technocamps

UNDEB EWROPEAIDD
EUROPEAN UNION

Llywodraeth Cymru
Welsh Government

Cronfa Gymdeithasol Ewrop
European Social Fund

Prifysgol Abertawe
Swansea University

CARDIFF UNIVERSITY
PRIFYSGOL CAERDYDD

PRIFYSGOL BANGOR UNIVERSITY

Cardiff Metropolitan University
Prifysgol Metropolitan Caerdydd

i.t.wales

PRIFYSGOL ABERYSTWYTH UNIVERSITY

PRIFYSGOL glyndŵr Wrecsam
Wrexham glyndŵr UNIVERSITY

University of South Wales
Prifysgol De Cymru

technocamps

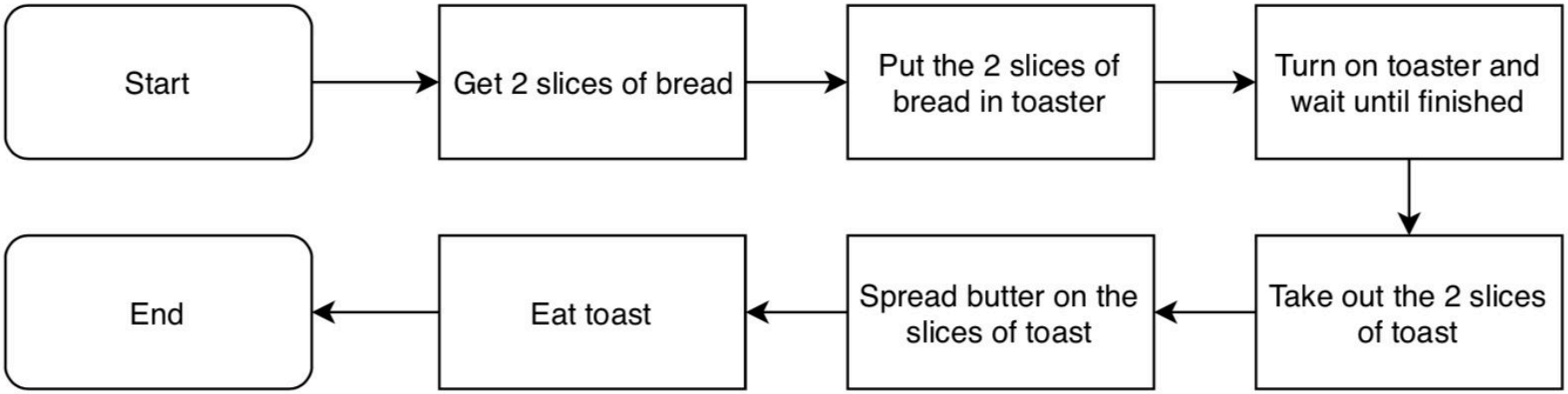
Algorithms II

Activity: Recap Algorithms

Recap Algorithms

An Algorithm is a set of simple instructions that are done in a certain order to solve a problem.

Here's an example: Making and eating toast.

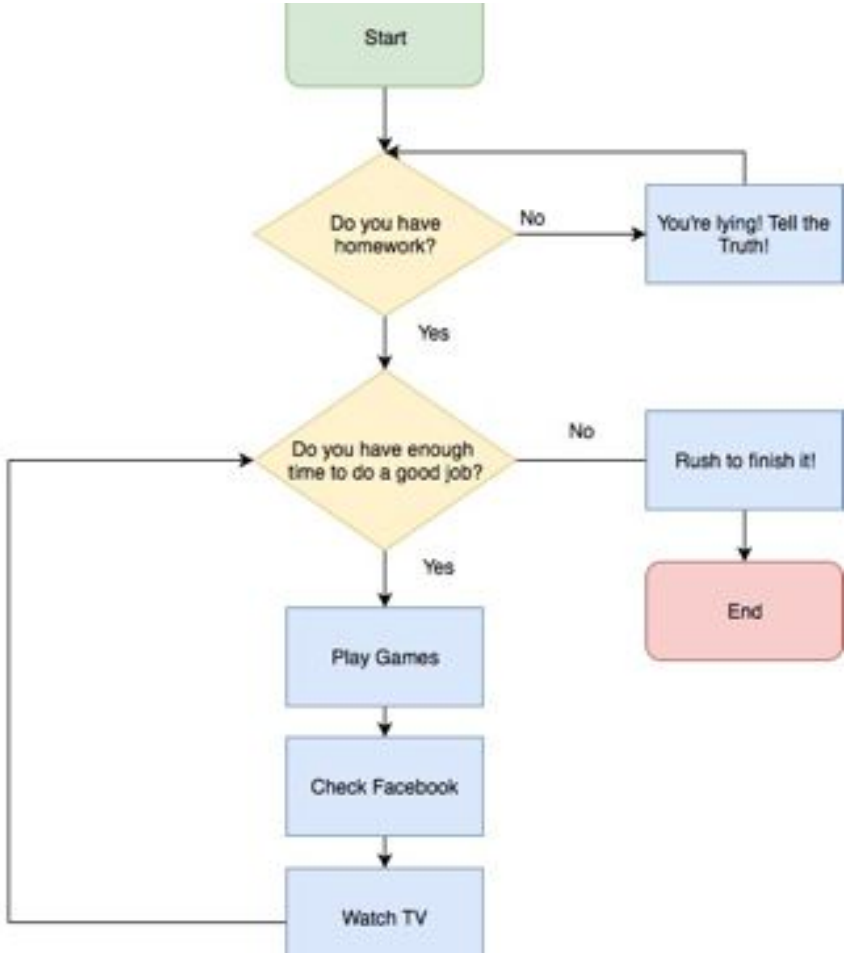


Algorithms

It is important to remember when writing an algorithm to keep instructions:

- Simple
- In the correct order
- Unambigious
- Relevant to solving the problem at hand

Where do we use algorithms in everyday life?



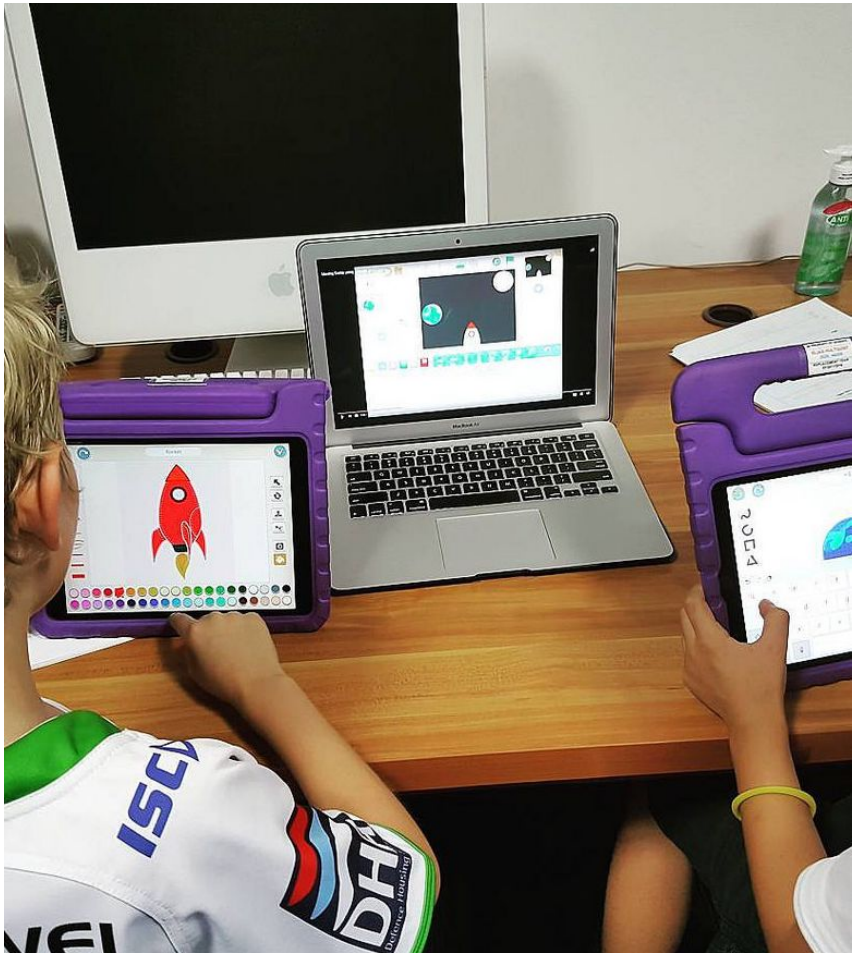
Recap Decomposition

Recap Decomposition

Decomposition is the process of breaking a complex problem down into smaller component parts.

