

technocamps



UNDEB EWROPEAIDD
EUROPEAN UNION



Llywodraeth Cymru
Welsh Government

Cronfa Gymdeithasol Ewrop
European Social Fund



Prifysgol
Abertawe
Swansea
University



PRIFYSGOL
BANGOR
UNIVERSITY



Cardiff
Metropolitan
University

Prifysgol
Metropolitan
Caerdydd

it.wales



PRIFYSGOL
ABERYSTWYTH
UNIVERSITY

PRIFYSGOL
Glyndŵr
Wrecsam

Wrexham
glyndŵr
UNIVERSITY

University of
South Wales
Prifysgol
De Cymru

Coding across the Curriculum For Wales



Coding Across the CFW

Coding can be implemented across all the Areas of Learning and Experience, reinforcing learning in the classroom and improving digital literacy in the process.

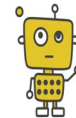
In today's world digital literacy is an essential skill for learners to develop. The technological requirements for jobs are ever increasing, and a strong start in digital skills will prepare learners and give them an advantage.



Expressive Arts



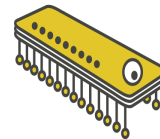
Health and Wellbeing



Humanities



Languages, Literacy and Communication



Mathematics and Numeracy



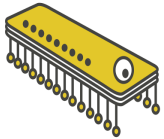
Science and Technology

Ideas for Coding Across the Curriculum



Health and Wellbeing

- Food Pyramid
- Football Pong



Mathematics and Numeracy

- Estimating Pi
- Compass/Clock



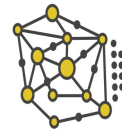
Science and Technology

- States of Matter
- Water Cycle



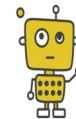
Languages, Literacy and Communication

- Translating Words
- Pronouns Quiz



Expressive Arts

- Learning Colours
- Matching Art Styles



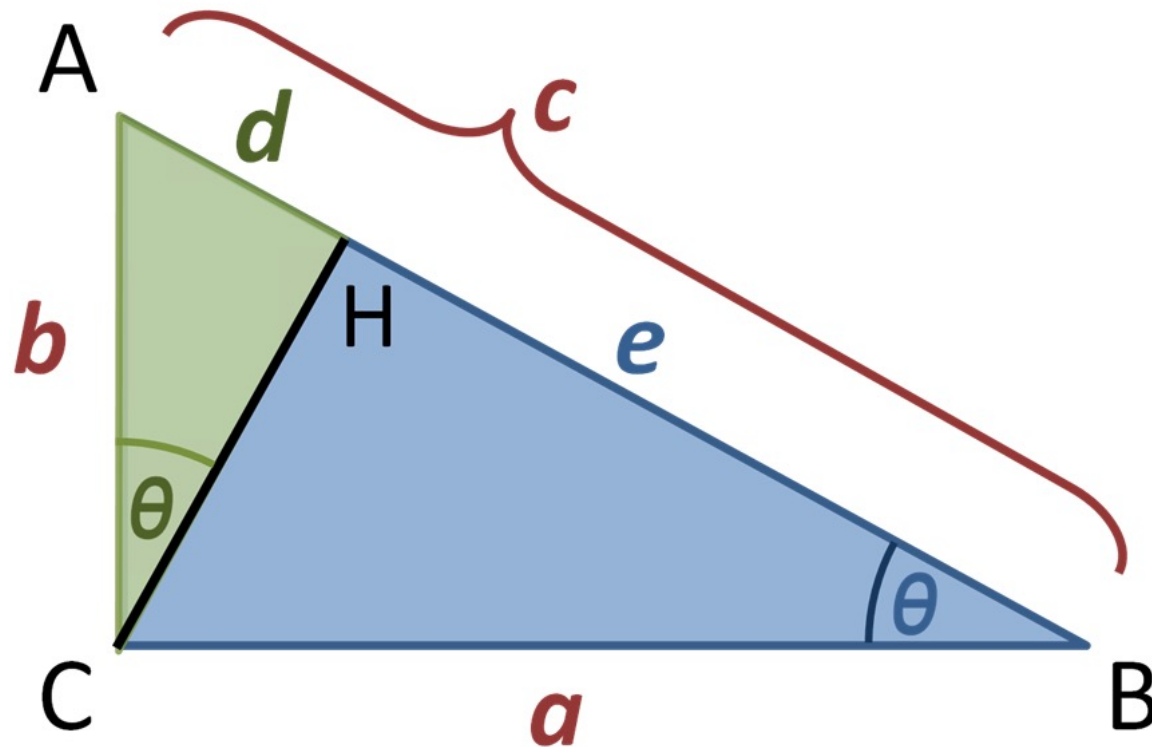
Humanities

- Animating a Timeline
- Migration Simulation

Pythagorean Theorem Calculator - Python



Pythagorean Theorem Calculator



Importing Libraries

To use the Pythagorean theorem to calculate the lengths and angles of a right triangle we will need to import some functions from the python math library.

```
from math import sqrt, asin, acos, atan
```

