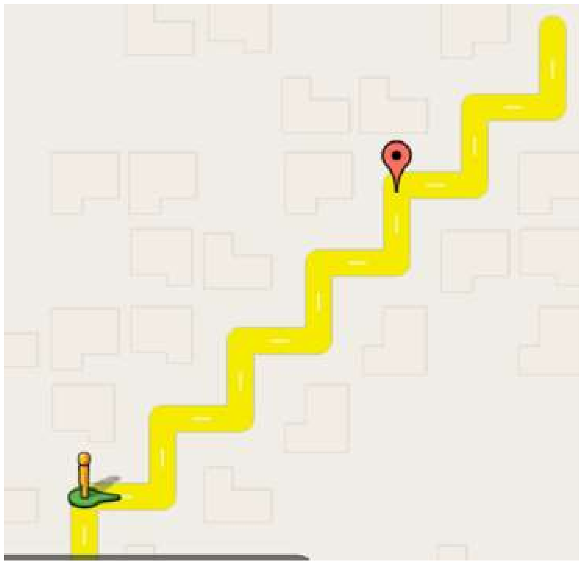
Task (TODO): Simple moves



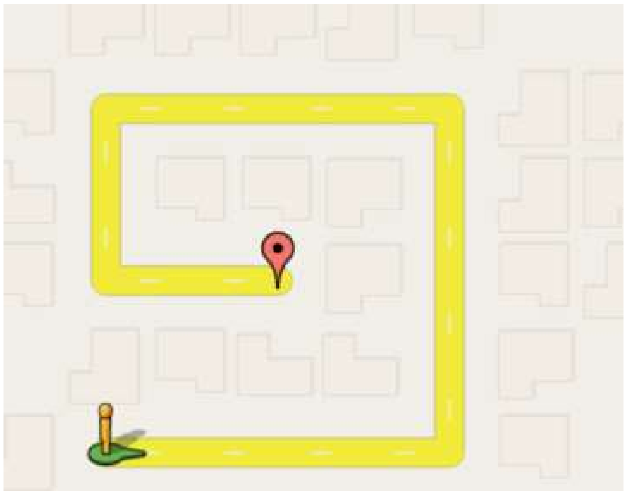
Task 1: moves need loop control



```
speed 2
pen ▾red
mode = 0
for x in [0...10]
  fd ▾100
  if x is mode
    rt ▾90
    mode += 2
    write x
  else
    lt ▾90
```



Task (TODO): move Tortoise on the route



Lecture 4 Text-based

This week both team will use text-based to do a similar task, that they already did in their environment.

CoffeeScript

- We will use CoffeeScript as text-based language.
- Do simple demo app on CoffeeScript
- Task|| Do simple demo app in Python?

Task (TODO)

1- Build calculator for summation. Then modify to find other operation

2- Find your age

3- Write app to find class average

4- Do the first example using function

Start Post-Test to evaluate the students

https://goo.gl/6SD3Zh

Appendix B

This Appendix contains the post-test questions that used to evaluate hybrid and block-based tool. Every student who enrolled in the experiment took this test.

Post- Student Evaluation Test

Comparison of visual programming and hybrid programming environments in programming transfer skills

* Required

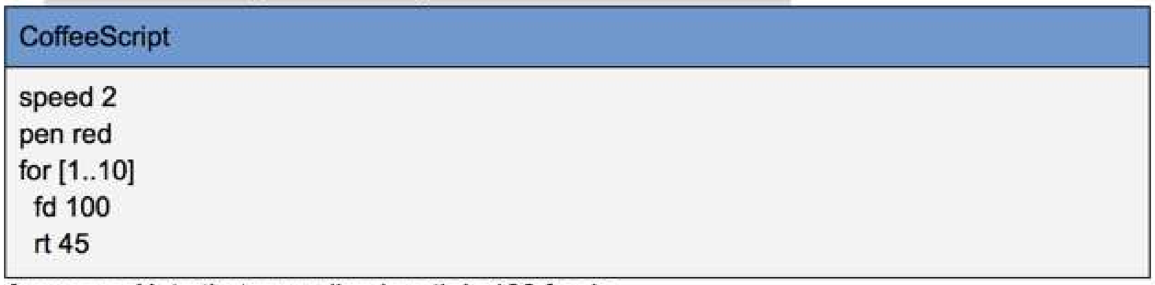
1. **Student email** *

2. **Your team name:**
Mark only one oval.
   - ◯ Team1: Study programming concept with Block-based
   - ◯ Team2: Study programming concept with Hybrid based.

