Code Quest – Playing, Learning, and Teaching Guide

For teachers, instructors, players from the Code Quest and Gaming4Coding project team.

Co-funded by the European Union

A joyful game for learning fundamental programming

The core idea in the Code Quest game is that players should collect the cute fantasy creatures that are called 'critters'. Players can train the critters by giving them instruction scripts build by real programming code. However, the training is purposefully not called programming in order to give the game more of a playful, and less of a technical feeling. The main gameplay consists of collecting interesting critters, training them for racing and entering them into a race against other players. Races are held on different obstacle courses where the critters compete either against bots or against other players. Training and racing courses are set in different environment such as on a beach.

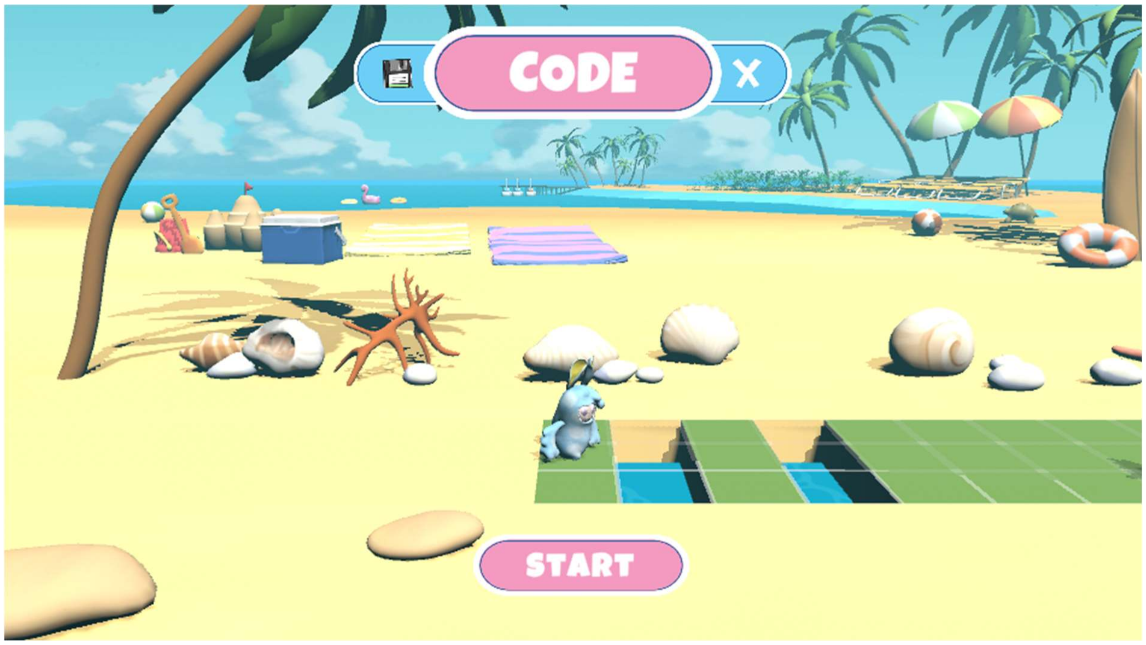


Figure 1. The Dream Beach course

A monster taming game with multi-player options and built in learning analytics

Many educational games for learning how to program are role-playing games (RPGs) built around puzzle mechanics. An RPG is a game in which participants assume the role of a character that can interact within an imaginary game world. Less games are designed to have a focus on training and collecting creatures. The game genre characterised by training and collecting creatures is called Monster-taming game, inspired by the popular Pokémon saga. Another under-explored feature of games for developing computational thinking and learning programming is the multiplayer design. Code Quest implements this option, allowing players to play against either against the computer or against each other. This dual design allows researchers to study learning outcomes, gaming behaviour, and the effects of competition during learning in the field of learning programming.

The game is designed and developed around the 4 pillars of:

1. **Collecting**: The activity of collecting things has a large appeal as a game element, especially when the player can collect creatures. The most prominent example of this is the Pokémon game. Since new creatures are awarded for winning races or for other achievements the interest in collecting critters has a strong impact on the motivation of the players to get better at coding, so that collecting new critters becomes easier.

2. **Rewards**: Rewards are very important for player motivation. The game rewards victories in races so that players get something out of having created well-working training scripts for their critters. However, the game also rewards personal achievements, even if the player doesn't end up winning the race. Thus, rewards play an important role in keeping the motivation high for any learning and gaining of experience. The biggest rewards are new collectable critters, smaller rewards are special items for the critters and trophies.

3. **Competition**: The competition element is a well-established way to challenge a player to become better. The game uses a mixture of players and bots as competitors so that the player always has the opportunity to win against someone, even if it is too hard to win first place. Competition should also give players an opportunity to learn from others.

4. **Matching Difficulty**: The game uses a smart algorithm to assign competence levels to players in order to match them with other players or bots that are performing at comparable levels. This avoids having a beginner crushed by a very advanced player.

Being able to choose between different creatures to play with, compete with other players, complete challenges and discover new blocks of code to progress with are intended to be the motivating engine that keeps players interested and motivated.

