**YEAR EIGHT CORNERSTONE**

**Logic and the Machine**

**Term Four**

**Weeks 2 - 5**

****

*Part of the schematic for an early difference engine – essentially a purely mechanical computer*

**TOPIC: Logic and the Machine**

**Lesson 1 - 2: Python, Shells and Turtles**

[](https://xkcd.com/353/)

**Not your average Cornerstone unit…**

This unit runs a little differently – you will be exploring the ideas of Logic in two ways: *discussing in class* (ok, normal Cornerstone so far) and also *writing computer code*.

At the end of the topic you will submit your favourite turtle drawing that you have made (more about this later).

**Prior views: Write in your Cornerstone journal**

1. Have you ever written any computer code before? Do you think it has anything to do with ‘logic’? Truth?
2. What makes something ‘logical’?

**Installing Python**

For this unit you will need to install the programming environment Python. Go to <https://www.python.org/downloads/> to install Python (choose version 3.x). If for some reason you are unable to install you can use Python at <http://www.skulpt.org>

**How the activities work**

Anything in a

|  |
| --- |
| grey filled box |

is code that you should try in the Python shell. Anything that is in an

|  |
| --- |
| orange double bordered box |

is an optional challenge that you may try.

**Activity: Operation decoding operations**

* Learning Intention: To explore what computer code looks like
* Success Criteria: I have experimented with some computer code and can understand some key computer terms (shell, input, output, operator)

*What is this Python anyway?*

Python is a **programming language**. It translates words into computer actions.

Once you have Python installed you will need to open a program called *IDLE*. When it opens you will see something like:

****

This is the Python **Shell** – and is where what you type gets turned into action. It is like a window that lets you see into what the computer is doing with your code. Let’s check to see if our computer knows it’s mathematics. Type in:

|  |
| --- |
| 4 + 9 |

And press enter. It should look like:



Here the computer has responded to the information you gave it (4 + 9) with a response (13). We call the information you give a computer the **input**, and in the Python Shell inputs will always appear after the >>>. We call the response from the computer the **output** and this appears in the Python Shell on the line(s) underneath the input.

It looks like the symbol “+” works like addition. Symbols that are used to combine two things are called **operators**. See if you can uncover what the following operators do by testing these inputs in the Python Shell:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operator | Input 1 | Output 1 | Input 2 | Output 2 | Description |
| + | 5 + 3 | 8 | + |  | Addition |
| - | 5 – 3 |  | - |  |  |
| \* | 5\*3 |  | \* |  |  |
| / | 5/3 |  | / |  |  |
| < | 5 < 3 |  | < |  |  |
| > | 5 > 3 |  | > |  |  |
| == | 5 == 3 |  | == |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Challenge: Experiment and discover what these operators do.   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Operation | Input 1 | Output 1 | Input 2 | Output 2 | Description | | \*\* | 5\*\*3 |  | \*\* |  |  | | // | 5//3 |  | // |  |  | | % | 5%3 |  | % |  |  | | <= | 5 <= 3 |  | <= |  |  | | == | 5 == 3 |  | == |  |  | | != | 5 != 3 |  | != |  |  |   . |

**Activity: A Variable Turtle**

* Learning Intention: To build my coding skills
* Success Criteria: I have used variables, loops and turtle to draw pictures in Python

*Part 1: Turtle*

We are going to experiment with some new inputs to create outputs in the form of pictures. To get the computer to translate our inputs into pictures we need to use a thing called turtle. We need to import the turtle into the Python Shell by typing:

|  |
| --- |
| from turtle import \* |

This will look like it does nothing, but it has actually told the computer to get it’s pen ready. We can now tell the computer to draw. To tell the computer how to draw type this into the Python Shell:

|  |
| --- |
| forward(100) |

and press enter. A new window will have opened with your turtle in it (your turtle actually looks more like a triangle). It should have moved forward 100 steps (it has very tiny steps). Next try:

|  |
| --- |
| left(45) |

and press enter. Your turtle should have turned to the left 45 degrees. Next try:

|  |
| --- |
| forward(50) |

and press enter. Your turtle should have moved forward another 50 (tiny) steps.