Initialising

The program then initialise by printing an explanation of its function, and asking the user to input which side it should calculate.

```
print('Pythagorean theorem calculator! Calculate your  
triangle sides and angles')
```

```
print('Assume the sides are a, o and h, where h is  
the hypotenuse')
```

```
formula = input('Which side (a, o, h) do you wish to  
calculate?')
```


Adjacent

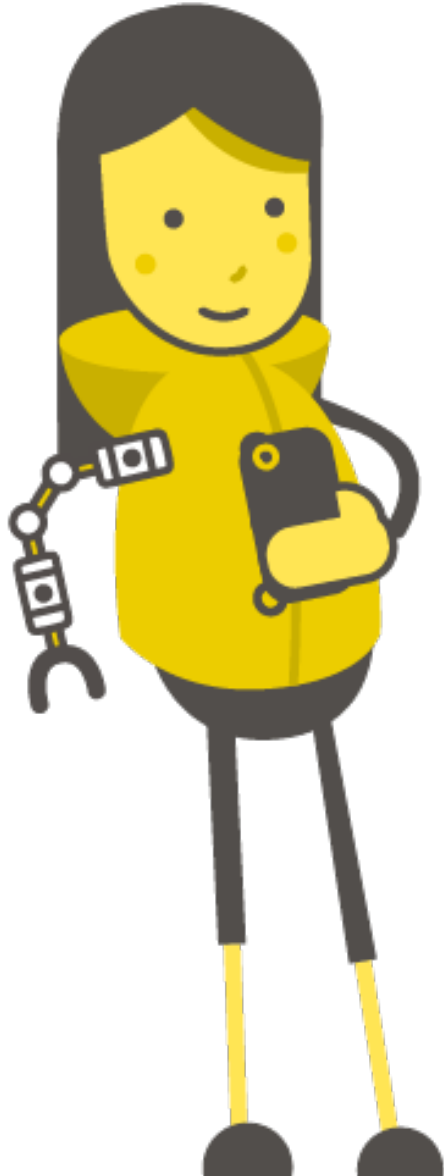
```
if formula == 'a':  
    sideO = int(input('Input the length of side o: '))  
    sideH = int(input('Input the length of side h: '))  
  
    sideA = sqrt((sideH * sideH) - (sideO * sideO))  
  
    theta = asin(sideO / sideH)  
  
    print('\nThe length of side a is : ' + str(sideA))  
    print('\nThe angle given by sin in radians is: ' +  
    str(theta))
```

Opposite

```
elif formula == 'o':  
    sideA = int(input('Input the length of side a: '))  
    sideH = int(input('Input the length of side h: '))  
  
    sideO = sqrt((sideH * sideH) - (sideA * sideA))  
  
    theta = acos(sideA / sideH)  
  
    print('\nThe length of side a is : ' + str(sideO))  
    print('\nThe angle given by sin in radians is: ' +  
    str(theta))
```

Hypotenuse

```
elif formula == 'h':  
    sideA = int(input('Input the length of side a: '))  
    sideO = int(input('Input the length of side o: '))  
    sideH = sqrt((sideA * sideA) + (sideO * sideO))  
  
    theta = atan(sideO / sideA)  
  
    print('\nThe length of side a is : ' + str(sideH))  
    print('\nThe angle given by sin in radians is: ' +  
str(theta))  
  
else:  
    print('Please select a side from a, o or h')
```



Nuclear Decay Simulation - Python

Nuclear Decay Simulation

If we have 100 coins we could simulate random nuclear decay by flipping them.

The probability of a coin landing on heads is practically 50%.

Each step we will eliminate all of the tails, as they are coins which have decayed.

Our results would look something like this:

Flip	1	2	3	4	5	6	7	8
Heads	100	51	24	13	5	2	1	0

Activity: Counting Heads

We could build a program that will simulate this coin flipping scenario for us.