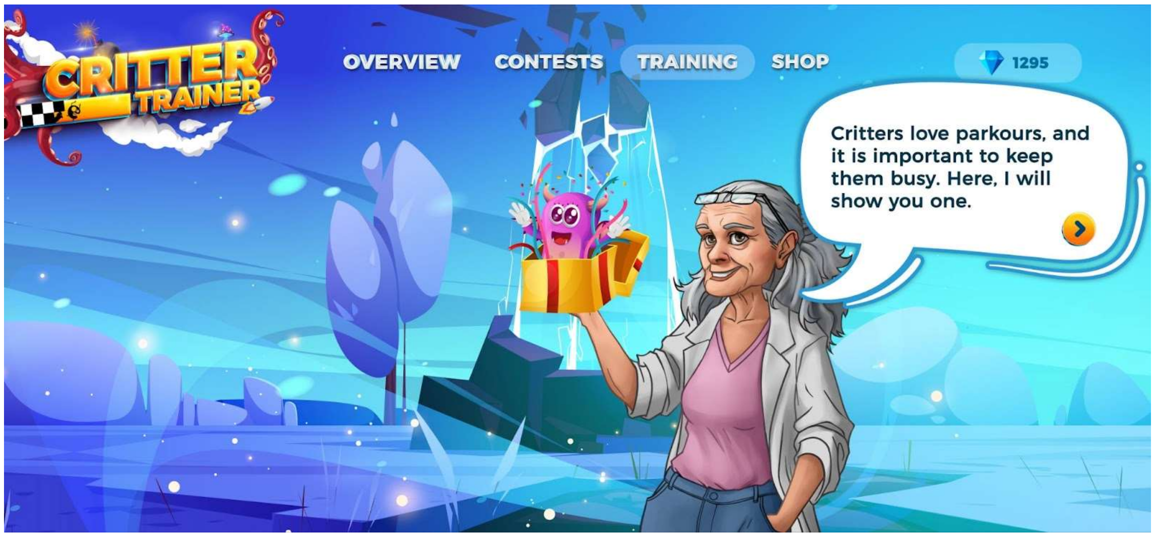
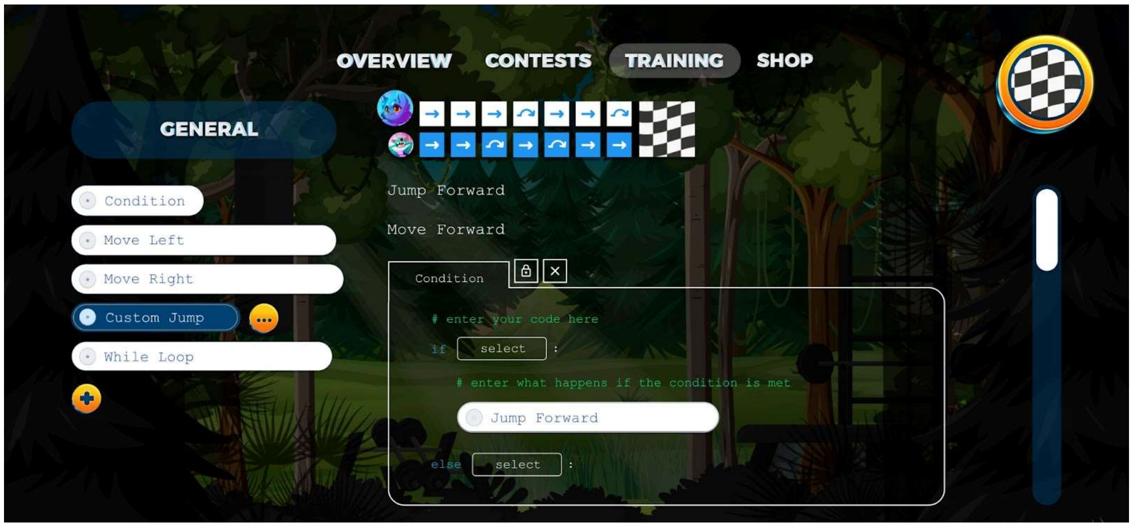

CRITTER TRAINER
OVERVIEW
CONTESTS
TRAINING
SHOP
1295
Critters love parkours, and it is important to keep them busy. Here, I will show you one.

Figure 2. Training critters by writing code

In a race the player only has limited control over the critters as they are mainly in the role of a spectator. Winning or losing a race has a lot to do with which critter is selected for the race track and which commands are given to the critter in the ‘training script’. This is where it is most important to get new creatures and train them to fit the different circuits. When training a critter, a player writes down a sequence of commands which the critter will execute in the race. This could start in a very simple way (“run”, “run”, “run”, “jump”) and then move to more advanced coding structures with while loops and conditionals. The game never pushes the player to advance in coding skills, however the competitive aspect and the rewards for winning races are a huge motivation to improve the performance of the critters, thus leading to players trying out more advanced code.