Real world examples of using decomposition include:

- Creating a video game
- Complex maths problems
- Cooking
- Cleaning your room



Recap
Abstraction

Recap Abstraction

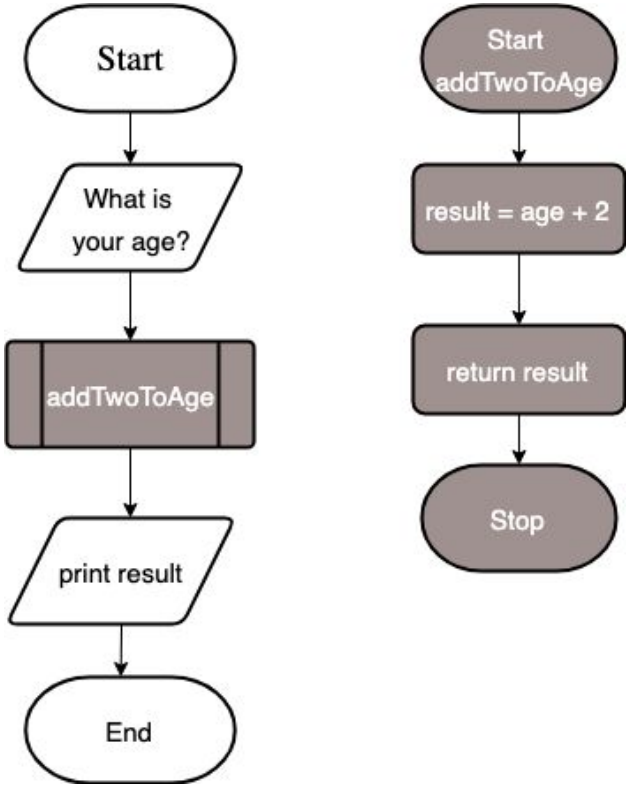
Abstraction is the process of removing unnecessary detail and simplifying.

Abstraction is used to remove unnecessary detail from a real-world situation and to model the simplified result in an algorithm or program.

Recap Subroutines

Recap Subroutines

Subroutines are a sequence of instructions that perform a specific task.

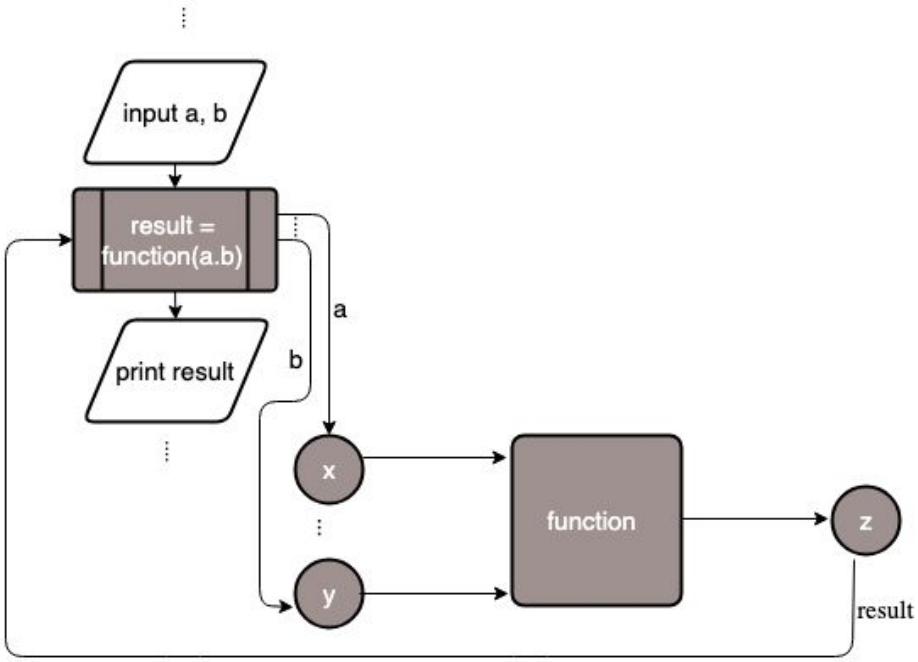


Subroutines in Python

Functions in Python

Subroutines are implemented as functions in Python.

A function is a subroutine that **usually takes in one or more values** from the program and then **returns a value back**.



Functions in Python - Example

```
#Functions in Python

def cubeVolume(side):
        vol = side ** 3
        return vol

sideLength = int(input("Enter side length of a cube: "))
result = cubeVolume(sideLength)
print ("The cube's volume is: ", result)
```

Functions in Python - Example

```
#Functions in Python

def cubeVolume(side):
        vol = side ** 3
        return vol

sideLength = int(input("Enter side length of a cube: "))
result = cubeVolume(sideLength)
print ("The cube's volume is: ", result)
```

Functions in Python - Example

```
#Functions in Python

def cubeVolume(side):
        vol = side ** 3
        return vol

sideLength = int(input("Enter side length of a cube: "))
result = cubeVolume(sideLength)
print ("The cube's volume is: ", result)
```

Functions in Python - Example

```
#Functions in Python

def cubeVolume(side):
        vol = side ** 3
        return vol

sideLength = int(input("Enter side length of a cube: "))
result = cubeVolume(sideLength)
print ("The cube's volume is: ", result)
```

Functions in Python - Example

```
#Functions in Python

def cubeVolume(side):
        vol = side ** 3
        return vol

sideLength = int(input("Enter side length of a cube: "))
result = cubeVolume(sideLength)
print ("The cube's volume is: ", result)
```

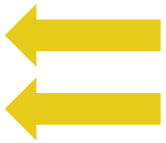
3 side = value of sideLength

1

2

Functions in Python – Example

```
#Functions in Python

def cubeVolume(side):
        vol = side ** 3
        return vol

sideLength = int(input("Enter side length of a cube: "))
result = cubeVolume(sideLength)
print ("The cube's volume is: ", result)
```



3 side = value of sideLength

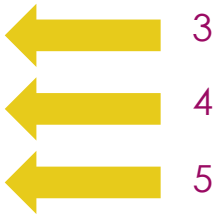
4

1

2

Functions in Python - Example

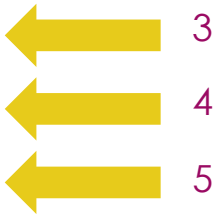
```
#Functions in Python


def cubeVolume(side):
        vol = side ** 3
        return vol


sideLength = int(input("Enter side length of a cube: "))
result = cubeVolume(sideLength)
print ("The cube's volume is: ", result)
```



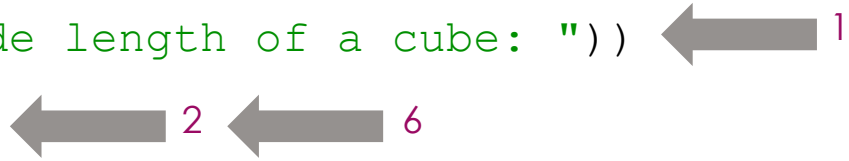
side = value of sideLength

Functions in Python - Example

```
#Functions in Python


def cubeVolume(side):
        vol = side ** 3
        return vol


sideLength = int(input("Enter side length of a cube: "))
result = cubeVolume(sideLength)
print ("The cube's volume is: ", result)
```

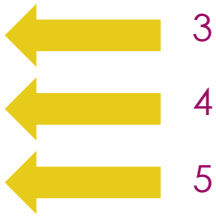


side = value of sideLength

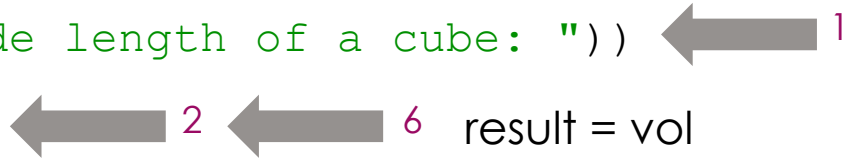


Functions in Python - Example

```
#Functions in Python


def cubeVolume(side):
        vol = side ** 3
        return vol


sideLength = int(input("Enter side length of a cube: "))
result = cubeVolume(sideLength)
print ("The cube's volume is: ", result)
```



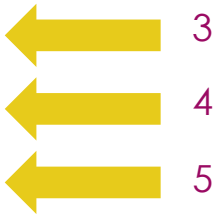
side = value of sideLength



result = vol

Functions in Python - Example

```
#Functions in Python


def cubeVolume(side):
        vol = side ** 3
        return vol


sideLength = int(input("Enter side length of a cube: "))
result = cubeVolume(sideLength)
print ("The cube's volume is: ", result)
```



side = value of sideLength

result = vol

Functions in Python – main()

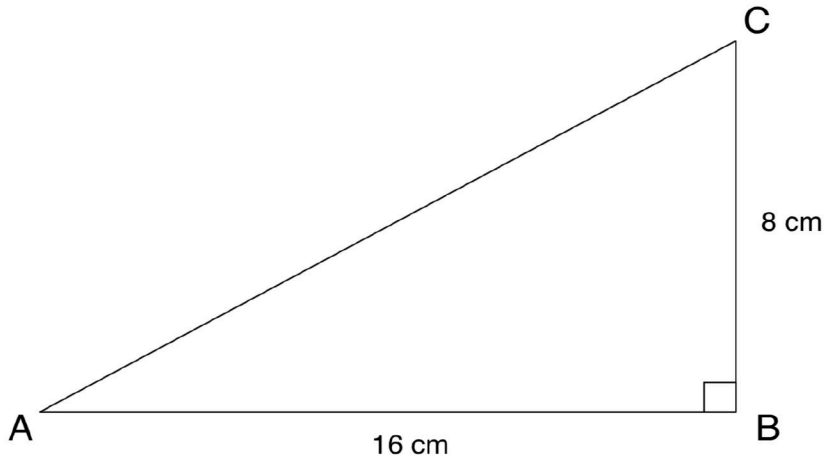
```
#Functions in Python


def cubeVolume(side):
        vol = side ** 3
        return vol
def main():
        sideLength = int(input("Enter side length of a cube: "))
        result = cubeVolume(sideLength)
        print("The cube's volume is: ", result)
main()
```

We have extended the previous program to define a function called main() and call other functions from it.
The Python interpreter will call main() as the entry point to our program.