Questions

Answer all question bellow

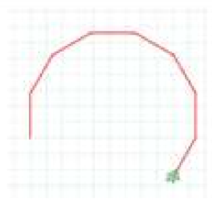
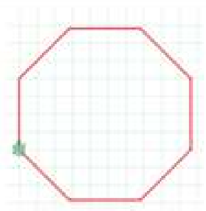
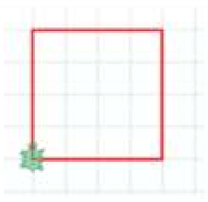
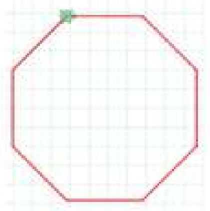
3. **1- For the following code what you think Tortoise will draw** *



CoffeeScript

```
speed 2
pen red
for [1..10]
  fd 100
  rt 45
```

Answers: Note that every line length is 100 feeds

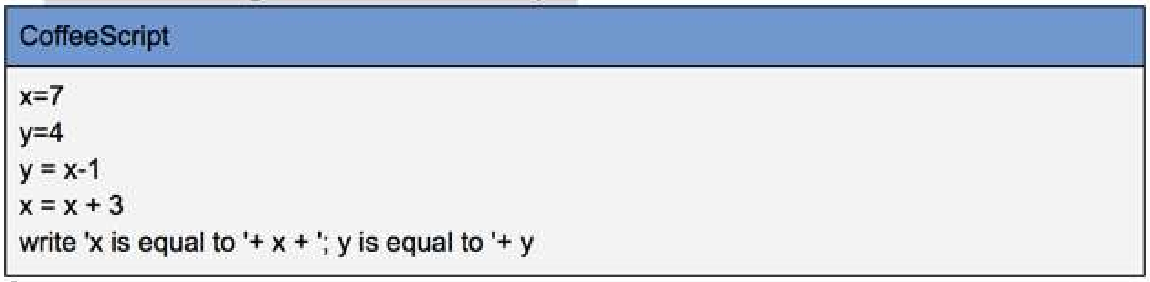
A	B	C	D
			

Mark only one oval.

- A
- B
- C
- D

Questions

4. **2- For the following code what is the output** *



CoffeeScript

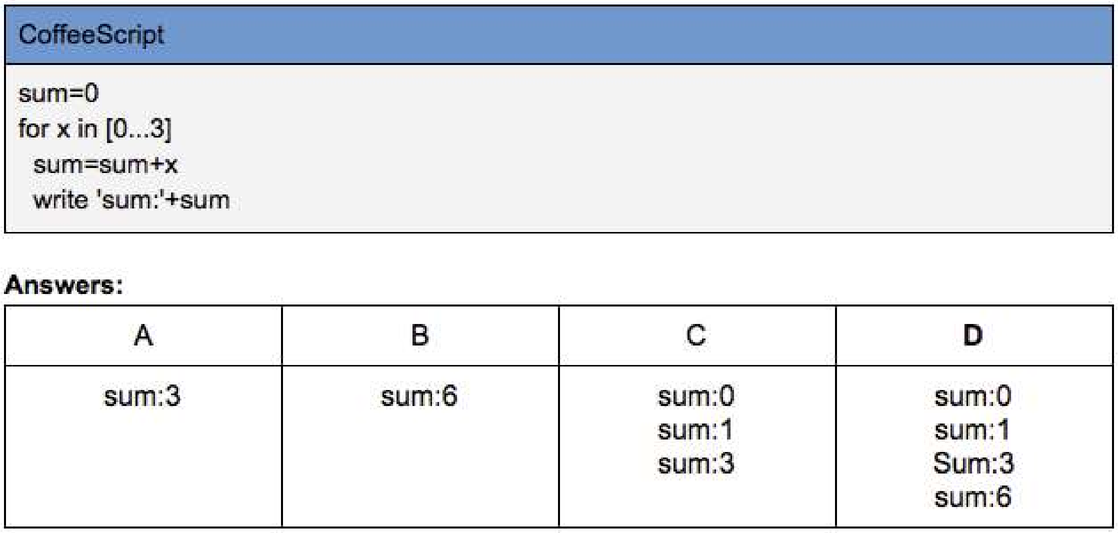
```
x=7
y=4
y = x-1
x = x + 3
write 'x is equal to '+ x + '; y is equal to '+ y
```

Mark only one oval.