Another feature is that the view where the code is implemented mixes both a more visual part, of predefined code blocks, with a textual part that follows the Python syntax. Python syntax was chosen for the coding elements since Python is one of the most popular programming languages which is also widely recommended for school environments in Europe. It is also easy to learn. In the actual game the players first learn how to give critters commands in order to guide them through an obstacle course. Commands correspond to functions in Python but the players will be using predefined functions from the beginning (such as run, dodgeLeft, dodgeRight, jump, etc.) before learning how to create their own functions.



```
Jump Forward

Move Forward

Condition
    # enter your code here
    if  select  :
        # enter what happens if the condition is met
        Jump Forward
    else  select  :
```

Figure 3. An advanced level in the Code Quest game

Sections and levels of training mode

In this game the players first learn how to give critters commands in order to guide them through an obstacle course. Commands correspond to functions in Python but the players will be using predefined functions from the beginning (such as run, dodgeLeft, dodgeRight, jump, etc.) before learning how to create their own functions. The game starts with tutorials which challenge the player to solve a situation, so they are essentially little puzzles that help the player acquire the skills needed. The tutorials gradually introduce the player to the competitive racing game in which more critters can be unlocked and rewards can be won. Tutorials that are recommended to be used as parts of lessons and workshops in programming education.

Players with earlier experience of fundamental programming could skip the tutorials and immediately start training their critters and enter them into races. However, in educational settings the strong recommendation is to start with teacher led sessions, where lessons or workshop activities could be built around the learning sections and levels that are presented here below.

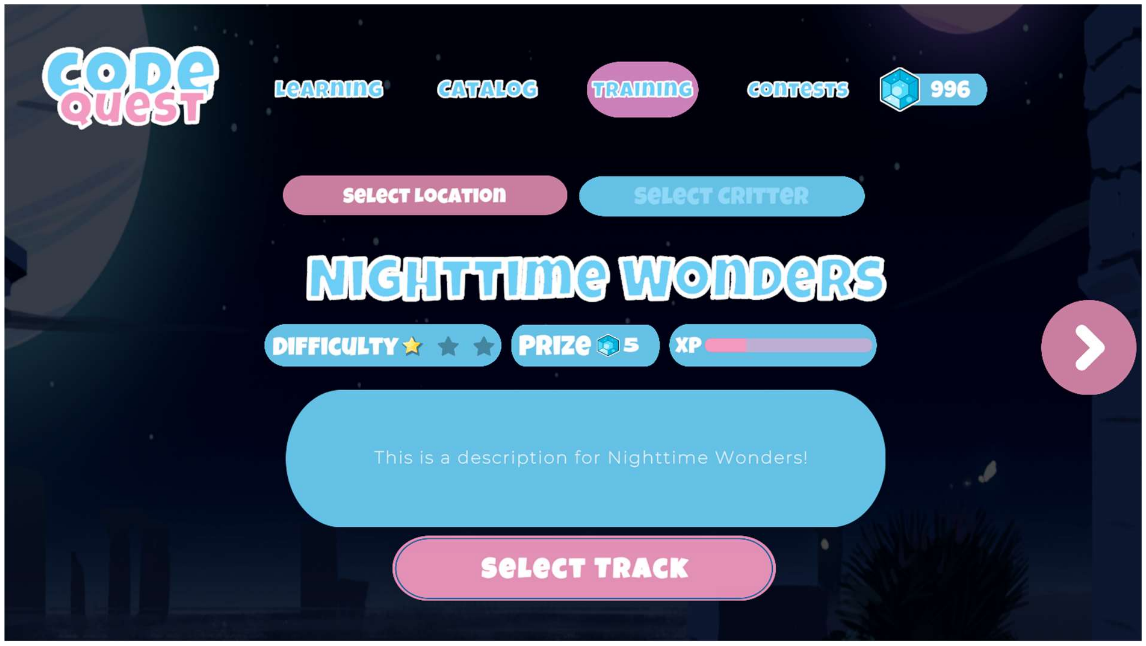


Figure 4. The training track selection

Section 1: The Game Environment and the Race Tracks

The game has been designed with the idea of being intuitive and with a low initial threshold. However, to meet the aim of an educational game for all this section can be helpful to get all learners aboard and feel comfortable in the Code Quest environment. As for all other sections players could be divided into groups where the players with more experience of gaming and coding could help the novices.

Level 1: Getting used to how race tracks work

Training mode starts out nice and gently with only one lane, with only one button to make the critter walk. Here in Level 1, the player is introduced to taking care of a critter and moving around in the game world. The game interaction starts with only a simple button to carry out the 'Move forward' command.

Level 2: Learning which individual track types the critter needs to perform different tasks

In the next level a water pit is added, but still with only one lane in the track. Here the player is introduced to having several possible moves, when a ‘Jump forward button’ has been added.

Section 2: Patterns and Repetitive Tasks

From this section, and through the rest of the training mode, the player uses the game's coding menus instead of just simple buttons. Patterns and repetitive task are fundamental as well as very useful building blocks in both programming and computational thinking.