Functions - Implementation

Consider the question below:

We have three points, A, B and C. We know the distance from A to B and B to C. We are asked to calculate the distance from A to C.



ABC is a right-angled triangle.
AB = 16 cm.
BC = 8 cm.

Calculate the length of AC

Let us create a function that can calculate the hypotenuse for any right-angled triangle.

Activity: Calculating the Hypotenuse

Functions - Implementation

Length of hypotenuse h = $\sqrt{(a^2 + b^2)}$

It is important to remember the **main()** function. The main function is the first function in our code. Inside this function is where we call/use all other functions that we have created.

It is the main body of the program and all other functions are the subroutines.

```
#A program that calculates the hypotenuse.
def main():

#Main entry to the program
main()
```

Define Hypotenuse Function

A function has three parts:

1. **Name:** Name of the function - `hypotenuseCalculator`
2. **Parameter(s):** Variables that provide input to the function. These variables exists only inside these functions. - `a, b`
3. **Body:** Block of code that processes the input(s) and returns a value. - `return hypotenuse`

The first line of the function begins with the keyword **def** which stands for define.

```
def hypotenuseCalculator(a, b):
    hypotenuse = (a ** 2 + b ** 2) ** 0.5
    return hypotenuse
```

User Output - Refresher

print("...") is a function that prints the given output data to the console.

The following print function prints "Python is fun." in the console.

```
print("Python is fun.")
```

We can combine multiple texts using a comma as a separator.

```
a = 5
b = 10
print("a = ", a, " b = ", b)
```

User Input - Refresher

input("...") is a function that prints the given question to the console and waits for the user to provide an input.

The following input function prints "Is Python fun? [Y/N] " on the console and expects the user to provide an answer.

```
input("Is Python fun? [Y/N] ")
```

The program will not proceed until the user has given an input.

Input as a String

User input comes into Python as a string, which means that when you type the number 10 on your keyboard, Python saves the number 10 into a variable as a string, not as a number.

These two statements are different in how the computer processes them.

```
age = 10 #This is a number
age = "10" #This is a text/string
```

The result of an **input("…")** function is always a string. To convert it to a number (int, float etc.) we need to use the appropriate converter function.

Converter Functions

The **int function** converts a string or a number into a whole number (an integer), which basically means that everything after the decimal point is dropped.

int(123.456) = 123

int('123') = 123

The **float function** converts a string or a number into a *floating-point* number, which is a number with a decimal place.

float(12) = 12.0

float("123.456") = 123.456

Define Main Function

Once we have created the hypotenuse function, all that is needed to be done is to call that function with the input values inside the main function.

```
def main():
    ab = int(input("Enter AB length: "))
    bc = int(input("Enter BC length: "))
    ac = hypotenuseCalculator(ab, bc)
    print("Length of AC is: ", ac)
```

Hypotenuse Program

The whole program should look like this:

```
# A program that calculates the hypotenuse.
def main():
    ab = int(input("Enter AB length: "))
    bc = int(input("Enter BC length: "))
    ac = hypotenuseCalculator(ab, bc)
    print("Length of AC is: ", ac)

def hypotenuseCalculator(a, b):
    h = (a ** 2 + b ** 2) ** 0.5
    return h

main()
```

Hypotenuse Program - Execution

The whole program should look like this:

```
# A program that calculates the hypotenuse.
def main():
    ab = int(input("Enter AB length: "))
    bc = int(input("Enter BC length: "))
    ac = hypotenuseCalculator(ab, bc)
    print("Length of AC is: ", ac)

def hypotenuseCalculator(a, b):
    h = (a ** 2 + b ** 2) ** 0.5
    return h

main()
```

Hypotenuse Program - Execution

The whole program should look like this:

```
# A program that calculates the hypotenuse.
def main():
    ab = int(input("Enter AB length: "))
    bc = int(input("Enter BC length: "))
    ac = hypotenuseCalculator(ab, bc)
    print("Length of AC is: ", ac)

def hypotenuseCalculator(a, b):
    h = (a ** 2 + b ** 2) ** 0.5
    return h

main()
```

2

1

Hypotenuse Program - Execution

The whole program should look like this:

```
# A program that calculates the hypotenuse.
def main():
    ab = int(input("Enter AB length: "))
    bc = int(input("Enter BC length: "))
    ac = hypotenuseCalculator(ab, bc)
    print("Length of AC is: ", ac)

def hypotenuseCalculator(a, b):
    h = (a ** 2 + b ** 2) ** 0.5
    return h

main()
```

Hypotenuse Program - Execution

The whole program should look like this:

```
# A program that calculates the hypotenuse.
def main():                                          2
    ab = int(input("Enter AB length: "))             3
    bc = int(input("Enter BC length: "))             4
    ac = hypotenuseCalculator(ab, bc)
    print("Length of AC is: ", ac)

def hypotenuseCalculator(a, b):
    h = (a ** 2 + b ** 2) ** 0.5
    return h

main()                                               1
```

Hypotenuse Program - Execution

The whole program should look like this:

```
# A program that calculates the hypotenuse.
def main():                                          2
    ab = int(input("Enter AB length: "))             3
    bc = int(input("Enter BC length: "))             4
    ac = hypotenuseCalculator(ab, bc)                5
    print("Length of AC is: ", ac)

def hypotenuseCalculator(a, b):
    h = (a ** 2 + b ** 2) ** 0.5
    return h

main()                                               1
```

Hypotenuse Program - Execution

The whole program should look like this:

```
# A program that calculates the hypotenuse.
def main():                                          2
    ab = int(input("Enter AB length: "))             3
    bc = int(input("Enter BC length: "))             4
    ac = hypotenuseCalculator(ab, bc)                5
    print("Length of AC is: ", ac)

def hypotenuseCalculator(a, b):                      6
    h = (a ** 2 + b ** 2) ** 0.5
    return h

main()                                               1
```

Hypotenuse Program - Execution

The whole program should look like this:

```
# A program that calculates the hypotenuse.
def main():                                          2
    ab = int(input("Enter AB length: "))             3
    bc = int(input("Enter BC length: "))             4
    ac = hypotenuseCalculator(ab, bc)                5
    print("Length of AC is: ", ac)

def hypotenuseCalculator(a, b):                      6   a = ab, b = bc
    h = (a ** 2 + b ** 2) ** 0.5
    return h

main()                                               1
```

Hypotenuse Program - Execution

The whole program should look like this:

```
# A program that calculates the hypotenuse.
def main():                                          2
    ab = int(input("Enter AB length: "))             3
    bc = int(input("Enter BC length: "))             4
    ac = hypotenuseCalculator(ab, bc)                5
    print("Length of AC is: ", ac)

def hypotenuseCalculator(a, b):                      6   a = ab, b = bc
    h = (a ** 2 + b ** 2) ** 0.5                     7
    return h

main()                                               1
```

Hypotenuse Program - Execution

The whole program should look like this:

```
# A program that calculates the hypotenuse.
def main():                                          2
    ab = int(input("Enter AB length: "))             3
    bc = int(input("Enter BC length: "))             4
    ac = hypotenuseCalculator(ab, bc)                5
    print("Length of AC is: ", ac)

def hypotenuseCalculator(a, b):                      6   a = ab, b = bc
    h = (a ** 2 + b ** 2) ** 0.5                     7
    return h                                         8

main()                                               1
```