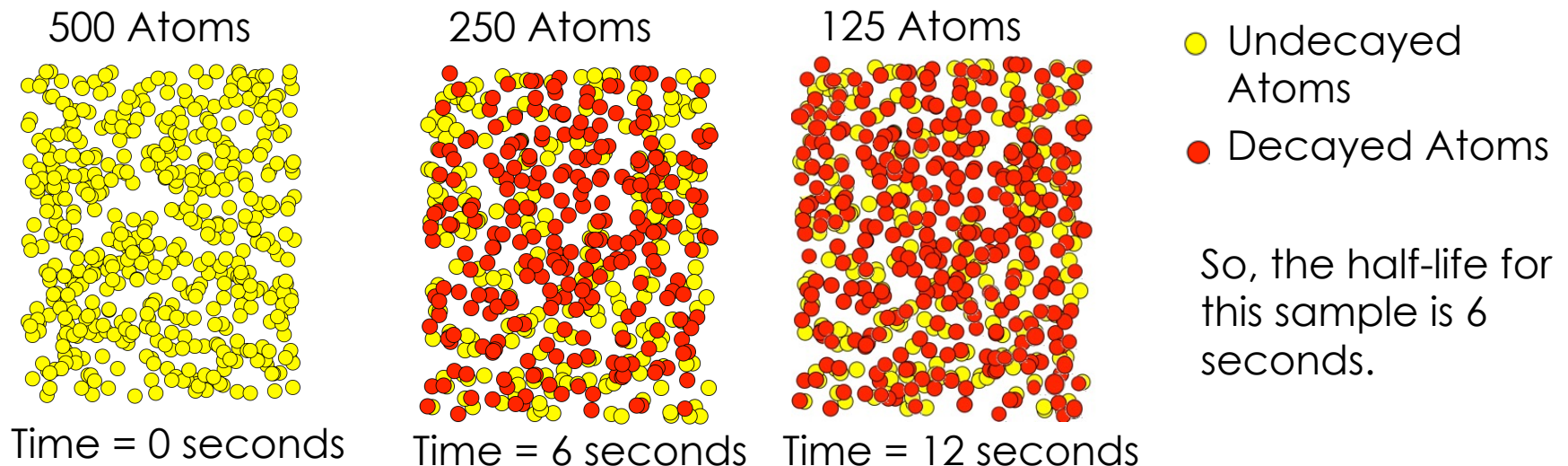
The program would need to do the following:

1. Ask the user to enter the initial number of coins
2. Set the current number of coins to the initial number of coins
3. Enter in the probability of getting Heads
4. While the **currentNumberOfCoins** is more than zero
 - a) For each coin in the range of 0 -> currentNumberOfCoins
 - b) headsOrTails = random integer between 0 and 1
 - c) If headsOrTails == 1 then take away a coin
 - d) Print out the number of Throws + currentNumberOfCoins

Half-Life

Half-life is the average amount of time it takes for the number of undecayed atoms in a sample to **halve**.

For example: If we begin with 500 atoms of an element. The amount of time it takes for 250 of these atoms to decay is the half-life of that sample.



Importing Libraries

To simulate the random decay of radioactive nuclei we will need to import a function from the python math library.

```
from random import randint
```


Input Values

The user is asked to input the number of initial nuclei (which is stored as an **integer**) and the probability of decay (which is stored as a **float** – allowing it to take any value).

```
# Initial Number of nuclei
initialNuclei = int(input('What is the starting
number of nuclei?'))
currentNuclei = initialNuclei

# The probability that a given nucleus will decay
probability = float(input('\nEnter a percentage value
for the chance of a nucleus decaying each second i.e.
50: '))
```

Initial Values

Some initial values are defined so that the simulation begins at 0 seconds with the half life unknown (the reason this is required will become clear).

```
# Setting initial values for variables
```

```
seconds = 0
```

```
halflife = 0
```

```
halflifeFound = False
```

Main Program

The main body of the program consists of a **while** loop which will run until all the nuclei have decayed.

```
while currentNuclei > 0:

    # Print the number of nuclei remaining and increase the
    # time by 1
    print(currentNuclei)
    seconds += 1

    # Consider each nucleus in turn and decide whether it
    # decays or not
    for nuclei in range(currentNuclei):
        rand = randint(0, 100)
        if rand < probability:
            currentNuclei -= 1
```

Extension Task

As an extension to the main program the half life can be estimated. This is done within the `while` loop.

```
# Store the value of the halflife when more than half of  
the nuclei have decayed
```

```
if currentNuclei < round(initialNuclei / 2) and  
halflifeFound == False:
```

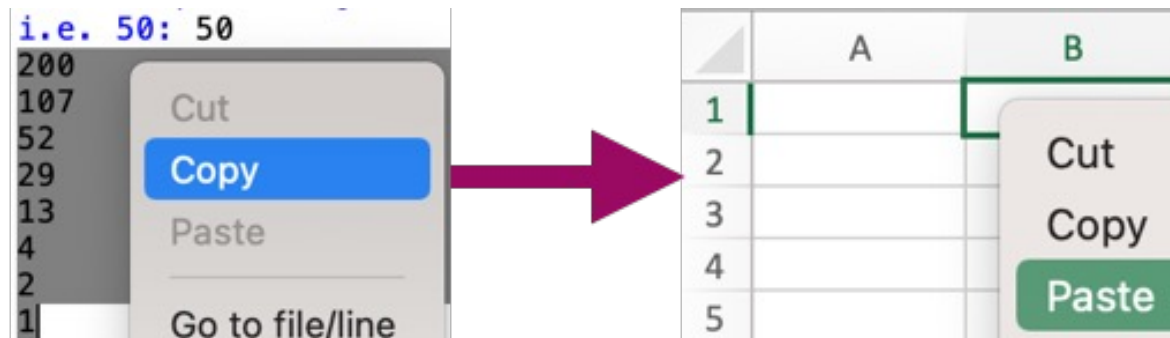
```
    halflife = seconds
```

```
    halflifeFound = True
```

```
print('The half life for this sample is between %d and  
%d seconds.' %(halflife - 1, halflife))
```

Plotting


The values outputted in the Python Shell can be copied into Excel in one by highlighting them all, copying and pasting in Excel.