Now close the window with the turtle in it. See if you can draw a square, a triangle and a cross. Every time you want to start a new picture make sure you close your old picture window.

|  |
| --- |
| *Challenge*: Can you write your name using turtle? You may want to use some of the commands listed here - <https://docs.python.org/3/library/turtle.html#turtle-motion> |

*Part 2: Variables*

Go to your Python shell and type in the following (and press enter each time – the ‘enter’ key has that name because you are *entering data* to the computer):

|  |
| --- |
| n = 4 |

You will notice nothing appears to happen when you press enter. But then try:

|  |
| --- |
| n |

Your computer remembered! We can then do things with this n as well. Try:

|  |
| --- |
| 3 + n  n\*n |

What do you notice? We have transformed this symbol n into a thing called a **variable**.

A variable is a place for the computer to store some information.

They are called variables because they can change values. Try:

|  |
| --- |
| n = 3.1415 |

and then try:

|  |
| --- |
| n |

The computer has changed the value of what n is.

*Part 3: Loops*

Now for a **loop** – possibly the most important thing a computer can do. A loop repeats the same instructions again and again (ok, that sounds boring) but with variables those instructions can change a little bit each time (now that might actually be useful). Loops are very fussy – they need to be told lots of things in order to run. They need to know:

1. That they are a loop. We use the word for in Python for this
2. What the name of the variable is. The example below uses n. You could use any word you like.
3. What values the variable will take. After the variable we write the special word in and then a *list of values*. A list in Python uses square brackets with the elements of the list separated by commas, like [1, 2, "cat", 3]
4. When to start repeating. This is fussy in Python, you need to tell it to “do” something, and in Python the symbol for this is a colon :
5. What to repeat. These are written below and they must be *indented* (tabbed or spaced in, evenly!). When you stop repeating the code is no longer indented (see the example below)

Next try the following (copy it *exactly –* and you will need to press enter *twice* this time)

|  |
| --- |
| for n in [1, 4, 5, 10]:  print(n)  print("This bit repeats")  print("This bit does not repeat") |

But it is really tiresome to have to write all those numbers in a big long list, so instead we can use a special function called range. This works by giving Python a list start and stop points, so range(4,12) starts at 4 and stops *before* it gets to 12. Try the following to see it in action:

|  |
| --- |
| for n in range(4,12):  print(n) |

It should print out the numbers 4, 5, 6, 7, 8, 9, 10, 11.

|  |
| --- |
| *Challenge*: Can you change the code to print out the numbers 3, 5, 7, 9, 11? |

*Part 4: Strange turtle loops*

Now we use loops to draw pretty pictures! These can get long and complicated, so when we are writing lots of instructions we leave the shell and write a **program**. A program is just a list of instructions that are stored as text that the computer quickly sends to the Python Shell for you.

To create a text file for the program in IDLE go to *File -> New File* (or command-n or control-n) and it should open a new window for you to type your code into. Make sure you save your program somewhere sensible first (create a new folder called “Code” in your documents).

When you want to run the code press F5 or choose *Run -> Run module.*

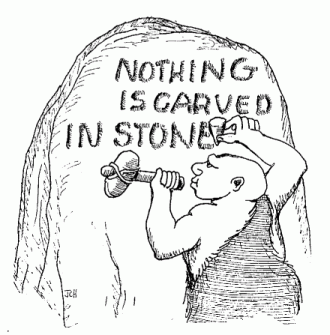
Here is an example program for you to try. Enter it into a new file and run it. Then change something and run it again!

|  |
| --- |
| from turtle import \*  for n in range(1,40):  forward(n)  left(2)  backward(n) |

|  |
| --- |
| *Challenge*: Experiment! Make the craziest picture you can. |

**TOPIC: Logic and the Machine**

**Lesson 3 - 4: Propositions and Conditionals**



Logic is the art of going wrong with confidence.

*Joseph Wood Krutch*

**Discuss**

1. A proposition is a statement that is either True or False. Which of the following are propositions?
2. 2 + 3 = 5
3. Dogs have six legs
4. Asparagus
5. Mr Parker has a pet unicorn
6. Where is my difference engine!?
7. All of your bases belong to us
8. Look at the **operations** that you summarised last section (such as +, < and ==) : which of them are **propositions**?
9. A **conditional** is something of the form *if this, then that*. An example would be “If I say it then I mean it” or “I mean what I say”.