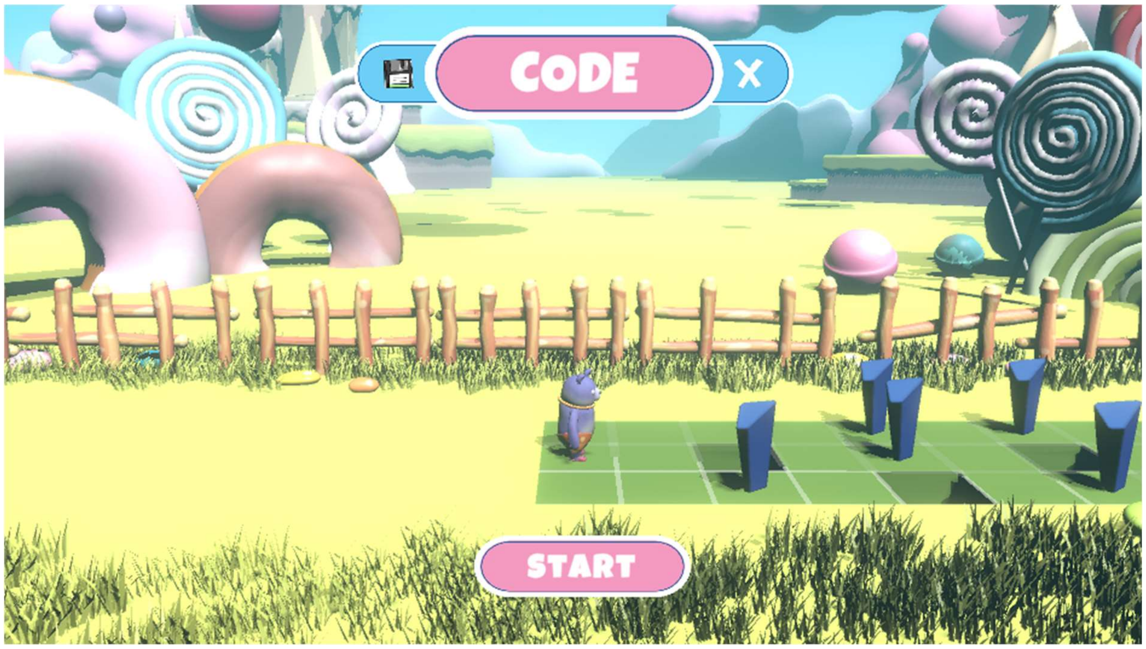


Figure 5. A track with a coding menu

Level 1: Understanding repetition

A strength of a computer system is the ability to repeat simple instructions very fast. Instead of pushing a button 10 times to walk 10 steps forward, the instruction can be repeated 10 times in a repetition instruction. Here in Level 1, the user has not learnt about the for loop yet, so s/he must use the walk forward coding element 10 times.

Level 2: Understanding that a for loop reduces the workload in repetitive tasks

In this level the 10 step of walking forward, is executed with a for-loop. A programming construction that is a fundamental part of all modern programming languages. *The for-loop coding element has been added and the player is reminded of how s/he had to use the same coding element 10 times in the previous level. With the difference that the 10 repetitions now can be executed in a for-loop.*

Level 3: Understanding the timing of for-loops

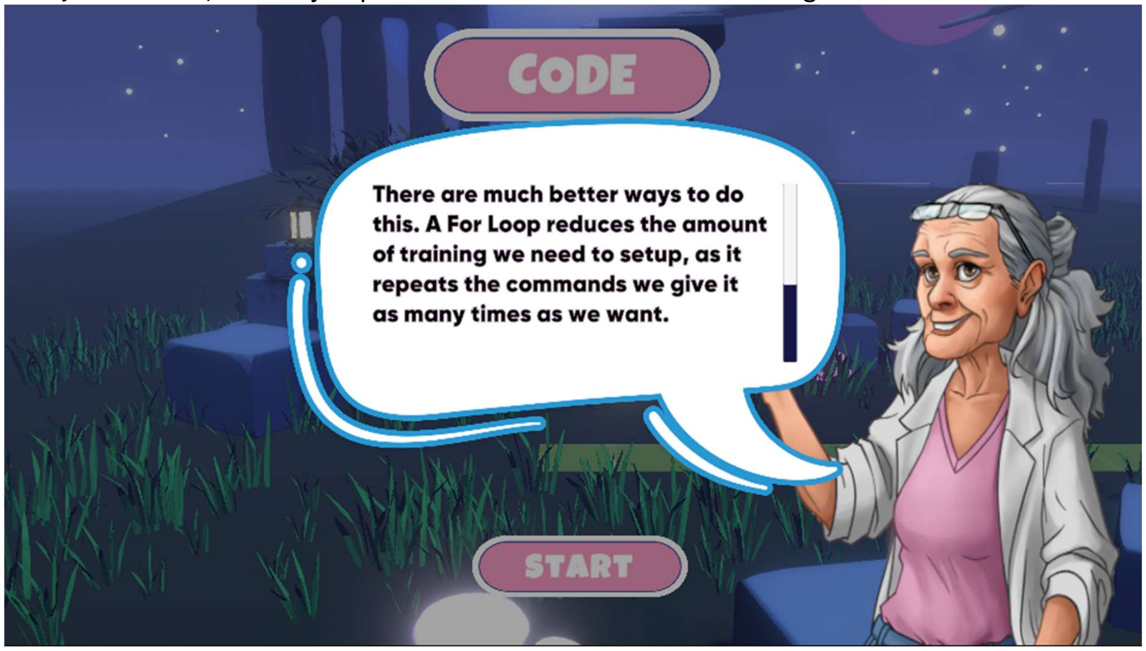
Repetition or, as it is called in Computer science iteration, could be used for more than one instruction. As an example, an iterated pattern in the Code Quest game at this level is to for 6 iterations, walk forward, then jump and then walk forward again. This should better be explained in a bit longer activity since this for-loop is the most complex instruction in the training guide so far.

