DT Challenge Year 7 Python + Biology – Classification

https://groklearning.com/course/aca-dt-7-py-biology/

About this activity

The process of classifying species in Biology relies on identifying key features that help determine their relative position in a taxonomy. These questions are often phrased in terms of whether a feature is present or not present - a dichotomous (or binary) comparison.

One of the critical decision-making constructs in computer programming is that of branching - often implemented in programming languages as an if statement. If statements determine whether a condition being tested is either True or False, and will execute different statements on that basis.

In this challenge, students build a simple biological species classifier based on physical features and characteristics of different species. This requires an understanding of direct text input/output (I/O), branching, and in the extension part of the challenge, iteration, file I/O and collection data structures called lists (or arrays).

Age

This challenge targets students in year 7, though it can also be used as an introductory course for students in later years who have not yet been exposed to basic programming concepts. There is also a version of this challenge for year 5 students that uses Blockly (Google Doc | PDF).

Language

Python — a general-purpose programming language that is widely used and easy to learn.

Time

The course is designed to be completed in approximately 15 hours of class time, with another 15 hours included in the extension challenge.

Key Concepts

Key Concept	Coverage
Abstraction	Focusing on specific details of species allows for the classification
Data: collection, representation, interpretation	Species can be represented in individual parts (through strings in variables) or; in collections (lists) stored in files (extension). A taxonomy is a representation of relationships between species.
Specification, algorithms, implementation	Simple Algorithms, user input, branching, looping over files (extension)
Interaction	Users interact with programs via interfaces that can be primarily text-based (command line input/output)

Objectives (Content Descriptions)

Digital Technologies

ACTDIK024	Investigate how digital systems represent text, image and audio data in binary
ACTDIP026	Analyse and visualise data using a range of software to create information, and use structured data to model objects or events
ACTDIP029	Design algorithms represented diagrammatically and in English, and trace algorithms to predict output for a given input and to identify errors
ACTDIP030	Implement and modify programs with user interfaces involving branching, iteration and functions in a general-purpose programming language
ACTDIP031	Evaluate how student solutions and existing information systems meet needs, are innovative, and take account of future risks and sustainability

Science

ACSSU111	Classification helps organise the diverse group of organisms
ACSIS129	Construct and use a range of representations, including graphs, keys and models to represent and analyse patterns or relationships in data using digital technologies as appropriate

What are we learning? (Abstract)

At the conclusion of these activities students will be able to:

- Understand the taxonomy used to classify species, and why that taxonomy is useful
- Represent a dichotomous key for classifying species in multiple ways
- Identify key features of various biological species
- Write programs using the Python programming language
- Recognise that breaking down a problem into smaller steps (decomposition) makes it easier to solve problems
- Debug algorithms
- Recognise that steps in algorithms need to accurate and precise
- Recognise that problems can have multiple solutions
- Utilise branching (if, if-else, and if-elif-else) in programs
- Utilise user input
- Define the term *algorithm*
- Define the term *decomposing*
- Define the term *branching*

Extension:

- Access structured data stored in lists
- Manipulating structured data to produce new data
- Access and process data stored in files
- Utilise for loops in programs
- Define the term *iteration*

Module outline

The course consists of five core modules:

1. Output
   This module introduces printing to the screen with Python and the importance of classifying data. It introduces the concept of variables.
2. Input
   This module goes into more detail about variables, accepting input from the user and using that input in different ways for feedback. In introduces taxonomic rank and how it divides the animal kingdom, as well as how scientific names for species are determined.
3. Decisions
   This module introduces the term algorithm, decisions, and allows students to write programs that change their behaviour on the basis of information from the user. It explains how dichotomous keys allow for classification of species within the taxonomy.
4. Complex decisions
   This module goes into further detail about decisions using multiple values, and the importance of sequencing in the decision-making process.
5. Mini project: Simple classifier
   This module has students develop an algorithm and program that classifies four different animals on the basis of their physical features. It consolidates the learning from the previous modules.

There is also an extension course consisting of four extension modules:

1. Lists
   This module introduces structured collections of data (Python lists) and how they allow us to simplify and generalise our programs.
2. Working with lists
   This module demonstrates how lists can be created and manipulated to process and present data input by a user.
3. Looping
   This module demonstrates the *for loop*, to loop over each item in a list and process each one. It explains how we can process collections of information that is stored in data structures, and process that information regardless of how much data is stored.
4. Reading from files
   This module explains how files allow us to store data that can be accessed at a later date in our programs and how we use that data. It explains how scientists rely on files and databases to analyse data and make conclusions.
5. Project: Animal classifier
   A more advanced version of the classifier at the end of the core modules. Uses files, lists and a larger collection of animals to present more varied data about each species. It demonstrates how a larger collection could be processed using programming structures.

Types of component:

 **Discussion**

 **Worksheet**

 **Plugged Activity**

 **Group Activity**  **Unplugged Activity**  **Video**

 **Animation**  **Reflection**  **Game**  **App**

Challenge Introduction — how does programming help biologists?