Paste the values in column B, so that it plots as the y axis. Further results can be added in the following columns, to plot multiple simulations.

Plotting

In column A add the seconds passed from 0. This can be done by entering 0 and 1, selecting both, then dragging the bottom corner.

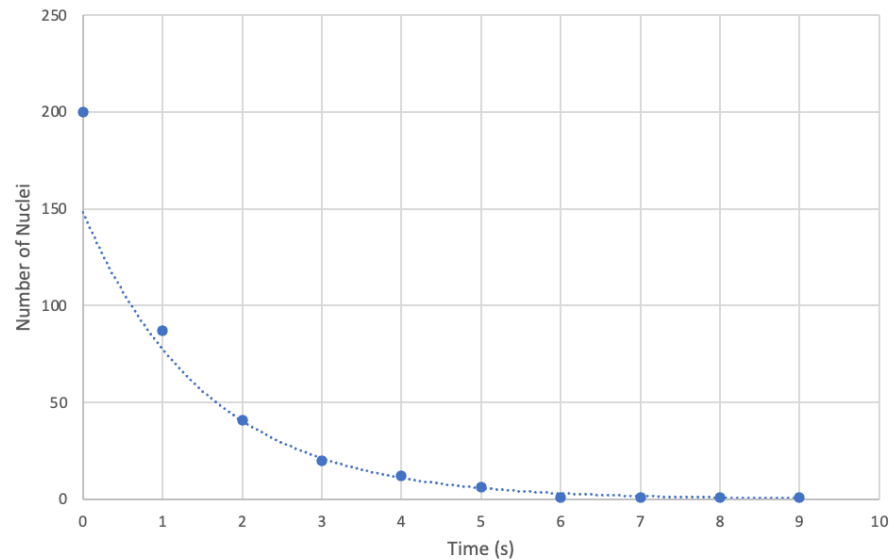


	A	B
1	0	200
2	1	107
3		52
4		29
5		13
6		4
7		2
8		1

	A	B
1	0	200
2	1	107
3	2	52
4	3	29
5	4	13
6	5	4
7	6	2
8	7	1

Plotting

Select all the data and plot through Insert > Charts > X Y (Scatter).

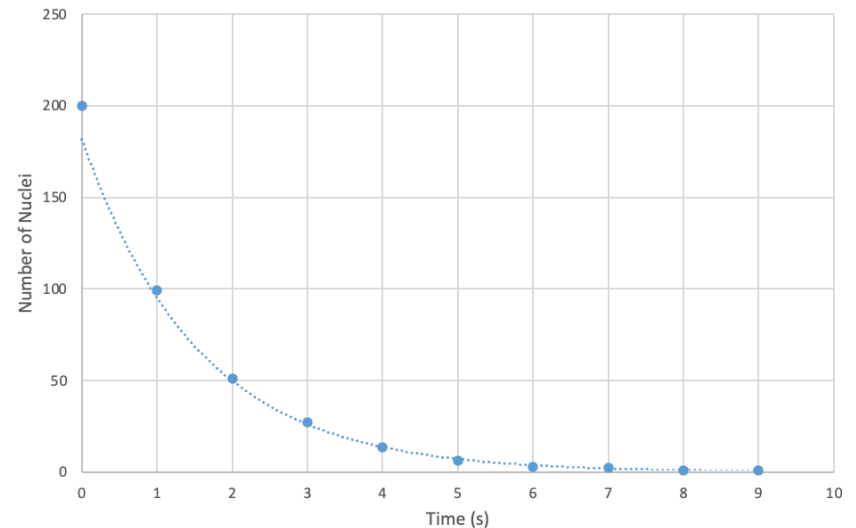
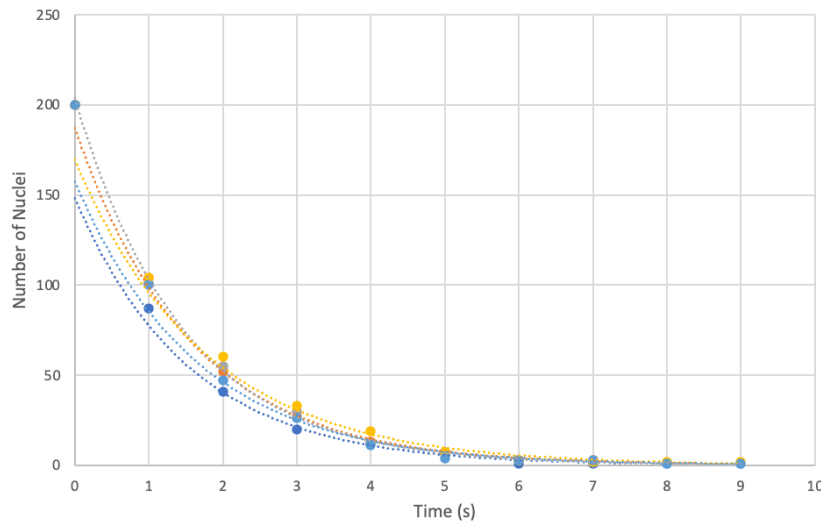