- x is equal to 7; y is equal to 4
- x is equal to 10; y is equal to 6
- x is equal to 4; y is equal to 7
- x is equal to ‘x+3’; y is equal to ‘x-1’
- x is equal to 6; y is equal to 8

Questions

5. **3- For the following code what is the output** *



CoffeeScript

```
sum=0
for x in [0...3]
  sum=sum+x
  write 'sum:'+sum
```

Answers:

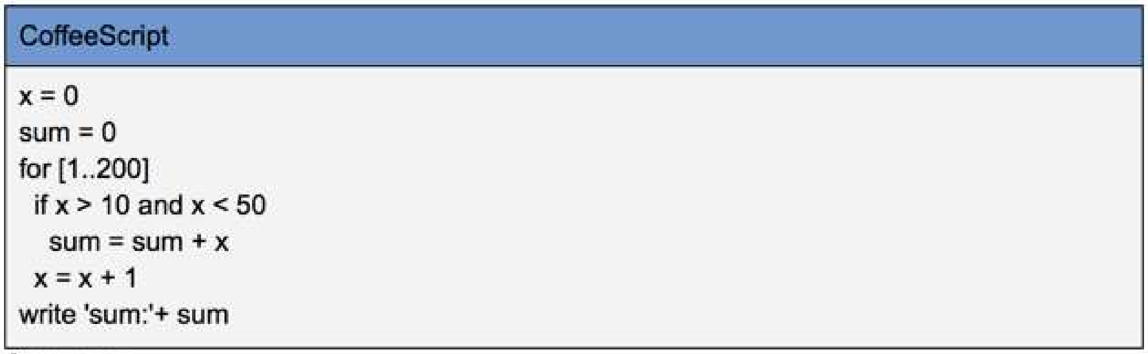
A	B	C	D
sum:3	sum:6	sum:0 sum:1 sum:3	sum:0 sum:1 Sum:3 sum:6

Mark only one oval.

- A
- B
- C
- D

Questions

6. **4- What does the script do?** *



CoffeeScript

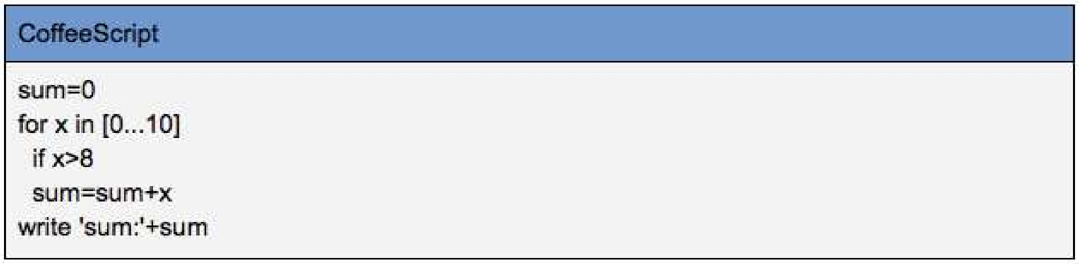
```
x = 0
sum = 0
for [1..200]
  if x > 10 and x < 50
    sum = sum + x
  x = x + 1
write 'sum:'+ sum
```

Mark only one oval.

- ⬭ Sum number between 1-200
- ⬭ Sum number greater than 10
- ⬭ Sum number less than 50
- ⬭ Sum number between 10-50

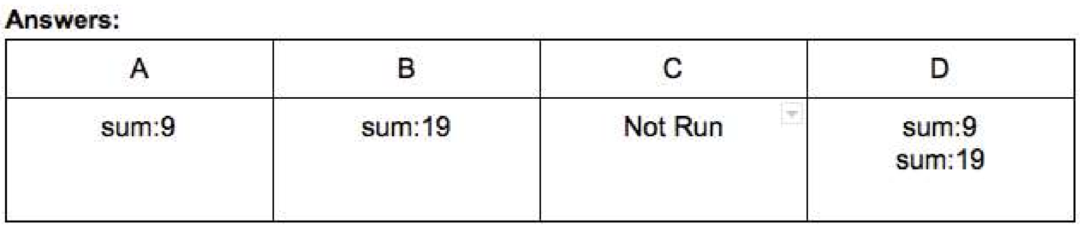
Questions

7. **5- What is the output?** *



CoffeeScript

```
sum=0
for x in [0...10]
  if x>8
  sum=sum+x
write 'sum:'+sum
```