**Activity: Alice’s conditionals**

* Learning Intention: To understand what a conditional is
* Success Criteria: I can identify a conditional statement, and understand if it’s converse is true

**Read** (or play out) the following (from *Alice in Wonderland*):

MARCH HARE: You should say what you mean, I do.

ALICE: I say what I mean at least — at least I mean what I say — that's the same thing, you know.

HATTER: Not the same thing a bit! You might just as well say that "I see what I eat" is the same thing as "I eat what I see"!

MARCH HARE: You might just as well say, that "I like what I get" is the same thing as "I get what I like"!

DORMOUSE: You might just as well say, that "I breathe when I sleep" is the same thing as "I sleep when I breathe"!

Each of the conditional statements can be broken down into a conditional in the form of

**if** *this,* **then** *that*

For example, “I breathe when I sleep” can be written as: **if** *I am sleeping,* **then** *I breathe*.

**Discuss**

1. In groups, come up with five more conditional statements that make no sense when reversed. The reverse of a conditional statement is called it’s **converse**.
2. With your conditional statements try to write them in the form: **if** *this* **then** *that*
3. Extension: If **if** *this,* **then** *that* is not the same as **if** *that,* **then** *this*, then what is it the same as? [Hint: you may need the word **not** in there)

In Python, **if** is used for **if** and : is used for **then**

**Activity: Outsourcing your decision making**

* Learning Intention: To examine what a conditional is
* Success Criteria: I can represent conditionals with flow charts and code a conditional statement using if and then (:)



All programming languages have special variables for Yes/True and No/False which help make decisions. In Python they are cleverly named True and False.

What is more useful is the use of a **conditional statement**. This involves an **if** and a **then**. For instance:

**if** *it is raining* **then** *there is no sport training*

But how to write a program that does this?

We need to be able to tell the computer if it is raining, and help it remember by assigning it a **variable** (remember that a variable is a place for the computer to store information). Enter the code below into the Python Shell and press enter. What it is doing is creating a variable called isRaining and assigning it a value using the input function. The input function waits for the user (you) to type something into the shell after it asks you a question (in this case, "Is it raining?")

|  |
| --- |
| isRaining = input("Is it raining? ") |

The computer should then ask you “Is it raining?” and it will wait for a response. You need to type Yes or No (remember the capital letters) and then *press enter*.

Then check to see if the computer has remembered the value of isRaining by typing

|  |
| --- |
| isRaining |

and pressing enter.

Now try checking the value of isRaining with the string “Yes”

|  |
| --- |
| isRaining == “Yes” |

The output should be either True or False. We can use this type of output to make decisions.

Now we will combine the input function with a conditional (if *this* then *that*) to create a short program. Create a new file and enter the following code, then run it.

|  |
| --- |
| isRaining = input("Is it raining? ")  if isRaining == “Yes”:  print("There is no sport training.") |

Note that the if represents if, and the : represents then. So the code would read (in human-speak):

if it isRaining then print "There is no sport training."

We can make this even useful if we add an else statement. The else statement is like the “No” branch on our flow chart. It is where we go if it is not raining.

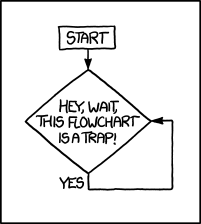
|  |
| --- |
| isRaining = input("Is it raining? ")  if isRaining == “Yes”:  print("There is no sport training.")  else:  print("Sport training is still on.") |

In human-speak this code would read:

if it isRaining then print "There is no sport training.", otherwise print "Sport training is still on."

Below, or In your Cornerstone journal **draw** a flow chart of any decision you might make.

|  |
| --- |
| *Challenge*: Can you write a program that does what your flow chart does? |



**Activity: I can turtle a rainbow**

* Learning Intention: To understand what a conditional is
* Success Criteria: I have played with conditionals by drawing pictures in Python with alternating colours

Now we will link together **propositions** and **conditional** **statements** to make turtle draw in pretty colours. Try the following and experiment to see what you can change.

|  |
| --- |
| from turtle import \*  for n in range(1,180):  if n%2 == 0:  color("red")  else:  color("blue")  forward(100)  backward(100)  left(1) |