Programming a computer means giving it instructions - a sequence of steps - to follow in a way that it “understands”. Another word for the sequence of instructions given to it is an algorithm.

Biologists, like all scientists, use data they collect from experiments and observations to test ideas and learn new things about the world. As more data becomes available, analysing that data so that you can draw conclusions about it becomes more difficult to do manually.

Being able to write computer programs to access, manipulate and process data allows you to test your ideas against larger amounts of data. This allows for simulations to be built, new questions and hypotheses to be developed, and for new solutions and products to be designed.

Many scientists now spend more time analysing their data than they do conducting their experiments. To do this efficiently, they need to work out how to represent their data so they can write code to analyse it. Some good examples from biology include how geneticists are able to analyse DNA by representing it as text strings, and being able to determine how diseases can spread throughout a population so they can design a suitable strategy for preventing further outbreaks.

The examples we explore in this challenge are very introductory, but these same principles are used in more complicated programs and on larger sets of data to make conclusions about scientific experiments.

New Vocabulary

From Digital Technologies:

Algorithm: A set of rules or step by step instructions to solve a problem or achieve an objective. A recipe is an example of an algorithm - it sets out what you need and the steps you follow to combine everything to create your food item(s).

Decomposition: breaking down a problem into smaller parts which can then be dealt with individually. This allows very complicated problems to be solved by first solving their individual parts separately and then working out how those individual solutions can be used together.

Branching: Changing the instructions executed by the program based on a certain condition. This allows you to specify that your program should behave one way in some cases, but a different way in others. In python, this is achieved through the use of `if-elif-else`.

Iteration: Repeating a set of instructions a number of times as part of the program. This allows your solution to scale to larger amounts of data and is important for generalising your solution to work for lots of similar problems. In this challenge, a `for` loop is used for iteration.

CSV: A comma separated value file, CSV files are very commonly used by scientists to store data for processing.

From Science:

Taxonomy: The classification of living things, usually based on their physical properties and shared characteristics. The taxonomy used in biology is hierarchical from *Domain* (highest) to *Species* (lowest), and the *scientific name* of an organism is a combination of its Genus and Species.

Dichotomous key: A fixed arrangement of choices where each choice is binary (consists of two choices). A dichotomous key can be used to classify organisms by checking for whether a feature or characteristic is present or absent.

Activity 1 - Understanding variables

Preparation and timing

No prior knowledge is required for this activity. It can be done at the beginning of a lesson or as an interlude when students hit the point in the challenge where they struggle with the variable concept.

This activity would be best delivered during the latter part of module 1 (output) or concurrently with module 2 (input).

Overview

- What are variables?
- Why are variables important in computer programs?
- How do we use variables in Python/Blockly?

Suggested Implementation

Variables are a critically important concept for students to understand - they are fundamental to all programs because they allow data to be stored and then accessed and manipulated at a later stage of the program. Without variables, our programs would not be able to do things like store settings, saved games or the contact information of your friends.

Video
What are variables?

Watch the following video on YouTube, published by code.org: https://youtu.be/G41G_PEWFjE

The video demonstrates the concept of variables - what they are, how they are used in programming and why they are important. It uses the analogy of storing things in a container and, how that allows us to retrieve things later on when we want to access that information again.

There is no need to watch all of the variable - the key bit of understanding students need to get from the video is that when you need to store something in a computer program for retrieval later, you need to use a variable.

Discussion:
Did you know that every instruction in a computer happens in an order? This means that when we get information from somewhere we need to store it if we want to access it later in our programs. Variables let us do this, and all programming languages have some way of representing this idea.

How do you remember things? If you want to retrieve something that you own later on, how do you make sure you remember where it was? Have you ever lost your phone or something else you value? What kinds of things do you do to make sure you don't lose it again?

Usually answers to these questions will go along the lines of "I always put X in place Y" or "I have a special thing (like a pencil case) which I use to store all of my things (pencils). This way, if I need a pencil, I just go back to that". This is what you're doing with a computer - you're identifying a place in the computer's memory where something has been stored, and then using that same label to retrieve it later, or to update what is being stored to something else.

 **Plugged Activity**

Challenge modules 1 & 2: variable questions

The challenges have been designed such that there are two programming exercises for each of the concepts introduced. The questions are very similar - often the only difference will be the text being printed an/or used in the input prompts.

In module 2 (and the latter part of module 1), you may find it useful to walk through the first question in each set as a group, demonstrating the use of variables for students and explaining the thinking process. You can then have students complete the second question in the set on their own or with a peer, giving them a chance to apply the knowledge and what they've seen demonstrated to solve a problem on their own.

Discussion questions:

 **Reflection**  **Group Activity**

Variables are encountered everywhere

Students share some examples where they think examples similar to the code they've been writing are used (particularly variables). Things might include:

- Have you had your name appear in a high score list in a game?
- Substituting the player's chosen name throughout a game's interface
- Bus destinations appearing in the bus display
- Usernames appearing on websites

But in all of these cases, the same data isn't always displayed - things can change depending on the situation. We'll learn about that later - *this can be the teaser for future lessons*.