Hypotenuse Program - Execution

The whole program should look like this:

```
# A program that calculates the hypotenuse.
def main():                                          2
    ab = int(input("Enter AB length: "))             3
    bc = int(input("Enter BC length: "))             4
    ac = hypotenuseCalculator(ab, bc)                5      9
    print("Length of AC is: ", ac)

def hypotenuseCalculator(a, b):                      6   a = ab, b = bc
    h = (a ** 2 + b ** 2) ** 0.5                     7
    return h                                         8

main()                                               1
```

Hypotenuse Program - Execution

The whole program should look like this:

```
# A program that calculates the hypotenuse.
def main():                                          2
    ab = int(input("Enter AB length: "))             3
    bc = int(input("Enter BC length: "))             4
    ac = hypotenuseCalculator(ab, bc)                5        9 ac=h
    print("Length of AC is: ", ac)

def hypotenuseCalculator(a, b):                      6   a = ab, b = bc
    h = (a ** 2 + b ** 2) ** 0.5                     7
    return h                                         8

main()                                               1
```

Hypotenuse Program - Execution

The whole program should look like this:

```
# A program that calculates the hypotenuse.
def main():                                          2
    ab = int(input("Enter AB length: "))             3
    bc = int(input("Enter BC length: "))             4
    ac = hypotenuseCalculator(ab, bc)                5        9 ac=h
    print("Length of AC is: ", ac)                   10

def hypotenuseCalculator(a, b):                      6   a = ab, b = bc
    h = (a ** 2 + b ** 2) ** 0.5                     7
    return h                                         8

main()                                               1
```

Hypotenuse Program - Execution

The whole program should look like this:

```
# A program that calculates the hypotenuse.
def main():                                                    2
    ab = int(input("Enter AB length: "))                       3
    bc = int(input("Enter BC length: "))                       4
    ac = hypotenuseCalculator(ab, bc)                          5          9 ac=h
    print("Length of AC is: ", ac)                                  10

def hypotenuseCalculator(a, b):                                6    a = ab, b = bc
    h = (a ** 2 + b ** 2) ** 0.5                                   7
    return h                                                       8

main()                                                     1              11
```

Activity: Currency Converter

Write a program which can convert Pounds(£) to Euros(€), and vice versa. Implement the flowchart on the following slide in Python using functions.

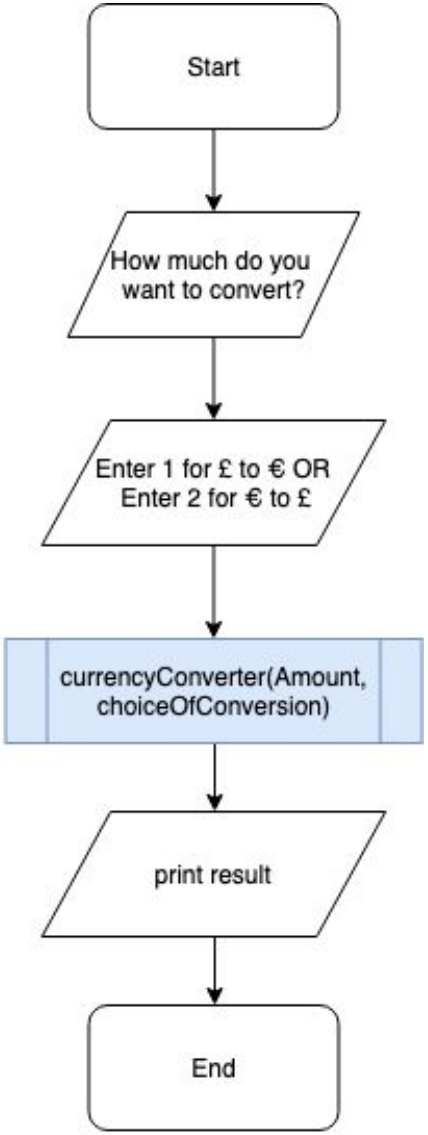
main():

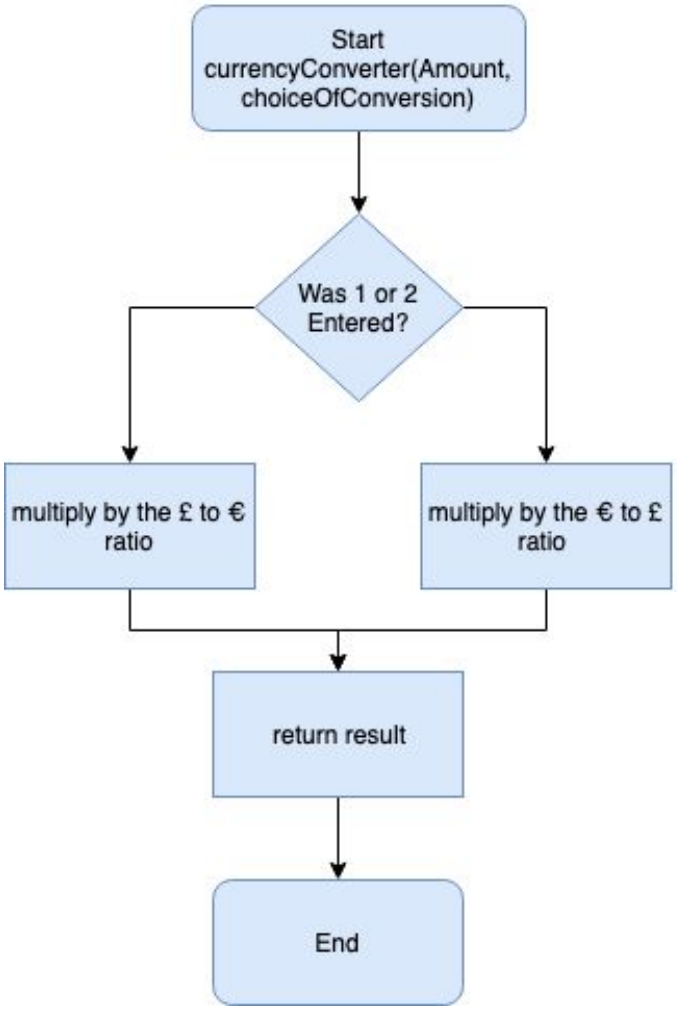
1. Ask the user to enter "1" to convert £ to € or "2" to convert € to £.
2. Call the function that performs the conversions with these inputs.

currencyConverter():

1. Takes in two inputs: user's choice of 1 or 2, value to be converted.
2. The function then performs the calculation (Note: research must be done to find out the conversion rates for pounds and euros.)
3. Returns the result to two decimal places.

Activity: Currency Converter





Activity: Temperature Converter

Implement the following converter which prompts the user to enter a temperature in degrees Celsius and then converts this to degrees Fahrenheit. Implement the flowchart in Python using functions.

The program should produce the following output:

Degrees Celsius: xx.xx

Degrees Fahrenheit: yy.yy

Where xx.xx and yy.yy are the temperatures to 2 decimal places displayed in Celsius and Fahrenheit respectively. Hint: To convert the temperate in degrees Celsius to degrees Fahrenheit use the following equation: **Fahrenheit = (Celsius x 9/5) + 32**

Hint: Be aware of integer division!

Activity: Jacket Potato

A jacket potato vendor has asked you to write a program to help calculate prices for his shop. The customer will be asked two questions: would they like either a medium or large size jacket potato, and the number of toppings they would like.

Implement in Python, a program which prompts the user to enter the letter "M" for medium and "L" for large. Next ask the user for the number of toppings they would like to have. The total price is calculated according to the following table:

	Up to 2 toppings	3 or more toppings
Medium Jacket Potato	£2.50 + 50p / topping	£2.50 + 40p / topping
Large Jacket Potato	£3.50 + 55p / topping	£3.50 + 45p / topping

Activity: Jacket Potato

The code should then print the total cost of their order. An example run might be:

Medium (M) or Large (L) jacket potato: L

Number of toppings: 3

Total cost: 4.85 pounds

The cost is calculated as 3.50 + (0.45 * 3) = 4.85.

Hint: Loops are not required for this program, but nested if statements are.

Activity: Lottery

The national lottery has contacted you to make a new lottery game.

The game will ask the user how many weeks they want to play and for 3 numbers they want to select; First between 1-10, second between 11-20 and third between 21-30.

If they match 1 number they win £10, 2 numbers £500, 3 numbers £1,000,000.