And a more complicated version…

|  |
| --- |
| from turtle import \*  # this makes the turtle go fast  hideturtle()  speed(0)  for n in range(1,180):  if n%2 == 0:  color("#aa6622")  if n%3 == 0:  color("#005534")  if n%4 == 0:  color("#990066")  if n%5 == 0:  color("#ab3211")  forward(100)  backward(100)  left(1) |

|  |
| --- |
| *Challenge*: How do the statements n%4 == 0 work?  Why are they helpful for writing a program that repeats a pattern?  You may want to experiment with the n in the proposition n%4 == 0 to find when it is True. |

**TOPIC: Logic and the Machine**

**Lesson 5: Combining truth**

**Warm up riddle:**

*A man is asked what his daughters look like. He answers, "They are all blondes, but two, all brunettes, but two, and all redheads, but two." How many daughters did he have?*

**Activity: A robot that listens**

* Learning Intention: To explore Boolean algebra (mathematics with True and False statements)
* Success Criteria: I have played with a program that uses an and statement

Download the program ***a robot that listens.py*** from the Canvas module. If that does not work then make a new file by going to *File -> New File* and copy the following in there:

|  |
| --- |
| whoilike = "David"  colorilike = "Blue"  you = input("What is your name? ")  if you == whoilike:  print("I like people called " + whoilike + "!")  else:  print("I guess you could call yourself " + you + " if you had to. ")  favcolor = input("What is your favourite colour? ")  if favcolor == colorilike:  print("I like " + colorilike + " too! ")  else:  print("Ewww.")  if you == whoilike and favcolor == colorilike:  print("Let’s be friends!") |

**Run the file** – see if you can make friends with the Robot (you can always cheat by *editing* the program!)

This new program introduced a new operator – the and operator. Together with the or operator they are the two most important ways of combining truth in logic.

**Discuss:**

1. What were the conditions required for the robot to be your friend?

**Activity: and and or**

* Learning Intention: To explore Boolean algebra (mathematics with True and False statements)
* Success Criteria: I have completed an and and or table, and used Boolean operators (and and or) in a Google search

We can combine truth values (True’s and False’s) by using the operators and and or. But how do they work? Test each of the following to see what they output:

|  |
| --- |
| True and False  True or False  4 > 2 or 4 < 2  1 == 1 and 1 == 1000  1 == 1 or 1 == 1000 |

**Discuss**

1. What do you think the and and the or do?
2. If *it is raining* and *you are wearing green socks*, then is it true that *it is raining and you are wearing blue socks*? Is it true that it is *not raining or you are wearing green socks*?
3. If *you have brown hair*, is it true that *you have brown hair or you have blonde hair*? Is it true that *you have brown hair or 1 is a number*?

**Complete** the following tables (they are like a multiplication table, but with Truth values – the first has been partially completed for you), using Python to help if you need.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | and | True | False | | True | True |  | | False | False |  | | |  |  |  | | --- | --- | --- | | or | True | False | | True |  |  | | False |  |  | |

Think of two words that are not usually related, such as spaghetti and elephants. Go to [www.google.com](http://www.google.com) and **search** for (you replace the spaghetti and elephants with your two words) the following

“spaghetti” AND “elephants”

“spaghetti” OR “elephants”

“spaghetti” AND –“elephants”

**Discuss** what you notice about the results. What do you think the – does? Try experimenting with different words, or research what they do here:

<https://supple.com.au/tools/google-advanced-search-operators/>

|  |
| --- |
| *Supreme* *Challenge* (A riddle from Einstein!)  Five friends have their gardens next to one another, where they grow three kinds of crops: fruits (apple, pear, nut, cherry), vegetables (carrot, parsley, gourd, onion) and flowers (aster, rose, tulip, lily).   1. They grow 12 different varieties. 2. Everybody grows exactly 4 different varieties 3. Each variety is at least in one garden. 4. Only one variety is in 4 gardens. 5. Only in one garden are all 3 kinds of crops. 6. Only in one garden are all 4 varieties of one kind of crops. 7. Pears are only in the two border gardens. 8. Paul's garden is in the middle with no lily. 9. Aster grower doesn't grow vegetables. 10. Rose grower doesn't grow parsley. 11. Nuts grower has also gourd and parsley. 12. In the first garden are apples and cherries. 13. Only in two gardens are cherries. 14. Sam has onions and cherries. 15. Luke grows exactly two kinds of fruit. 16. Tulips are only in two gardens. 17. Apples are in a single garden. 18. Only in one garden next to the Zick's is parsley. 19. Sam's garden is not on the border. 20. Hank grows neither vegetables nor asters. 21. Paul has exactly three kinds of vegetable.   Who has which garden and what is grown where? (You can find the solution and more super challenging riddles here - <http://brainden.com/einsteins-riddles.htm> |