Activity 2 - Branching: Simon says

Preparation and timing

For very young students you may need to explain and/or demonstrate the rules of *Simon Says*. Most students will have played this game before.

This activity could be delivered prior to or concurrently with module 3.

Overview

- What is branching?
- How does it relate to programming?
- How do we determine the conditions that we need to check to perform different instructions?

Suggested Implementation

Unplugged Activity **Group Activity**

Simon says...

Ask the student if they've all played simon says before and, if any of them haven't, explain the rules of the game. You should then play a game with the group. Olders students may not make many mistakes, and you may choose to end the game early if they aren't being knocked out very quickly.

You should then say that you'll play another game, but this time make the rule a little more difficult. Instead of the rule being *does the instruction start with "Simon Says"*, try something like *does the instruction includes the word "you"* or *does the instruction has exactly five words in it*. You are demonstrating that the condition can change, but regardless what determines if the instruction should be performed is some condition that you've set as your rule.

Branching in programming is what allows us to define different behaviour in the same program. It allows us to check some value or condition in our program, and respond differently based on what the result is. Without branching, all of our programs would perform exactly the same thing every time, and we would need to write brand new programs every time the tiniest little thing changed. Our programs would not be able to respond differently to new users or data.

Both Blockly and Python provide ways of performing branching, and these are covered in modules 3 and 4 of the challenges.

Plugged Activity

Challenge modules 3 & 4: branching/decision questions

The challenges have been designed such that there are two programming exercises for each of the concepts introduced. The questions are very similar - often the only difference will be the text being printed an/or used in the input prompts.

In modules 3 and 4, you may find it useful to walk through the first question in each set as a group, demonstrating the use of branching for students and explaining the thinking process. You can then have students complete the second question in the set on their own or with a peer, giving them a chance to apply the knowledge and what they've seen demonstrated to solve a problem on their own.

Discussion questions:

 **Reflection**  **Group Activity**

Variables are encountered everywhere

Students share some examples where they think branching occurs in the programs they use regularly. Things might include:

- If the phone rings for 30 seconds and isn't answered, it automatically goes to voicemail
- If you choose a certain character in a game, then it loads that character so that you can play as that person
- If you try to use an ability in a game and it isn't charged, it doesn't let you do it
- If the user types in different answers to a question in a program, different options are presented to them for their next action

All of these actions require the program to check some value or condition. The way the computer responds is determined either by the user themselves, or due to the state of the program at the time the check is performed.

Activity 3 - Unplugged: Classifying organisms

Preparation and timing

Before starting this activity, consider printing out the relevant slide with pictures of animals and give a coy to each student. You may also want to make large versions to work through the activity on the board (either electronically on an IWB or by using blu-tack on a whiteboard).

This activity could be delivered any time prior to the mini-project module (module 5).

Overview

- How does classification work?
- How do we use the features of animals to distinguish between them?
- What is a dichotomous key? How do these related to decision trees?
- How do we minimise the number of questions we need to ask to identify an animal in our set of species?

Suggested Implementation

Unplugged Activity

Classifying organisms with computer science

The content and structure of this lesson is laid out in this Google Slides presentation:

Classifying living things with computer science (downloads)
Google Slides | MS Powerpoint | PDF

The slide deck includes instructions and guidance of how to use it. You can tell the audience for each slide based on its layout:

- Slides that are split are teacher information slides that provide suggestions and additional information about each part of the activity
- Green slides are title slides for sections designed for students. These can be used as visual cues and prompts for students, and can be displayed on a projector screen
- Orange slides are indicators for where the answers to problems can be found. These can be displayed to students after they have completed the activity.

Working through this activity prepare students for the mini-project at the end of the challenge (module 5). The process they follow here is the same one they will go through during that module but without the programming elements.

Discussion questions:

- How do we decide which features to use at each level of our decision-tree?
- Why is it best to use questions that have yes/no answers?
- How is this similar to if statements in Python?

Overarching Activity - The DT + Biology challenge

Preparation and timing

The challenge consists of 5 modules, with an additional 4 extension modules for students who have had some experience with programming before or who move through the material quickly.

Completing all of the core modules should take around 10-15 hours, with the extension modules offering an additional 15 hours.

Overview

- The challenge assumes no programming knowledge
- It teachers programming concepts, biology concepts and python syntax concurrently
- The challenge is broken into two parts - a core and extension - allowing students with no experience to get started but providing some challenge for experienced or very capable students.

Suggested Implementation

Plugged Activity

Challenge modules

To get the most out of the modules, students should always:

- Students should read interactive notes, including running any example code provided
- Attempt all problems and review questions

Clicking the **Run** button allows students to check their code by running it and observing the output. When they believe they have the solution correct, pressing the **Mark** button will check to see if the code passes the test cases and will provide feedback if it does not.

You can interleave the unplugged activities above with completion of the online modules, especially if you find students are struggling with a concept explored in one of the activities.

Almost every slide and problem includes “Teacher Notes” that verified teachers can access. These provide additional information, suggestions and activities for teaching each concept and exploring ideas further with students in class.