Level 4: Repetitive Patterns

The training proceeds at this level with a slightly more in-depth use of for-loops, using several types of instructions in the same loop. The repeated pattern here is to walk forward and to jump, which is executed 4 times in a row from a for-loop.

Level 5: More Complex Repetitive Patterns

An even more in-depth practice of for loops, with the use of several types of coding elements in the loop, and also with other code instructions outside of the loops. The training active is structured as repeat 4 times in a row the execution of: {move forward 2 times and jump one time}. Moreover, do one jump at the end to reach the desired target.



There are much better ways to do this. A For Loop reduces the amount of training we need to setup, as it repeats the commands we give it as many times as we want.

Figure 6. Learning about for-loops and repeated patterns at 5 levels with increasing difficulty.

Section 3: Boolean conditions and selection

Several programming constructions such as iteration and selection are controlled by conditions based on Boolean binary algebra. The algebra part has been simplified to suit the target audience, but without conditions and selection, no advanced critter training scripts. Selection is, in almost all programming languages implemented with if- and else-clauses. As all other code in the Code Quest game the involved code is given, and entered, with executable Python code.

Level 1: Understanding conditions and the if-statement

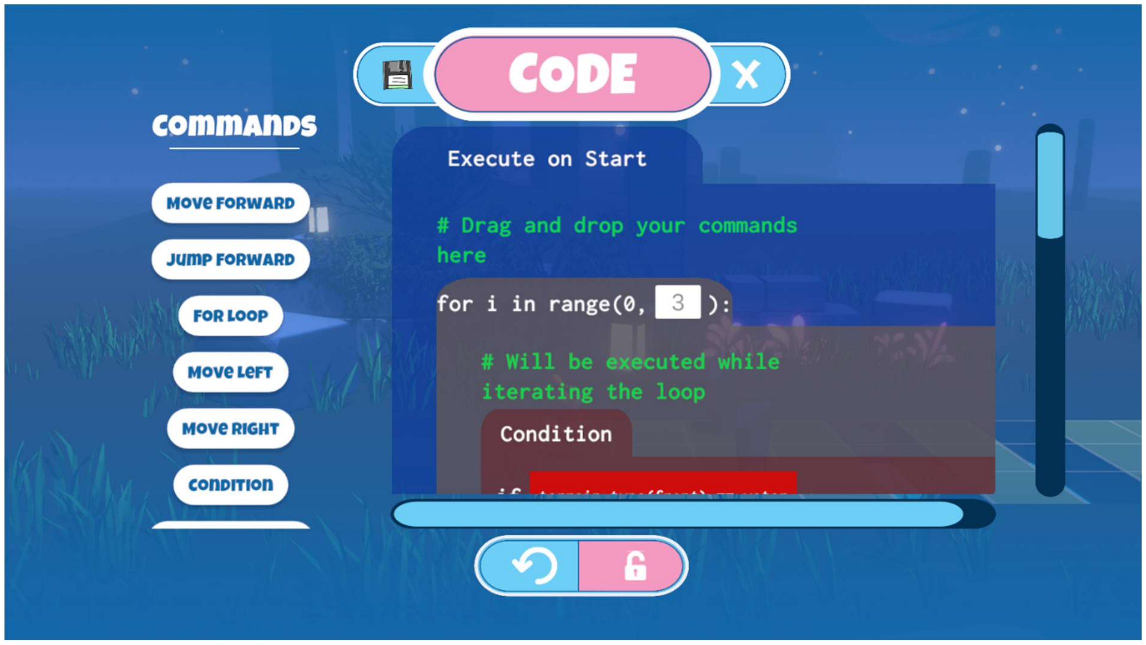
This activity involves the creation of a condition that checks if your next tile type is a pit or ground. The critter should only move forward, **if it is a grass tile**. Otherwise, do nothing. In this first level the track is only 1-2 units long, and the player is instructed to use this new coding elements, the conditions and an if-statement to move forward in the given track.

Level 2: Understanding the else-statement

The if-statements best friend is the complementing else-statement. At this level the instruction from the previous level is extended as: Create a condition that checks **if** your next tile type is a pit or ground. Move forward **if it is a grass tile**. Otherwise, **if it is a pit**, do a jump, or **if it is ground**, do a walk command. The player should use both the if and the else part of a condition that leads to different execution paths.