Activity: Lottery

For the number of weeks playing:

1. Generate a winning number between 1-10

Activity: Lottery

For the number of weeks playing:

1. Generate a winning number between 1-10
2. Generate a winning number between 11-20

Activity: Lottery

For the number of weeks playing:

1. Generate a winning number between 1-10
2. Generate a winning number between 11-20
3. Generate a winning number between 21-30

Activity: Lottery

For the number of weeks playing:

1. Generate a winning number between 1-10
2. Generate a winning number between 11-20
3. Generate a winning number between 21-30
4. If num1 matches winningNum1 then: add one to counter

Activity: Lottery

For the number of weeks playing:

1. Generate a winning number between 1-10
2. Generate a winning number between 11-20
3. Generate a winning number between 21-30
4. If num1 matches winningNum1 then: add one to counter
5. If num2 matches winningNum2 then: add one to counter

Activity: Lottery

For the number of weeks playing:

1. Generate a winning number between 1-10
2. Generate a winning number between 11-20
3. Generate a winning number between 21-30
4. If num1 matches winningNum1 then: add one to counter
5. If num2 matches winningNum2 then: add one to counter
6. If num3 matches winningNum3 then: add one to counter

Activity: Lottery

For the number of weeks playing:

1. Generate a winning number between 1-10
2. Generate a winning number between 11-20
3. Generate a winning number between 21-30
4. If num1 matches winningNum1 then: add one to counter
5. If num2 matches winningNum2 then: add one to counter
6. If num3 matches winningNum3 then: add one to counter
7. If counter = 1, Then win £10

Activity: Lottery

For the number of weeks playing:

1. Generate a winning number between 1-10
2. Generate a winning number between 11-20
3. Generate a winning number between 21-30
4. If num1 matches winningNum1 then: add one to counter
5. If num2 matches winningNum2 then: add one to counter
6. If num3 matches winningNum3 then: add one to counter
7. If counter = 1, Then win £10
8. If counter = 2, Then win £500

Activity: Lottery

For the number of weeks playing:

1. Generate a winning number between 1-10
2. Generate a winning number between 11-20
3. Generate a winning number between 21-30
4. If num1 matches winningNum1 then: add one to counter
5. If num2 matches winningNum2 then: add one to counter
6. If num3 matches winningNum3 then: add one to counter
7. If counter = 1, Then win £10
8. If counter = 2, Then win £500
9. If counter = 3, Then win £1,000,000

Activity: Lottery

For the number of weeks playing:

1. Generate a winning number between 1-10
2. Generate a winning number between 11-20
3. Generate a winning number between 21-30
4. If num1 matches winningNum1 then: add one to counter
5. If num2 matches winningNum2 then: add one to counter
6. If num3 matches winningNum3 then: add one to counter
7. If counter = 1, Then win £10
8. If counter = 2, Then win £500
9. If counter = 3, Then win £1,000,000
10. Print winning numbers + how much they won.

Extension Activity: Lottery

Try adding in ticket costs, then the game can include profit, amount loss, or even running the game until a profit is made!

```
print("You played for ", i, "weeks:")
print("You spent £", i * ticketCost, "Trying to win, ouch..")
profit = winnings - (i * ticketCost)
print("You ended up profiting £", profit)
```

Counting Heads

We want to create a program that allows us to repeatedly throw multiple coins and each time they've been thrown, remove them if they land on heads.

Here's a visual example in Scratch of the program we want to recreate in Python:

Coin Throwing Simulation

Activity: Counting Heads

Your task is to simulate the amount of throws and how many coins are remaining after each throw.

Think about what variables you will need to keep track of.

Extension:

Keep track of when half of the coins had been removed. i.e. after which throw.

Adjust it for rolling dice instead of coins

```
Please enter the initial number of coins: 200

Throw = 0
Number of Coins remaining = 200

Throw = 1
Number of Coins remaining = 102

Throw = 2
Number of Coins remaining = 50

Throw = 3
Number of Coins remaining = 27

Throw = 4
Number of Coins remaining = 17

Throw = 5
Number of Coins remaining = 7

Throw = 6
Number of Coins remaining = 4

Throw = 7
Number of Coins remaining = 0

Half of the coins were removed after throw 2.
```

Activity: Counting Heads

1. Enter the initial number of coins

Activity: Counting Heads

1. Enter the initial number of coins
2. Set the currentNumberOfCoins to initial coins

Activity: Counting Heads

1. Enter the initial number of coins
2. Set the currentNumberOfCoins to initial coins
3. Enter in the chanceOfHeads

Activity: Counting Heads

1. Enter the initial number of coins
2. Set the currentNumberOfCoins to initial coins
3. Enter in the chanceOfHeads
4. While the currentNumberOfCoins is more than zero

Activity: Counting Heads

1. Enter the initial number of coins
2. Set the currentNumberOfCoins to initial coins
3. Enter in the chanceOfHeads
4. While the currentNumberOfCoins is more than zero
    a) For each coin in the range of 0 -> currentNumberOfCoins

Activity: Counting Heads

1. Enter the initial number of coins
2. Set the currentNumberOfCoins to initial coins
3. Enter in the chanceOfHeads
4. While the currentNumberOfCoins is more than zero
    a) For each coin in the range of 0 -> currentNumberOfCoins
    b) headsOrTails = random number between 0 and 1

Activity: Counting Heads

1. Enter the initial number of coins
2. Set the currentNumberOfCoins to initial coins
3. Enter in the chanceOfHeads
4. While the currentNumberOfCoins is more than zero
    a) For each coin in the range of 0 -> currentNumberOfCoins
    b) headsOrTails = random number between 0 and 1
    c) If headsOrTails == 1 then take away a coin

Activity: Counting Heads

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

Activity: Counting Heads

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

Extensions:

Keep track of when half of the coins had been removed. i.e. after which throw

Adjust it for rolling dice instead of throwing coins

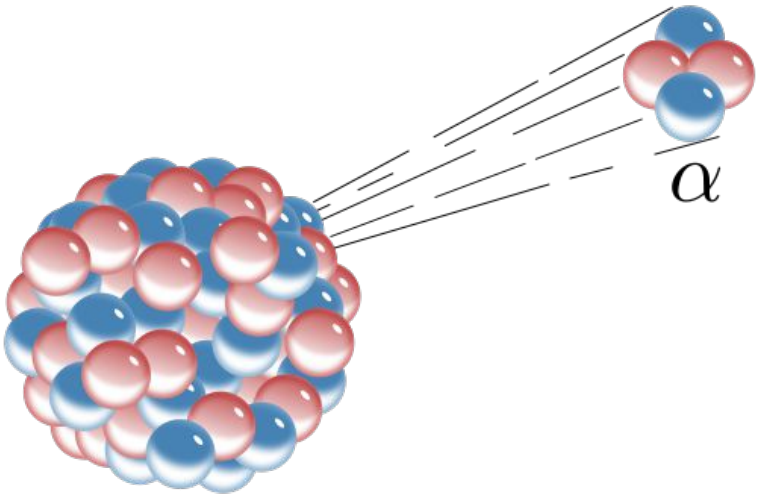
Activity: Visualising The Program

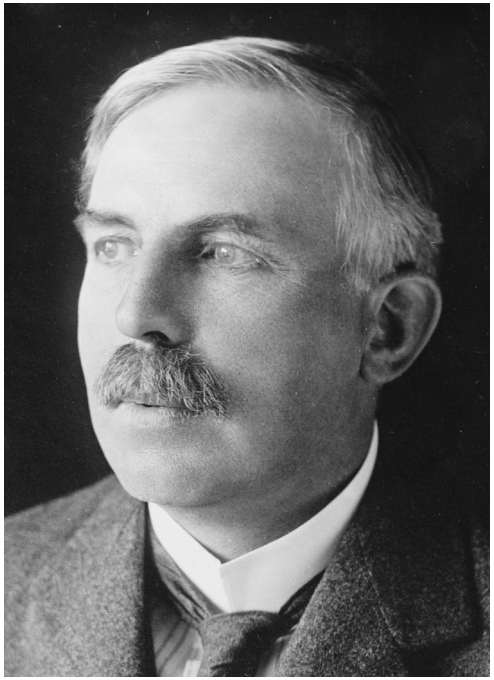
You can take the results from the program and add them to an excel sheet and visualise the coins removed after each throw.

Here is an example graph with number of coins set to 200.

Nuclear Physics

Believe it or not, you have just simulated a rather difficult concept in Nuclear Physics without even knowing it!