Answers:

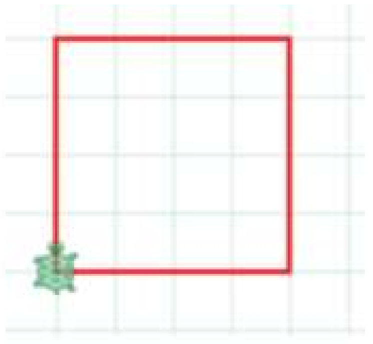
A	B	C	D
sum:9	sum:19	Not Run	sum:9 sum:19

Mark only one oval.

- ⬭ A
- ⬭ B
- ⬭ C
- ⬭ D

Questions

8. **6- Write app make Tortoise move on this route** *



Questions

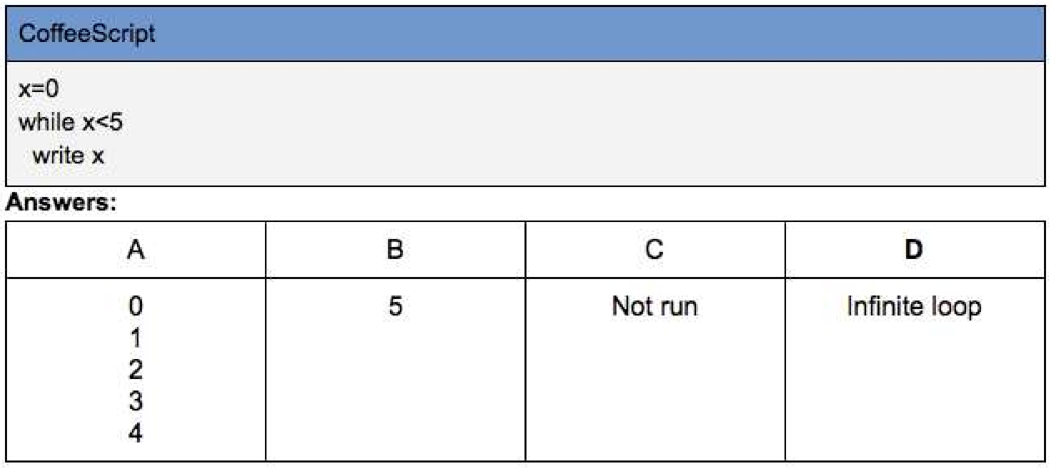
9. **7- If you want to write a program that asks a users to type in a sentence, then reports back to the users the number of times the letter 'e' appears in the sentence, which of these things would your programming language not need to able to do.** *

Mark only one oval.

- ◯ Store user entered information
- ◯ Display text on the screen
- ◯ Compare two letters to each other to determine if they are the same
- ◯ Convert letters into numbers and numbers into letters
- ◯ Create and modify data as a program runs

Questions

10. **8- What is the output** *



CoffeeScript

```
x=0
while x<5
  write x
```

Answers:

A	B	C	D
0 1 2 3 4	5	Not run	Infinite loop

Mark only one oval.