Add an Exponential Trendline from Add Chart Element to fit the data.

Plotting

Many trendlines could be fitted to multiple data sets on one axes, this would demonstrate the similarity in sets of random results.

Alternatively a trendline could instead be fitted to the average result of many simulations.





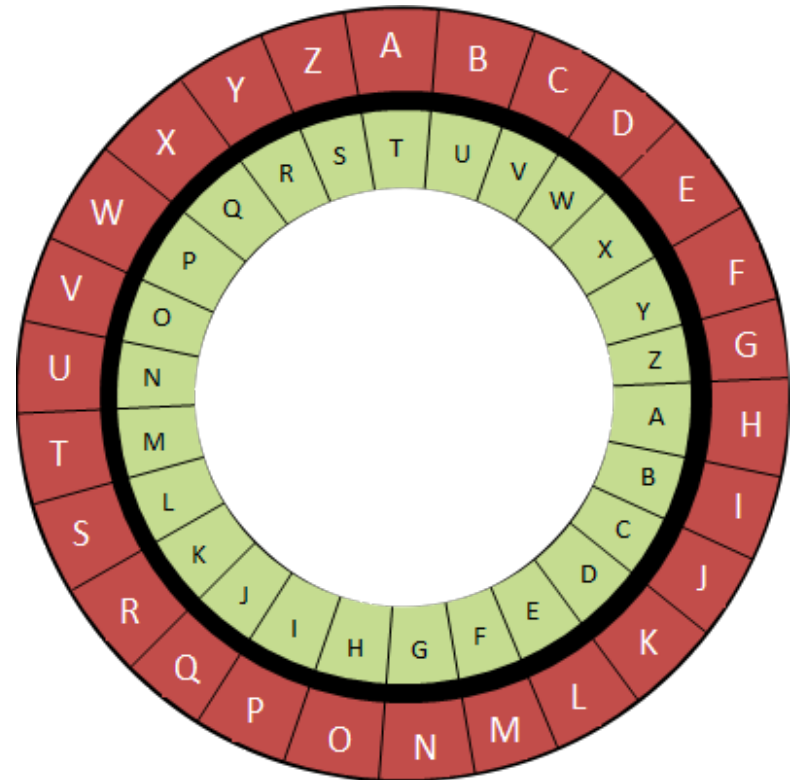
Caesar Cipher - Python

Caesar Cipher

A Caesar Cipher is an ancient method of encryption (named for Julius Caesar who was known to use it).

Each letter of the alphabet is shifted by a fixed number of letters (in this example they are shifted by 19, so that $A \rightarrow T$, $B \rightarrow U$ etc).

This is a simple cipher and can be coded simply.



Defining the Cipher

To build a working Caesar Cipher we will need to define the set of symbols we will be using to encrypt our text.

```
global alpha,bet  
alpha = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'  
bet = 'abcdefghijklmnopqrstuvwxyz'
```

The variables have been made **global** so that they can be used in any of the following functions.

Encrypt Function

This function will encrypt any plain text fed into it using key specified and the global variables.

```
def encrypt(text, key):  
    # The result is saved as an empty string and added to  
    result = ''
```

Encrypt Function

```
# For each character in the input text
for letter in text:
    if letter.isupper(): # if letter is uppercase
        num = alpha.find(letter) #find its position

        num += key # Shift the position by the key

# check if adding the key passes length of alphabet
if num > 25:
    num -= len(alpha)
result += alpha[num]
```

Encrypt Function

```
elif letter.islower(): # if lowercase
    num = bet.find(letter)
    num += key

    if num > 25:
        num -= len(bet)
    result += bet[num]
else:
    result += letter
```

Encrypt Function

```
# At the end of the text return the result from the  
function  
    return result
```

Note: There is no call or print command in this function - to call and `print` the function we could use something like `print(encrypt('My Plain Text' , 5))`

Decrypt Function

This function will decrypt any encrypted text fed into it using key specified and the global variables.