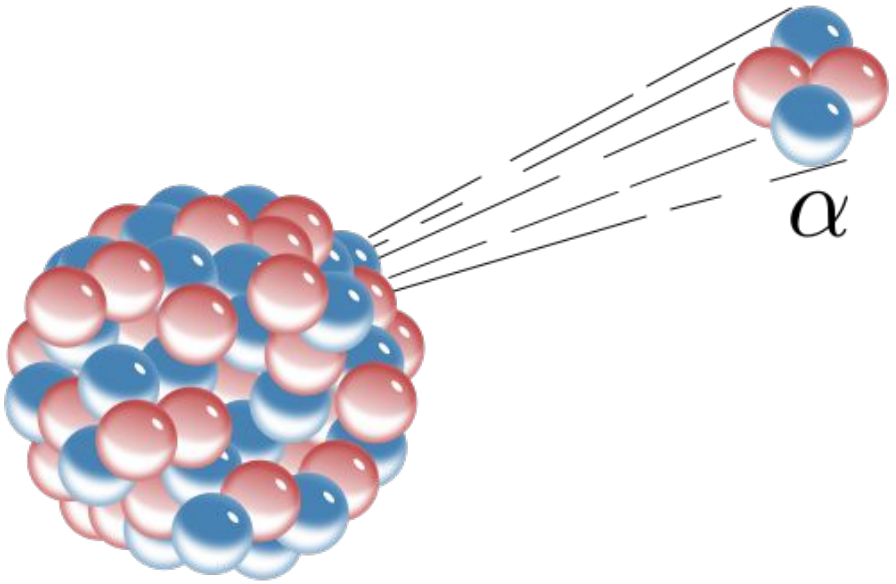


Ernest Rutherford – Discovered the Structure of Atoms

Radioactive Decay & Half-Life

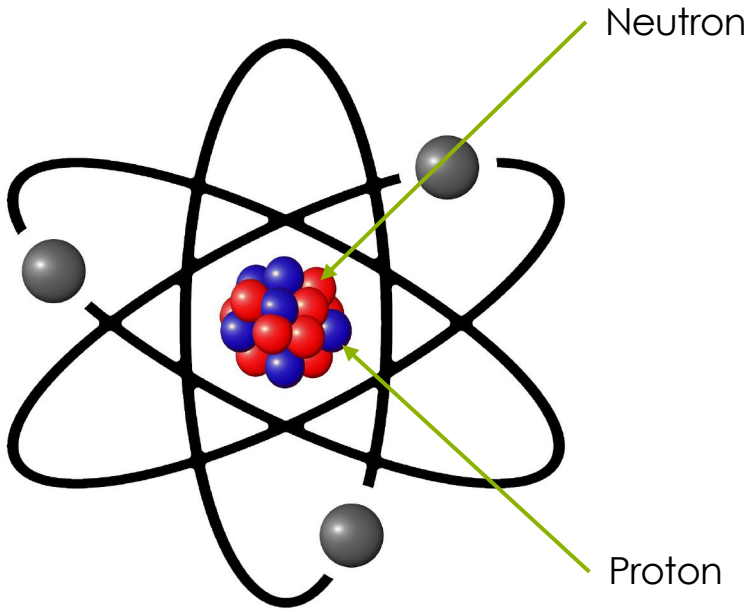
What is Radioactive Decay?

What is Half-Life?



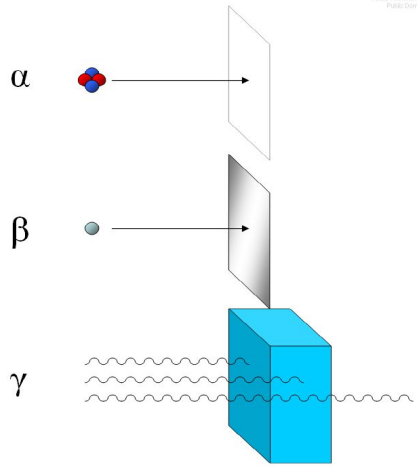
Radioactive Decay

Radioactive decay is the process which an unstable nucleus goes through due to an imbalance between proton and neutron numbers. Radioactive decay produces radiation which can be harmful in high doses!



Radioactive Decay

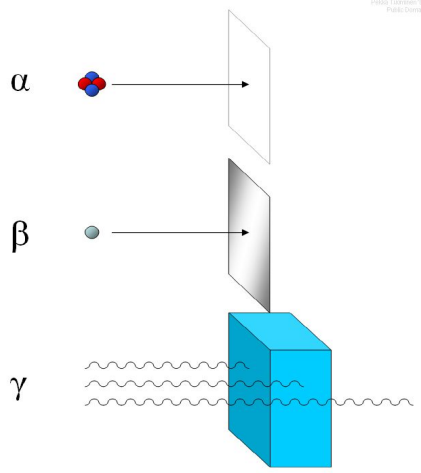
- Alpha Decay
- Beta Decay
- Gamma Radiation



What is it stopped by?

Radioactive Decay

- **Alpha Decay**
- **Beta Decay**
- **Gamma Radiation**

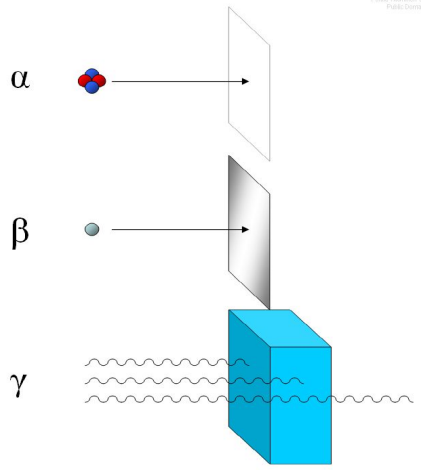


What is it stopped by?

Paper

Radioactive Decay

- **Alpha Decay**
- **Beta Decay**
- **Gamma Radiation**



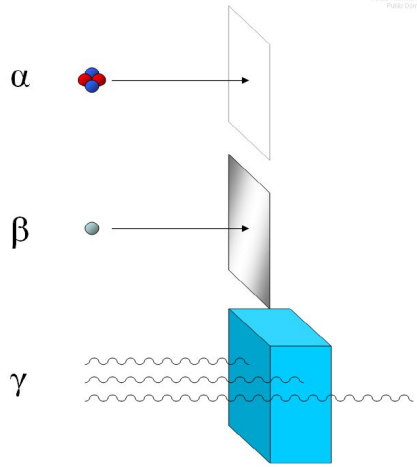
What is it stopped by?

Paper

Thin aluminium plate

Radioactive Decay

- **Alpha Decay**
- **Beta Decay**
- **Gamma Radiation**



What is it stopped by?

Paper

Thin aluminium plate

Nothing, but is **reduced** by heavily dense materials (Lead, Concrete etc.)

Which kind of radiation do you think is most similar to x-rays?

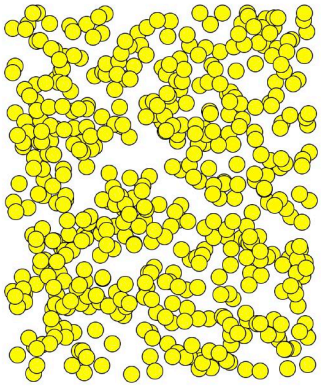
If you wanted to measure the perfect thickness of toilet paper, which one would you use to do this?

Half-Life

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

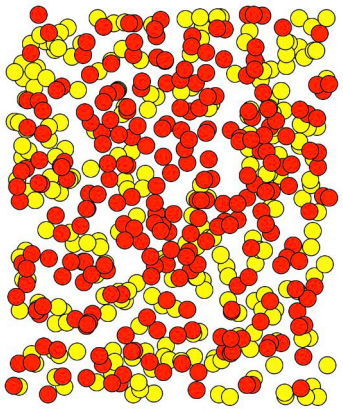
For example: If we begin with 500 atoms of an element. The amount of time it takes for 250 of these atoms to decay is the half-life of that sample.