◯ A

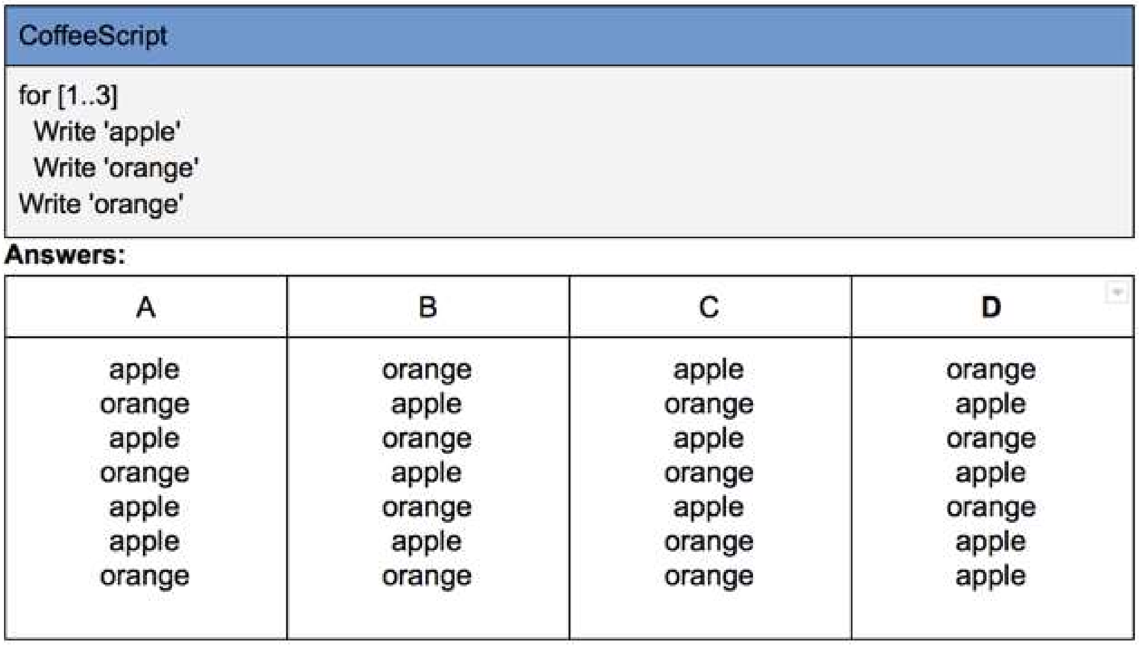
◯ B

◯ C

◯ D

Questions

11. **9- What is the output** *



CoffeeScript

```
for [1..3]
  Write 'apple'
  Write 'orange'
Write 'orange'
```

Answers:

A	B	C	D
apple orange apple orange apple apple orange	orange apple orange apple orange apple orange	apple orange apple orange apple orange orange	orange apple orange apple orange apple apple

Mark only one oval.

- A
- B
- C
- D
- No one

Questions

12. **10- What is the output** *

CoffeeScript

```
for x in [0...10]
  if x > 5
    write x
```

Answers:

A	B	C	D
0 1 2 3	5 6 7 8	6 7 8 9	6 7 8 9 10

Mark only one oval.

- ⬭ A
- ⬭ B
- ⬭ C
- ⬭ D

Questions

13. **11- What is the output?** *

CoffeeScript

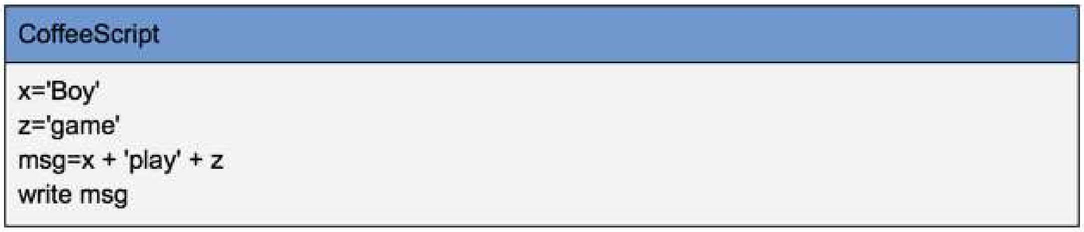
```
x=4
if x>10
  x=10
else
  if x<5
    x=5
Write x
```

Mark only one oval.

- ⬭ 4
- ⬭ 10
- ⬭ 5
- ⬭ Nothing
- ⬭ Error

Questions

14. **12- What is the output?** *



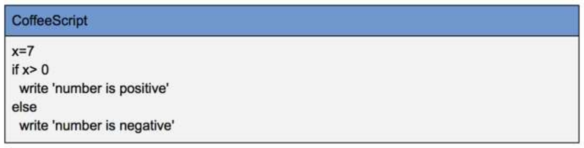
```
CoffeeScript
x='Boy'
z='game'
msg=x + 'play' + z
write msg
```

Mark only one oval.

- Boy play game
- play game boy
- Boyplaygame
- game play boy
- Error

Questions

15. **13- For the following code Answer the questions**



```
CoffeeScript
x=7
if x> 0
  write 'number is positive'
else
  write 'number is negative'
```

16. **I- What this code do** *

Mark only one oval.