It is almost identical code to the encrypt function.

```
def decrypt(text, key):  
    # The result is saved as an empty string and added to  
    result = ''
```


Decrypt Function

```
for letter in text:

    if letter.isupper():
        num = alpha.find(letter)
        num -= key    # Subtracting this time

    if num < 0:
        num += len(alpha)
    result += alpha[num]
```

Decrypt Function

```
elif letter.islower(): # for lowercase letters
    num = bet.find(letter)
    num -= key

    if num < 0:
        num += len(bet)
    result += bet[num]

else:
    result += letter

return result
```

Decrypt Function

There is no call or print command in this function - to call and print the function we could use something like

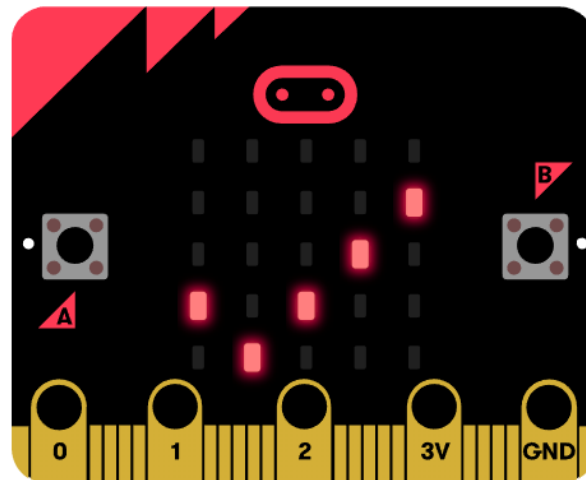
```
print(decrypt('Rd Uqfns Yjcy' , 5))
```



micro:bit Pedometer – Python

micro:bit

By using a micro:bit, we can create our own step counter.
A micro:bit is a small, micro computer



micro:bit Pedometer

Using a BBC micro:bit, we can program it to become a pedometer, and count our steps

We can check whenever it is shaken, and start counting how many steps we take

Making a Variable

Firstly, we will need to create a variable to track how many steps we take

This is simply done by making a variable, and setting it to 0

```
steps = 0
```

Adding a Condition

We now want to check when the micro:bit is moved, and to perform an action when it is

```
if accelerometer.was_gesture('shake') :
```

'if' is our conditional

'accelerometer' is the sensor we want to access

`was_gesture('shake')` is the condition we are checking

Manipulating the Variable

Now we have our conditional, we can manipulate the variable when it is triggered

This is done by taking the current version, and adding 1 to it

```
steps += 1
```

Showing the Variable

We can display the current number of steps on the micro:bit via the show command

```
display.show(steps)
```

Repeating our code

Now we have written all our code, we can make sure it runs for ever, constantly checking if we have stepped or not

This is done by putting the code inside a while loop

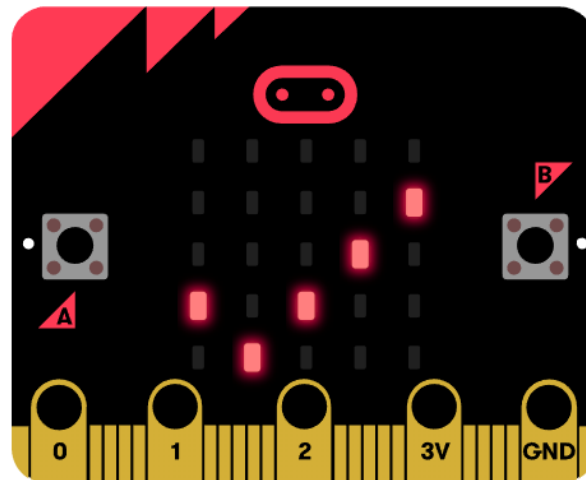
```
while True:
```



micro:bit Music Maker - Python

micro:bit

By using a micro:bit, we can create our own music.
A micro:bit is a small, micro computer.



MIDI Music Maker

MIDI is a technology standard which describes a communication protocol, interface & connectors for making electric instruments and sound devices

Using MIDI, we are able to make a variety of notes via electronic devices, allowing the composition and creation of music

Much of this is able to mimic existing instruments, i.e. the keyboard (hardware limiting)

Using Python, this is easily accessible, and can be done quickly and simply, and requires no specialist equipment

Importing Libraries

To use python to create sounds for us, we will be needing the music library. This can be imported quite easily, as we may want access to all of it.

```
import music
```

Playing Notes

Creating notes in python is quite easy!