Level 3: Conditions in a for loop

As mentioned earlier conditions are also used to steer repetition structures such as a for-loop. Here at Level 3 code instructions should be created for a script like: **for 5 times**, check the next tile type and **jump if it is a pit** or **walk if it is not**. Now the player must use all previous knowledge from the earlier sections and levels to complete this level. Moreover, this is the first track that is randomized. Step by step the difficulty level has increased and the player gets more blocks and skills to create winning critter scripts.



```
Execute on Start

# Drag and drop your commands
here

for i in range(0, 3 ):

    # Will be executed while
    iterating the loop

    Condition
```

Figure 7. Combining selection, iteration and conditions in advanced training script

Section 4: Variables and debugging

Without the important component variables, the scripts will never be particularly useful. For an older target group with a more mathematical background, the variable concept is one of the first things to introduce. In this game, where the actual coding is a bit hidden, the choice was to put this important component here in Section 4. Moreover, we find it suitable to combine activities on variables with the introduction of debugging. To trace errors in code by checking values of variables is an important part of computational thinking and programming.

Level 1: Understanding variables and the debugger

This level starts out easy by initialising a variable, and for 10 times, to walk forward and add one to the variable. The debugger shows the current state of the variable at all steps and the updated value is visualised. Variables are gently and clearly introduced and their values should be followed and checked by the debugger. While the code is running the debugger shows the values of the variables that the player uses. This is carried out simple and straightforward here in Level 1, to provide the basic understanding for more complex constructions in the following levels.

Level 2: Checking a variable in a loop

The next level is about the combination of a variable and a loop in the steps of.

1. A variable is initialised and set to the value of 5
2. A while loop is created with the variable as its iterator
3. The while loop should be executed until the variable is 0
4. Inside the while loop the variable is reduced by one in every execution step
5. At the same time the loop should move a critter one step forward
6. The player is instructed how to use a while loop with a variable to reach the end of a track

Level 3: Constants, variables, conditions and loops combined

Besides of variables a programmer sometimes also uses constants in the code. Both are used together with conditions to control steering structures such as while and for loops. The instructions for Level 3 are:

1. Initialise and set a value to a constant with the name 'trackLength'
2. Initialise and set a value to a variable that tracks the square that the player is in
3. Create a while loop that runs until the critter reaches the end of the track
4. Inside the while loop, check if a move action or a jump is needed
5. If critters need to jump, carry out that action and add 2 to the variable, otherwise move forward and add 1

As in earlier sections, this last level acts as a checkpoint where the player must use all previous learnt knowledge and skills.

Section 5: Multiple Lanes

This is the section that presents the full complexity of the game idea, and how more advanced code should be constructed to solve more complex critter training tasks. From here and onwards the tracks involves more than one lane, with possibilities to switch lanes.

Level 1: Understanding that it is possible to switch lanes

The player should Move Left one time as there is an obstacle that can't be passed (such as 3 pits in a row). Build a while loop to walk forward until the track ends, with the use of the track length constant. Every time when moving forward, add one to a variable. This is a simple level where the player gets used to seeing and using several lanes.

Level 2: Maze Runner

Build a while loop that moves forward until the track ends. Use track length constant as in Level 1. If there is an obstacle in front such as a stone block that can't be passed in any way, check if it is possible to move left or right. Depending on the result, move left or right, and if nothing is in front of the critter, just move forward. Every time when moving forward, add one to a variable. This level is in a maze where the player must be comfortable with coding and moving in between lanes to complete the task.

Level 3: Loops inside Loops

To write code with loops inside loops is a frequently used technique in programming, but not that easy to understand in detail. This is carried out in this level in a race track on which the critter has to:

1. move all the way to the right
2. then 2 steps forward,
3. then all the way to the left
4. then two steps forward
5. then all the way to the right again

The two loops should be coordinated as:

1. The outer loop counts how far the critter has come and moves the critter forward
2. The inner loop moves critter either to the right or to the left until it reaches the edge

A complex level where the player must be fully aware of the effects of the coding in order to reach the end, using various techniques from all the previous sections.

Section 6: Simple Functions

Functions in Python, or any other programming language, is much the same as coordinating instructions in a mathematical function. Once a function is completed and tested, it can be reused in other scripts for solving a specific task. A lesson learnt is that the function concept in programming is a bit hard to grasp and use for beginners. For that reason, the concept was placed here in Section 6.

Level 1: A Simple Function

As always, the first level starts out nice and gently, and here it is about defining a simple function that lets you jump, move, jump and jump. Building on techniques learnt in the previous section:

1. Write a while loop that lasts until the track ends (use the track length constant)
2. In the loop, always move until you find a pit.
3. If you find a pit, execute the function

Level 2: Side to Side Functions

This level gets a bit more complex when several collaborating functions should be defined:

1. Define a function that lets you moves you all the way left
2. Define a function that moves you all the way right
3. Write a while loop that lasts until the track ends (use the track length constant)
4. In the loop, always move until you find a pit
5. If you find a pit, jump over it
6. If you find a stone obstacle, use one of the movement functions
7. At the end of the function, if you’re still in an obstacle, use the other function

Section 7: Character Ability introductions (Tim elaborates here)

Level 1: Critter Abilities

This level introduces the fact that critters have special abilities and how that affect the gameplay. As an example, players are informed how the critter Frogette can navigate. This illustrates that players can finish the level without jumping, and just by walking on water.