**TOPIC: Logic and the Machine**

**Lesson 6: Syllogisms**



In the play *Rosencrantz and Guildenstern are dead* the two main characters are talking in the opening scene whilst flipping a coin that is continually landing on heads. Watch the scene here - <https://youtu.be/NbInZ5oJ0bc?t=1m47s> whilst reading the dialogue below.

[riding down trail, tossing a coin that is continually coming up heads]

ROS: Bet? Heads I win. Again. Heads. Heads. Heads. Heads...

[camping in the forest, still tossing the coin]

GUIL: It must be indicative of something besides the redistribution of wealth.

ROS: Heads.  
GUIL: A weaker man might be forced to re-examine his faith, if for nothing else at least in the laws of probability.

ROS: Heads.  
GUIL: Consider; **one: probability is a factor which operates within natural forces. Two, probability is not operating as a factor. Three, we are now held within un-, sub- or supernatural forces.** Discuss.

ROS: What?

Guildenstern has used an example of deductive reasoning called a *syllogism*. They always work in the same way, with two statements combining into a third truth.

**Activity: Syllogistic it like Socrates**

* Learning Intention: To explore what a syllogism is
* Success Criteria: I have read some syllogisms, and created my own

Syllogisms always follow the form:

|  |  |
| --- | --- |
| 1 | A proposition |
| 2 | Another proposition |
| 3  (the conclusion) | *Therefore*, some proposition that **must** be true if propositions 1 and 2 are true |

**Discuss** the following four examples, considering:

1. Is the conclusion (the third statement) true?
2. Does the conclusion follow from the first two statements?
3. For any false conclusions, why were they false? Was the problem with the initial statements or with the deduction?

|  |  |
| --- | --- |
| Syllogism 1   1. Some cats are kittens 2. All kittens are young 3. *Therefore,* some cats are young | Syllogism 2   1. All teachers are people 2. Some teachers are Mr Parker 3. *Therefore,* all teachers are Mr Parker |
| Syllogism 3   1. All numbers are trees 2. Some trees have leaves 3. *Therefore,* some numbers have leaves | Syllogism 4   1. All men are mortal 2. Socrates is a man 3. *Therefore*, Socrates is mortal |

Then **create** your own syllogism. Try creating one that is true (ie makes sense), and then try creating one that is a *valid deduction* (3 follows from 1 and 2) from *false statements* (1 or 2 are actually false), like:

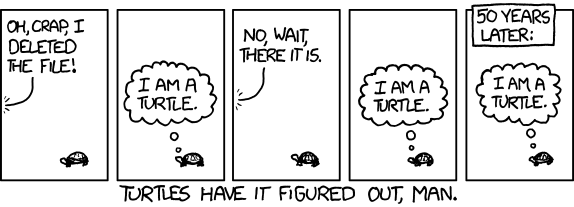
1. All men are potatoes
2. Some Jedi are men
3. *Therefore,* some Jedi are potatoes

|  |
| --- |
| *Challenge*: Play the Rosencrantz and Guildenstern game (download from Canvas)  ROS: I've never known anything like it!  GUIL: And syllogism: **One, he has never known anything like it. Two: he has never known anything to write home about. Three, it's nothing to write home about**... |

Extension: Examine the different types of syllogisms here - <https://www.fibonicci.com/logical-reasoning/syllogisms/examples-types/>

**TOPIC: Logic and the Machine**

**Lesson 7 - 8: Turtle Project**



Aim: To create the prettiest, most interesting picture you can using the turtle module in python.

You should try to include a for loop in your code somewhere, and for extension try to include a variable, change the color, use if statements (and even an else), or find some other interesting things to do from here – <https://docs.python.org/2/library/turtle.html#turtle-methods>

Happy coding!