We can simply specify the pitch of the note, and duration we want

```
music.play( [ 'NOTE:DURATION' ] )
```


Playing Notes: Note & Octave

```
music.play([ 'NOTE:DURATION' ])
```

Where NOTE is a musical note on a scale

```
music.play([ 'C4' ])
```

The letter being the note, and the number being the octave

```
music.play([ 'C4', 'C5' ])
```

This plays two notes, one after another, with C5 being one octave above the previous C4

Playing Notes: Length of Note

```
music.play([ 'NOTE:DURATION' ])
```

Where DURATION is the length of time we want it to play for

```
music.play([ 'C4:4' ])
```

The number showing how long many beats the note is played for

```
music.play([ 'C4:4', 'C4:8' ])
```

This plays two notes, the second note lasting for twice as long as the first

Creating Rhythms

```
music.play(['C4:4', 'D4', 'E4', 'C4'])
```

We can play notes together, to create melodies of note strung together

Adding Timing

```
music.play(['r'])
```

We can also manipulate the timings by adding a rest if needed

Duplicating Notes

```
for x in range(2):  
    music.play(['C4:4', 'D4', 'E4', 'C4'])  
    music.play(['r'])  
for x in range(2):  
    music.play(['E4:4', 'F4', 'G4:8'])
```

We can use all our standard python syntax (such as loops) to repeat bits of music