After training comes gaming

Code Quest is built around the idea of combining the initial learning with joyful gaming where programming skills should be practiced.

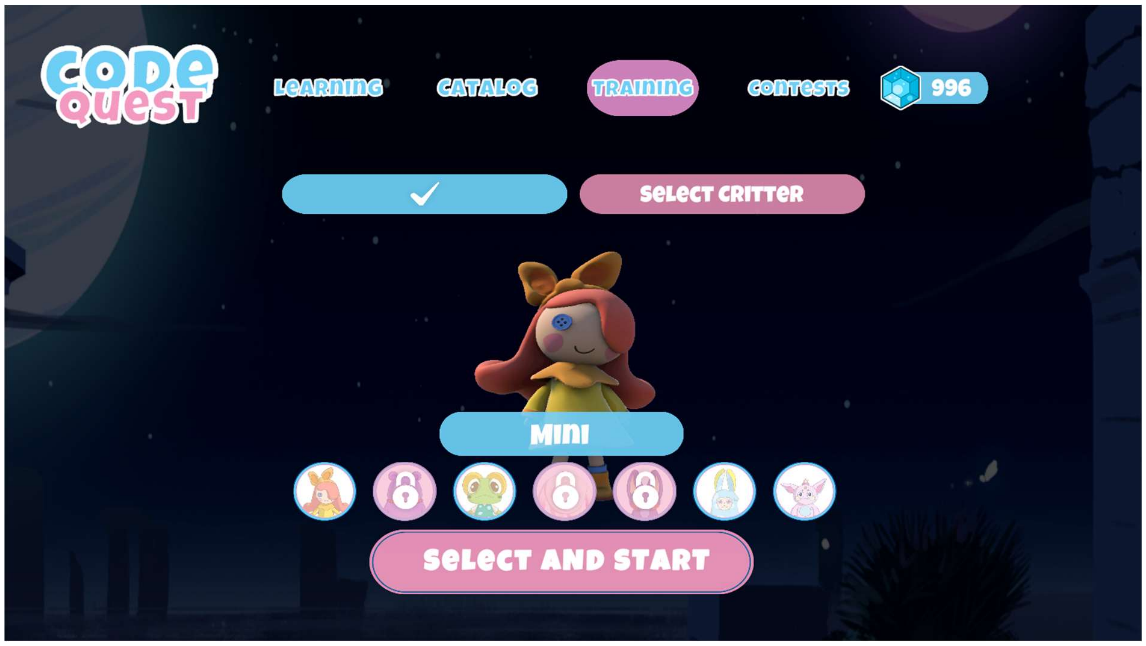


Figure 8. After training comes gaming

Learning to program should be fun, and the training sections should better be combined with playing sessions. Players would then understand the connection between training, coding skills and successful gaming. The exact mix and progression tempo will always be depending on both the learner group and the individual players. Some players learn fast and develop a strong passion for programming. As a teacher or instructor, you will probably get questions about coding details or programming syntax that can be hard to answer. There are many good resources on the Internet for programming and Python code examples. One that has been thoroughly developed and covers both fundamental and advanced programming in Python is the one on: https://python-course.eu/python-tutorial/

Happy Gaming, Happy Coding!