500 Atoms



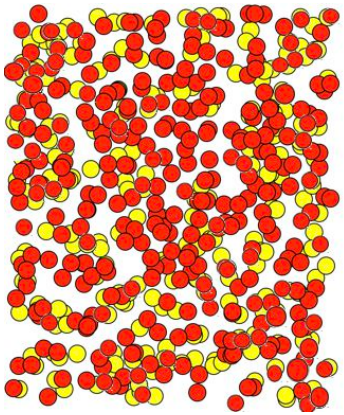
Time = 0 seconds

250 Atoms



Time = 6 seconds

125 Atoms

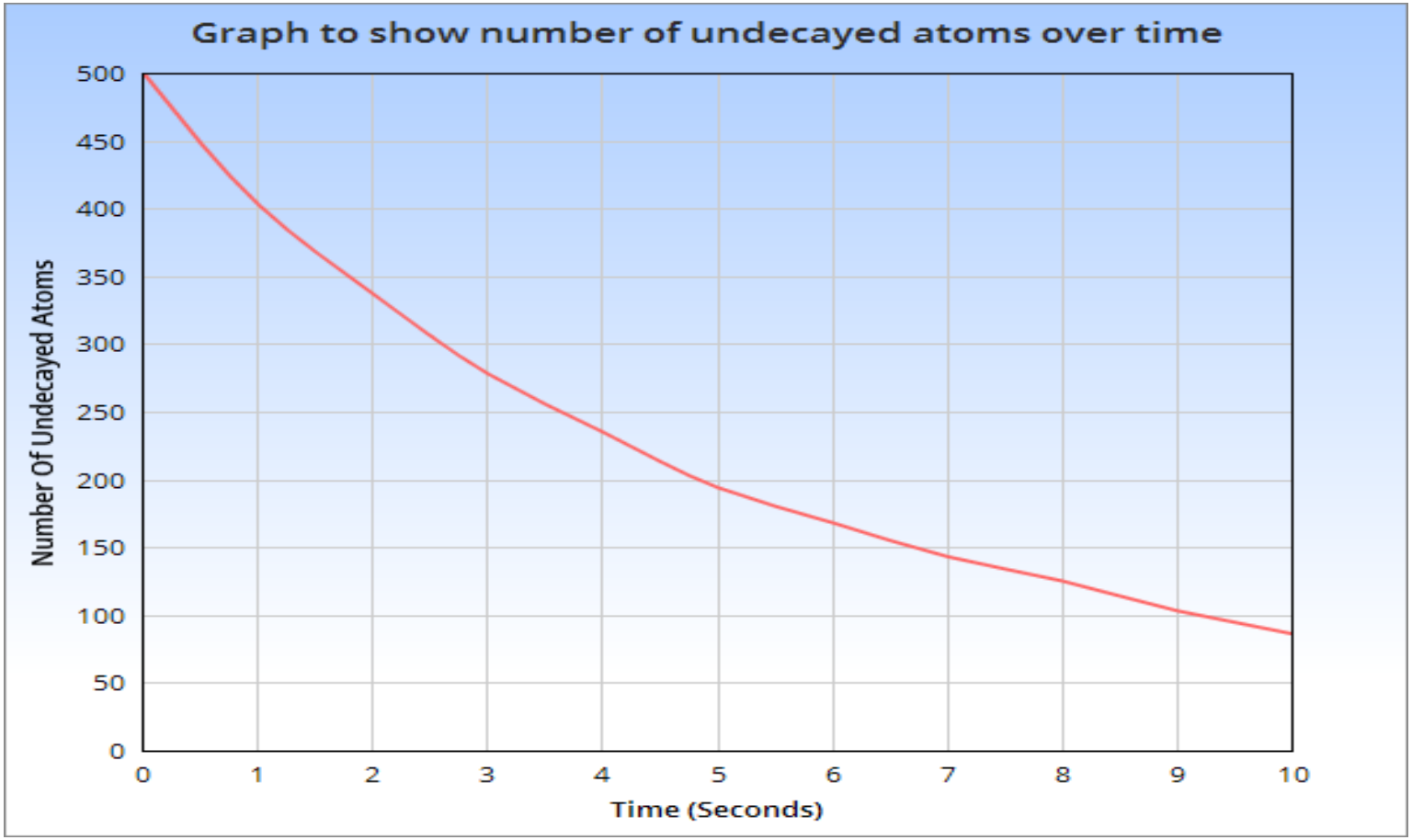


Time = 12 seconds

- Undecayed Atoms
- Decayed Atoms

So the half-life for this sample is 6 seconds.

Half-Life Visualised



Radioactive Decay

We measure radioactive decay using a Geiger counter (Geiger-Müller counter)

Every time the tube is hit by radiation it makes a clicking sound.

Chernobyl Nuclear Power Plant

On 26 April 1986 one of the nuclear reactors at the Chernobyl Nuclear Power Plant exploded during a safety test. This released a huge amount of radioactive material into the air which spread over 9 days.

Around 350,000 people have been evacuated since the explosion and the nearby villages have been abandoned.

Chornobyl Nuclear Power Plant

House and Primary School

Radioactivity at Chernobyl

Most areas had very low radioactivity, around 0.1 microSieverts per hour. (The safe amount is around 26 microSieverts per day)

However there are "hotspots" where it is much higher.

Half-Life and Carbon-Dating

Knowing how long a sample will remain at a certain level of radioactivity helps us decide which radioactive elements should be used in different situations.

Since the isotope Carbon-14 has a long half-life scientists and archeologists can use half life to figure out approximately how old an organic object is. This is known as carbon-dating.

Shroud of Turin supposedly depicting the face of Christ.

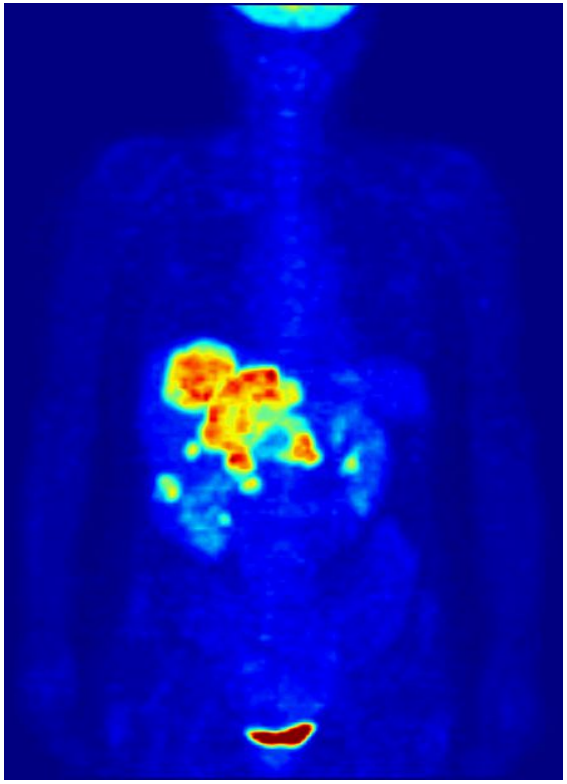
It's also used in Archeology

Isotopes

Isotope	Half-life
Thorium-232	14,000 million years
Uranium-235	704 million years
Plutonium-239	24,110 years
Carbon-14	5,730 years
Caesium-137	30 years
Cobalt-60	5.27 years
Polonium-210	138 days
Technetium-99m	6 hours
Polonium-218	3 minutes

Medical Tracers

Radioactive tracers are used to diagnose conditions in the human body. Doctors can trace the movement of these radioactive atoms as they move through the body's cells.



Radioactive tracer concentrated in the brain, kidneys and bladder.

Activity: Recap Lists

What are Lists?

A computer program often needs to store a sequence of values and then process them.

For example if you had to store the below sequence of values, how many variables would you need?

32	**54**	**67.5**	**29**	**35**	**80**	**115**	**44.5**	**100**	**65**

This is where lists become very useful, saving us time and making the process of storing a sequence of values much easier.

List definition: A List is a collection of values which is ordered and changeable.

Lists in Python

```
#Introduction to Lists

#Two ways of initializing a list

initialisingAnEmptyList = []

initialisingAListWithValues = [32,54,66,28,39,87,111]
```

Each item in a list has a corresponding **index number**, which is an integer value, starting with the index number 0. We use this index number to access an item in the list.

index	0	1	2	3	4	5	6
values	32	54	66	28	39	87	111

Accessing Lists in Python

```
#Introduction to Lists

#Two ways of initializing a list

initialisingAnEmptyList = []

initialisingAListWithValues = [32,54,66,28,39,87,111]

#Accessing values in a List

thirdItem = initialisingAListWithValues[2]

firstItem = initialisingAListWithValues[0]

#Error accessing the list

errorInAccess = initialisingAListWithValues[7]
```

Causes runtime error

Replacing Values in a List

To replace a value in the list we first identify the index number of the value to be changed and then set the value in the corresponding index number to a different value.

```
initialisingAListWithValues = [32,54,66,28,39,87,111]

#Replacing values in a List

initialisingAListWithValues[5] = 88
```

List Boundaries

You have to be careful that the index number stays within the valid range.

Attempting to access an element whose index number is not within the valid index range is called an **out-of-range error** or a bounds error which in turn causes a run-time exception.

Example:

```
values = [10,20,30,40,50,60,70,80,90]
values[9] = number #results in
#IndexError: list index out of range
```

Length of a List

We can use the **len()** function to obtain the length of the list; that is, the number of elements:

```
values = [10,20,30,40,50,60,70,80,90]

numElements = len(values)

#The value of numElements will be 9
```

Iterating a List

A **for** loop is ideal to iterate through the items in a list. We use a variable in the **for** loop that corresponds to the index number.

The **range(n)** function yields the numbers 0, 1, ... n-1, and **range(a, b)** returns a, a+1, ... b-1 up to but not including the last number (**b**). The combination of the for-loop and the range() function allows you to iterate through the list easily.

```
#Initialising a list with values
aListWithValues = [32,54,66,28,30,87,111,72,94,16]
#Loop over the index values
for i in range(len(aListWithValues)):
      print(i, aListWithValues[i])
```

Activity: Review Lists

Linear Search

A linear search is a simple search process where a list is searched sequentially until the required value is found.

For example if the required value is 4.

Linear Search

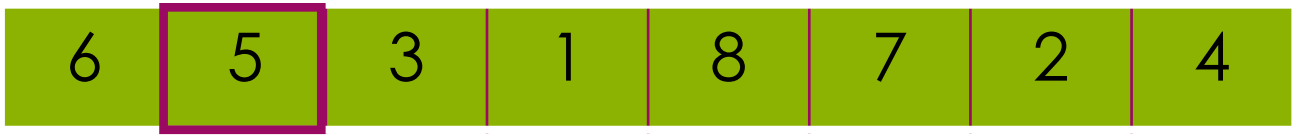
A linear search is a simple search process where a list is searched sequentially until the required value is found.