- Check number if it accept division by zero
- Find if the number is positive or negative
- Avoid division by zero

17. **What is the output** *

Mark only one oval.

- number is positive
- number is negative
- Nothing

18. **What is the output if x= -12** *

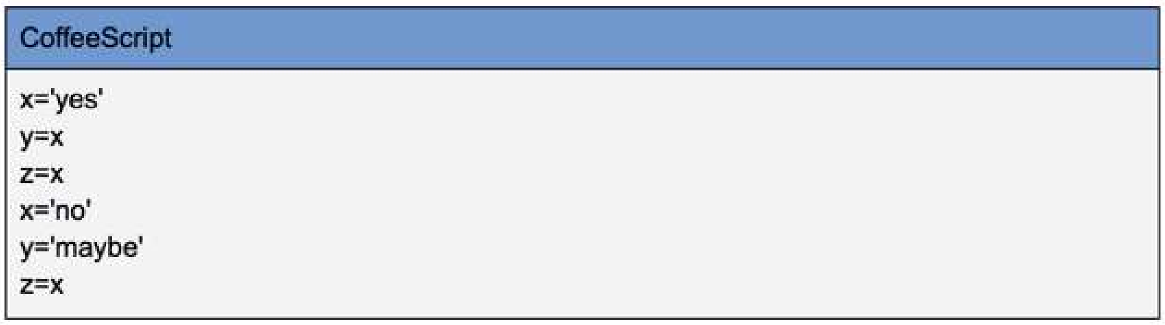
Mark only one oval.

- number is positive
- number is negative
- Nothing

19. **IV- Zero is unsigned number, but according to this code zero consider as negative number , update the code so when (x=0) the output will be ‘ number is unsigned’ . Also keep other functionality working correctly. (if x=0 as input then Output: number is unsigned)** *

Questions

20. **14- What is the output?** *



CoffeeScript

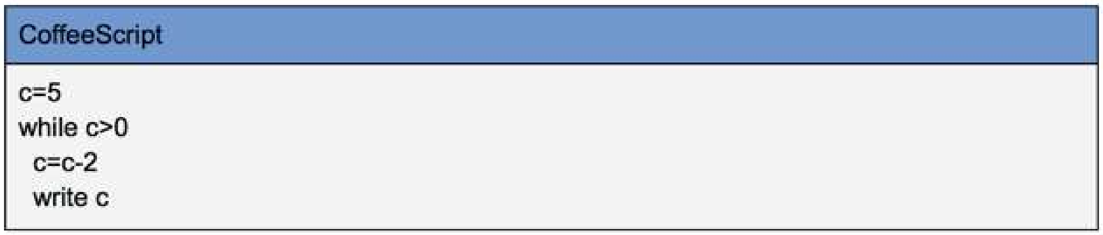
```
x='yes'
y=x
z=x
x='no'
y='maybe'
z=x
```

Mark only one oval.

- x is equal to 'yes, no'; y is equal to 'yes, maybe'; z is equal to 'yes, no'
- x is equal to 'no'; y is equal to 'maybe'; z is equal to 'yes'
- x is equal to ‘yes'; y is equal to 'maybe'; z is equal to 'no'
- x is equal to 'no'; y is equal to 'maybe'; z is equal to 'no'
- x is equal to 'no'; y is equal to 'maybe'; z is equal to 'x'

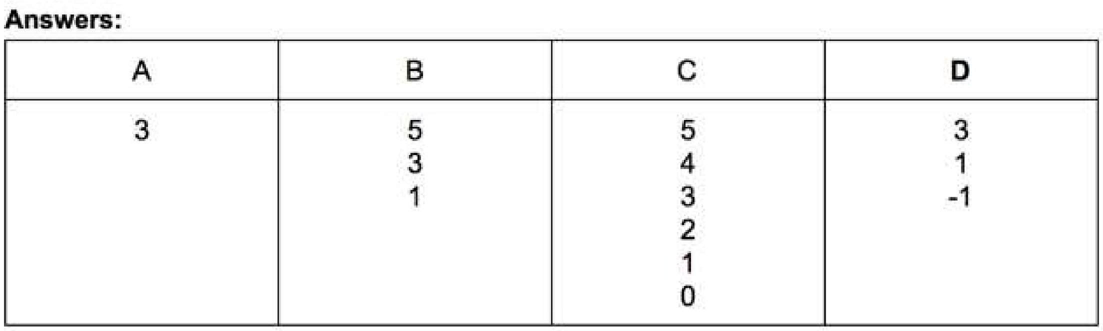
Questions

21. **15- What is the output?** *



CoffeeScript

```
c=5
while c>0
  c=c-2
  write c
```



Answers:

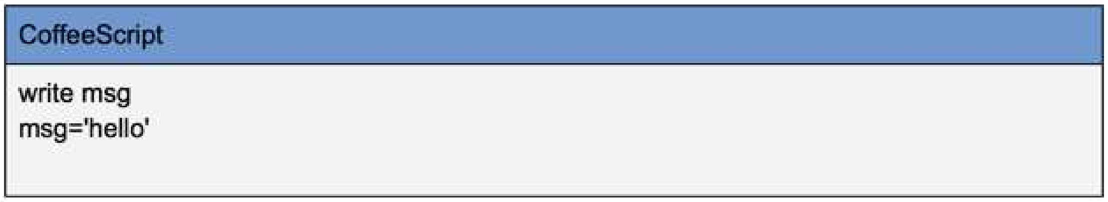
A	B	C	D
3	5 3 1	5 4 3 2 1 0	3 1 -1

Mark only one oval.

- A
- B
- C
- D

Questions

22. **16- Does the script Run? Why** *



CoffeeScript

```
write msg
msg='hello'
```

Mark only one oval.

- Yes
- No

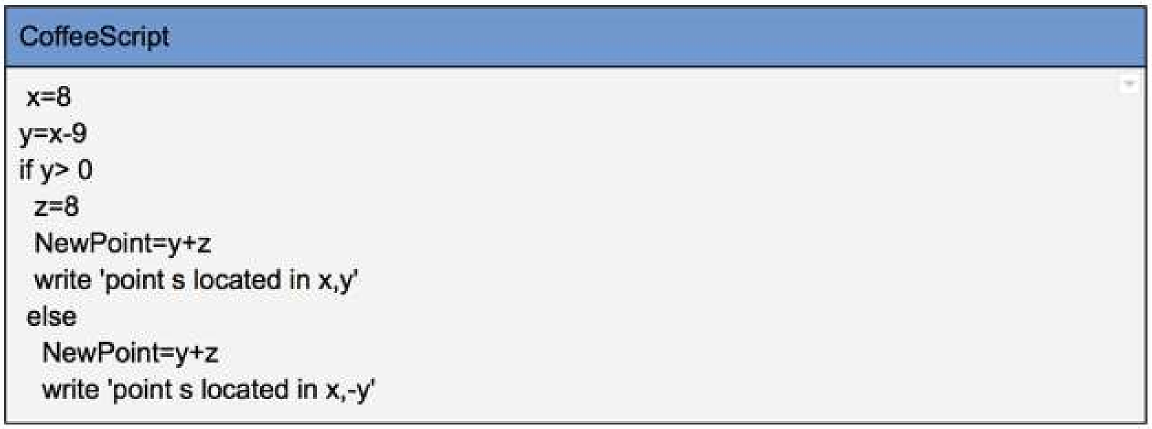
23. **IF you select no, how it should be to run**

Questions

24. **17- Write app that compare two numbers and print the greater one. Example if n1=5 and n2= 7 it will print n2.** *

Questions

25. **18- Does the script Run? Why** *



CoffeeScript

```
 x=8
y=x-9
if y> 0
  z=8
  NewPoint=y+z
  write 'point s located in x,y'
 else
   NewPoint=y+z
   write 'point s located in x,-y'
```

Questions

26. **19- Did you had hard time to understand loop, condition, and functions in code-based environment(5 star is good 1 star is bad)** *

Mark only one oval.

- 5 Star
- 4 Star
- 3 Star
- 2 Star
- 1 Star

Questions

27. **20- Are you able to memorize the command of loop, condition, and functions, when you start with in code-based environment(5 star is good 1 star is bad)** *

Mark only one oval.

- 5 Star
- 4 Star
- 3 Star
- 2 Star
- 1 Star

Questions

28. **21- When you started writing code in code-based environment. Did you feel you are in new development environment, or the commands for loops, conditions, and functions was familiar to you.** *

Mark only one oval.

- Yes, it looks new
- No, it looks familiar