/The Gaming4Coding Team

Co-funded by the European Union

	Kultuuri Rakendusameti (EACEA) seisukohti ja arvamusi. Euroopa Liit ega EACEA nende eest ei vastuta.
FI	Euroopan unionin rahoittama. Esitetyt näkemykset ja mielipiteet ovat ainoastaan tämän tekstin laatijoiden näkemyksiä eivätkä välttämättä vastaa Euroopan unionin tai Euroopan koulutuksen ja kulttuurin toimeenpanovirasto (EACEA) kantaa. Euroopan unioni ja EACEA eivät ole vastuussa niistä.
FR	Financé par l'Union européenne. Les points de vue et avis exprimés n'engagent toutefois que leur(s) auteur(s) et ne reflètent pas nécessairement ceux de l'Union européenne ou de l'Agence exécutive européenne pour l'éducation et la culture (EACEA). Ni l'Union européenne ni l'EACEA ne sauraient en être tenues pour responsables.
GA	Arna mhaoiniú ag an Aontas Eorpach. Is leis an údar/na húdair amháin na tuairimí agus na dearcthaí a léirítear agus ní gá gur léiriú iad ar thuairimí agus dearcthaí an Aontais Eorpaigh nó na Gníomhaireachta Feidhmiúcháin Eorpaí um Oideachas agus Cultúr (EACEA). Ní féidir freagracht a chur ar an Aontas Eorpach ná ar an EACEA astu.
HR	Financirano sredstvima Europske unije. Izneseni stavovi i mišljenja su stavovi i mišljenja autora i ne moraju se podudarati sa stavovima i mišljenjima Europske unije ili Europske izvršne agencije za obrazovanje i kulturu (EACEA). Ni Europska unija ni EACEA ne mogu se smatrati odgovornima za njih.
HU	Az Európai Unió finanszírozásával. Az itt szereplő vélemények és állítások a szerző(k) álláspontját tükrözik, és nem feltétlenül egyeznek meg az Európai Unió vagy az Európai Oktatási és Kulturális Végrehajtó Ügynökség (EACEA) hivatalos álláspontjával. Sem az Európai Unió, sem az EACEA nem vonható felelősségre miattuk.
IT	Finanziato dall'Unione europea. Le opinioni espresse appartengono, tuttavia, al solo o ai soli autori e non riflettono necessariamente le opinioni dell'Unione europea o dell'Agenzia esecutiva europea per l'istruzione e la cultura (EACEA). Né l'Unione europea né l'EACEA possono esserne ritenute responsabili.
LT	Finansuojama Europos Sąjungos lėšomis. Tačiau išreiškiamas požiūris ar nuomonė yra tik autoriaus (-ių) ir nebūtinai atspindi Europos Sąjungos ar Europos švietimo ir kultūros vykdomosios įstaigos (EACEA) požiūrį ar nuomonę. Nei Europos Sąjunga, nei EACEA negali būti laikoma už juos atsakinga.
LV	Eiropas Savienības finansēts. Paustie viedokļi un uzskati atspoguļo autora(-u) personīgos uzskatus un ne vienmēr sakrīt ar Eiropas Savienības vai Eiropas Izglītības un Kultūras izpildaģentūras (EACEA) viedokli. Ne Eiropas Savienība, ne EACEA nenes atbildību par paustajiem uzskatiem.
MT	Iffinanzjat mill-Unjoni Ewropea. Madankollu, il-fehmiet u l-opinjonijiet espressi huma dawk tal-awtur(i) biss u mhux neċessarjament jirriflettu dawk tal-Unjoni Ewropea jew tal-Aġenzija Eżekuttiva Ewropea għall-Edukazzjoni u għall-Kultura (EACEA). La l-Unjoni Ewropea u lanqas l-EACEA ma jistgħu jinżammu responsabbli għalihom.
NL	Gefinancierd door de Europese Unie. De hier geuite ideeën en meningen komen echter uitsluitend voor rekening van de auteur(s) en geven niet noodzakelijkerwijs die van de Europese Unie of het Europese Uitvoerende

	Agentschap onderwijs en cultuur (EACEA) weer. Noch de Europese Unie, noch het EACEA kan ervoor aansprakelijk worden gesteld.
PL	Sfinansowane ze środków UE. Wyrażone poglądy i opinie są jedynie opiniami autora lub autorów i niekoniecznie odzwierciedlają poglądy i opinie Unii Europejskiej lub Europejskiej Agencji Wykonawczej ds. Edukacji i Kultury (EACEA). Unia Europejska ani EACEA nie ponoszą za nie odpowiedzialności.
PT	Financiado pela União Europeia. Os pontos de vista e as opiniões expressas são as do(s) autor(es) e não refletem necessariamente a posição da União Europeia ou da Agência de Execução Europeia da Educação e da Cultura (EACEA). Nem a União Europeia nem a EACEA podem ser tidos como responsáveis por essas opiniões.
RO	Finanțat de Uniunea Europeană. Punctele de vedere și opiniile exprimate aparțin, însă, exclusiv autorului (autorilor) și nu reflectă neapărat punctele de vedere și opiniile Uniunii Europene sau ale Agenției Executive Europene pentru Educație și Cultură (EACEA). Nici Uniunea Europeană și nici EACEA nu pot fi considerate răspunzătoare pentru acestea.
SK	Financované Európskou úniou. Vyjadrené názory a postoje sú názormi a vyhláseniami autora(-ov) a nemusia nevyhnutne odrážať názory a stanoviská Európskej únie alebo Európskej výkonnej agentúry pre vzdelávanie a kultúru (EACEA). Európska únia ani EACEA za ne nepreberajú žiadnu zodpovednosť.
SL	Financirano s strani Evropske unije. Izražena stališča in mnenja so zgolj stališča in mnenja avtorja(-ev) in ni nujno, da odražajo stališča in mnenja Evropske unije ali Evropske izvajalske agencije za izobraževanje in kulturo (EACEA). Zanje ne moreta biti odgovorna niti Evropska unija niti EACEA.
SV	Finansieras av Europeiska unionen. De synpunkter och åsikter som uttrycks är endast upphovsmannens [upphovsmännens] och utgör inte Europeiska unionens eller Europeiska genomförandeorganet för utbildning och kulturs (EACEA) officiella ståndpunkt. Varken Europeiska unionen eller EACEA tar något ansvar för dessa.