For example if the required value is 4.

Linear Search

A linear search is a simple search process where a list is searched sequentially until the required value is found.

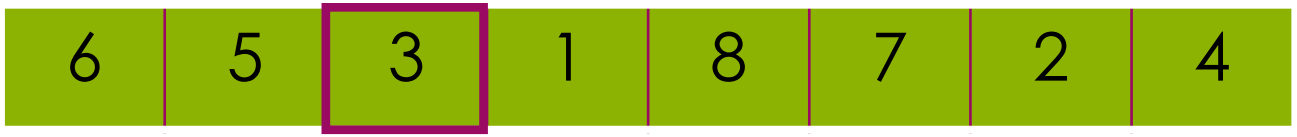
For example if the required value is 4.



Linear Search

A linear search is a simple search process where a list is searched sequentially until the required value is found.

For example if the required value is 4.



Linear Search

A linear search is a simple search process where a list is searched sequentially until the required value is found.

For example if the required value is 4.

Linear Search

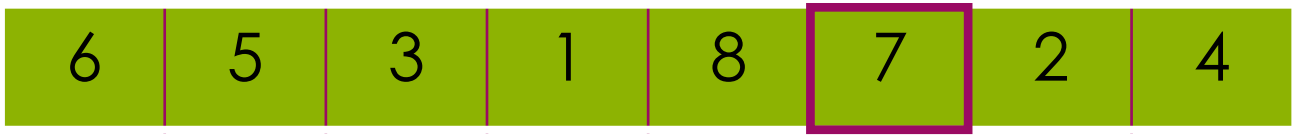
A linear search is a simple search process where a list is searched sequentially until the required value is found.

For example if the required value is 4.

Linear Search

A linear search is a simple search process where a list is searched sequentially until the required value is found.

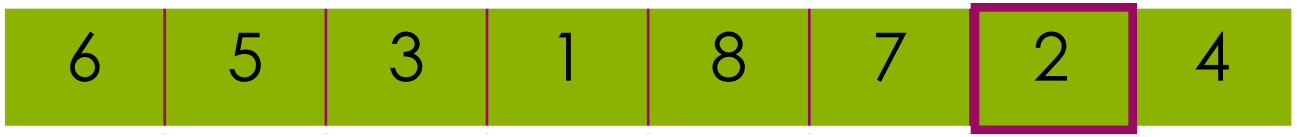
For example if the required value is 4.



Linear Search

A linear search is a simple search process where a list is searched sequentially until the required value is found.

For example if the required value is 4.



Linear Search

A linear search is a simple search process where a list is searched sequentially until the required value is found.

For example if the required value is 4.

Linear Search

A linear search is a simple search process where a list is searched sequentially until the required value is found.

For example if the required value is 4.

Binary Search

A binary search algorithm (also knowns as half-interval search) is the algorithm where:

1. The middle value in a **sorted** list is inspected to see if it matches the search value.
2. If the middle value is greater than the search value, the upper half of the list is discarded. If it is less than the search value, the lower half is discarded.
3. This process is repeated, with the list halving in size each time until the search value is found.

For example if the value we are searching for is **4**.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Binary Search

A binary search algorithm (also knowns as half-interval search) is the algorithm where:

1. The middle value in a **sorted** list is inspected to see if it matches the search value.
2. If the middle value is greater than the search value, the upper half of the list is discarded. If it is less than the search value, the lower half is discarded.
3. This process is repeated, with the list halving in size each time until the search value is found.

For example if the value we are searching for is **4**.

Binary Search

A binary search algorithm (also knowns as half-interval search) is the algorithm where:

1. The middle value in a **sorted** list is inspected to see if it matches the search value.
2. If the middle value is greater than the search value, the upper half of the list is discarded. If it is less than the search value, the lower half is discarded.
3. This process is repeated, with the list halving in size each time until the search value is found.

For example if the value we are searching for is **4**.

Binary Search

A binary search algorithm (also knowns as half-interval search) is the algorithm where:

1. The middle value in a **sorted** list is inspected to see if it matches the search value.
2. If the middle value is greater than the search value, the upper half of the list is discarded. If it is less than the search value, the lower half is discarded.
3. This process is repeated, with the list halving in size each time until the search value is found.

For example if the value we are searching for is **4**.

Binary Search

A binary search algorithm (also knowns as half-interval search) is the algorithm where:

1. The middle value in a **sorted** list is inspected to see if it matches the search value.
2. If the middle value is greater than the search value, the upper half of the list is discarded. If it is less than the search value, the lower half is discarded.
3. This process is repeated, with the list halving in size each time until the search value is found.

For example if the value we are searching for is **4**.

Binary Search

A binary search algorithm (also knowns as half-interval search) is the algorithm where:

1. The middle value in a **sorted** list is inspected to see if it matches the search value.
2. If the middle value is greater than the search value, the upper half of the list is discarded. If it is less than the search value, the lower half is discarded.
3. This process is repeated, with the list halving in size each time until the search value is found.

For example if the value we are searching for is **4**.

Activity: Binary Search in Python

Binary Search Solution – Search Function

```
# Binary Search
def binarySearch(sortedList, item):
    first = 0
    last = len(sortedList) - 1
    found = False

    while first <= last and not found:
        midpoint = round((first + last) / 2)

        if sortedList[midpoint] == item:
            found = True
        else:
            if item < sortedList[midpoint]:
                last = midpoint - 1
            else:
                first = midpoint + 1
    return found
```

Binary Search Solution – main Function

```
# Binary Search main entry

def main():

    mySortedList                                                    =
[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]

    itemToFind = 7

    print(binarySearch(mySortedList,itemToFind))



main()
```

Recap Sorting Algorithms

Bubble Sort

How to Bubble Sort:

From left to right, compare two numbers, swap if needed. Repeat until all numbers in correct order.

Bubble Sort

How to Bubble Sort:

From left to right, compare two numbers, swap if needed. Repeat until all numbers in correct order.

Bubble Sort

How to Bubble Sort:

From left to right, compare two numbers, swap if needed. Repeat until all numbers in correct order.

Bubble Sort

How to Bubble Sort:

From left to right, compare two numbers, swap if needed. Repeat until all numbers in correct order.

Bubble Sort

How to Bubble Sort:

From left to right, compare two numbers, swap if needed. Repeat until all numbers in correct order.

Bubble Sort

How to Bubble Sort:

From left to right, compare two numbers, swap if needed. Repeat until all numbers in correct order.

Bubble Sort

How to Bubble Sort:

From left to right, compare two numbers, swap if needed. Repeat until all numbers in correct order.

Bubble Sort

How to Bubble Sort:

From left to right, compare two numbers, swap if needed. Repeat until all numbers in correct order.

Bubble Sort

How to Bubble Sort:

From left to right, compare two numbers, swap if needed. Repeat until all numbers in correct order.

Bubble Sort

How to Bubble Sort:

From left to right, compare two numbers, swap if needed. Repeat until all numbers in correct order.

Bubble Sort

How to Bubble Sort:

From left to right, compare two numbers, swap if needed. Repeat until all numbers in correct order.

Bubble Sort

How to Bubble Sort:

From left to right, compare two numbers, swap if needed. Repeat until all numbers in correct order.

Bubble Sort

How to Bubble Sort:

From left to right, compare two numbers, swap if needed. Repeat until all numbers in correct order.

Bubble Sort

How to Bubble Sort:

From left to right, compare two numbers, swap if needed. Repeat until all numbers in correct order.

Bubble Sort

How to Bubble Sort:

From left to right, compare two numbers, swap if needed. Repeat until all numbers in correct order.

After every iteration/pass, each item "bubbles" up to the location where it belongs.

Bubble Sort

How to Bubble Sort:

From left to right, compare two numbers, swap if needed. Repeat until all numbers in correct order.

Repeat the process until all the numbers are in correct order.

1	2	3	4	5	6	7	8

Activity: Bubble Sort in Python

Bubble Sort Solution – Sort Function

```
# Bubble sort function
def bubbleSort(aList):
    n = len(aList)
    swapped = False
    while n > 0:
        swapped = False
        for i in range(1, n):
            if aList[i-1] > aList[i]:
                temp = aList[i]
                aList[i] = aList[i-1]
                aList[i-1] = temp
                                #aList[i],  aList[i-1]  =  aList[i-
1],aList[i]
                swapped = True
        n = n - 1
    return alist
```

Bubble Sort Solution – main Function

```
# Bubble sort main

def main():

    unorderedList = [34,23,56,89,23,43,55,75,4,2,6,10,11]

    print(bubbleSort(unorderedList))



main()
```