For example, these two lines will play the opening to Frère Jacques

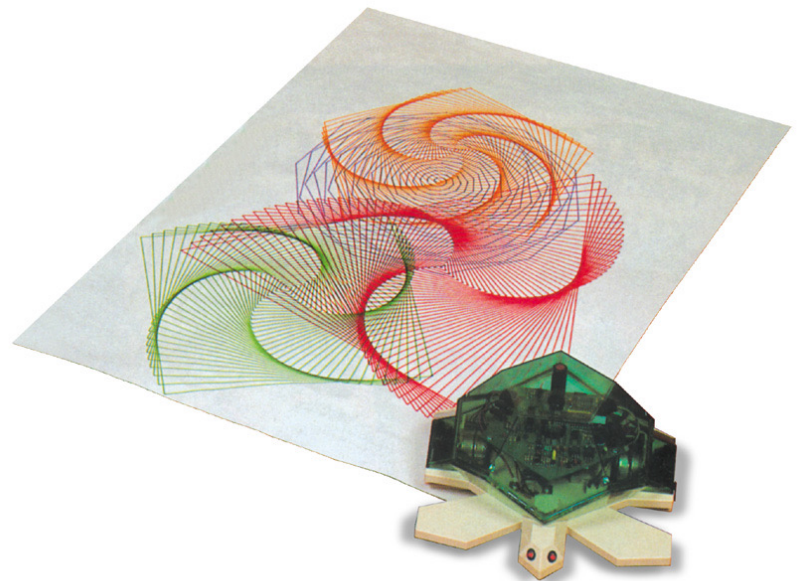
Making Art - Python



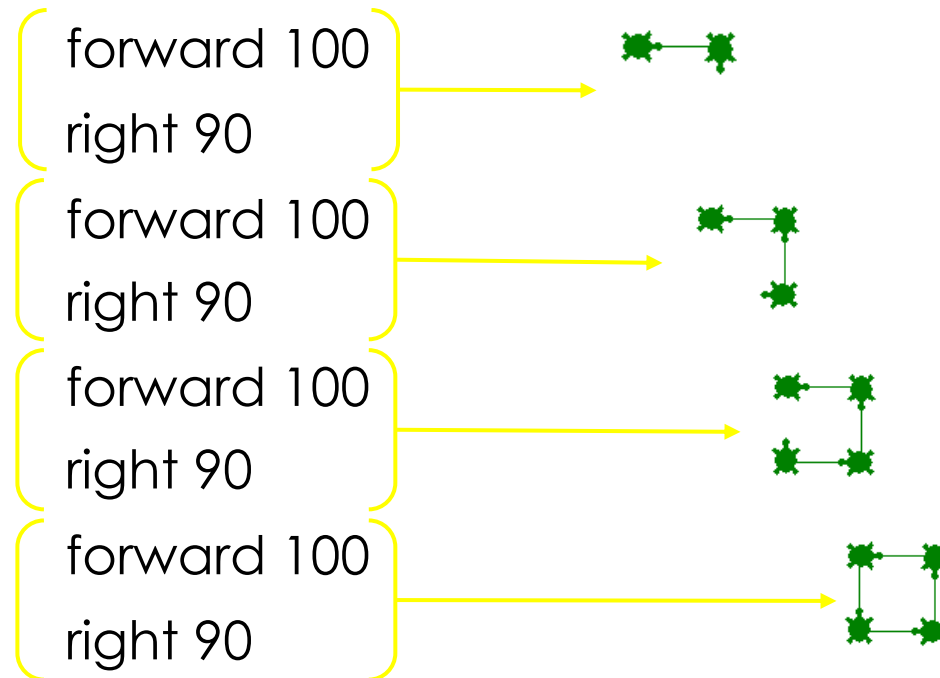
Drawing Shapes

Python can be used to draw shapes,
and create patterns

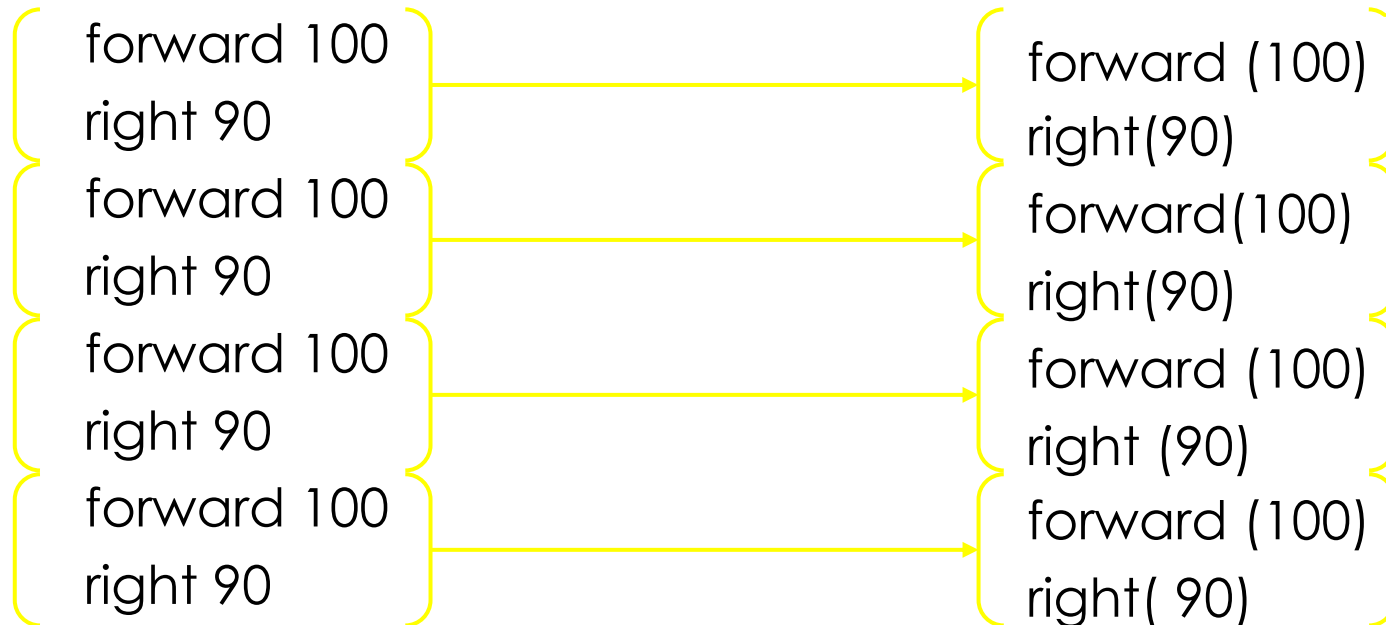
Using the `turtle` library, we can
make a turtle draw lines



Square - Instructions



Square - The Turtle Way



Turtle Drawing a Square

```
import turtle
```

Get the turtle commands

Turtle Drawing a Square

```
import turtle
```

Get the turtle commands

```
pen = turtle.Turtle()
```

Create a new Turtle

Turtle Drawing a Square

```
import turtle
```

Get the turtle commands

```
pen = turtle.Turtle()
```

Create a new Turtle

```
pen.shape("turtle")
```

Set the shape of the turtle

Turtle Drawing a Square

```
import turtle
```

Get the turtle commands

```
pen = turtle.Turtle()
```

Create a new Turtle

```
pen.shape("turtle")
```

Set the shape of the turtle

```
pen.forward(100)
```

Move the turtle forward by 100 steps

Turtle Drawing a Square

```
import turtle
```

```
pen = turtle.Turtle()
```

```
pen.shape("turtle")
```

```
pen.forward(100)
```

```
pen.right(90)
```

Get the turtle commands

Create a new Turtle

Set the shape of the turtle

Move the turtle forward by 100 steps

Turn right at 90 degrees

Turtle Drawing a Square

```
import turtle
pen = turtle.Turtle()
pen.shape("turtle")
pen.forward(100)
pen.right(90)
pen.forward(100)
pen.right(90)
pen.forward(100)
pen.right(90)
pen.forward(100)
pen.right(90)
```

Get the turtle commands

Create a new Turtle

Set the shape of the turtle

Move the turtle forward by 100 steps

Turn right at 90 degrees

Do it 3 more times

Loop it with a Variable

```
import turtle
pen = turtle.Turtle()
pen.shape("turtle")
pen.color("green")
for i in range(350):
    pen.forward(i)
    pen.right(98)
```

**Remember, the
loop variable is a
counter**

We can also use the loop variable (`i`) inside the design. The code on the left will produce the design below.

