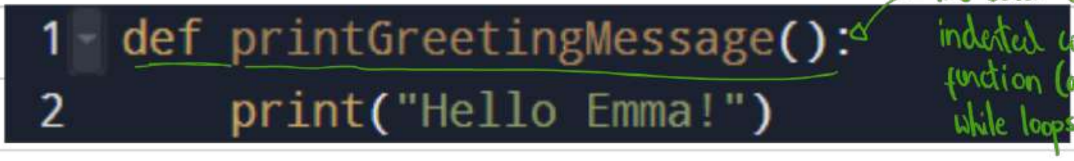
Functions

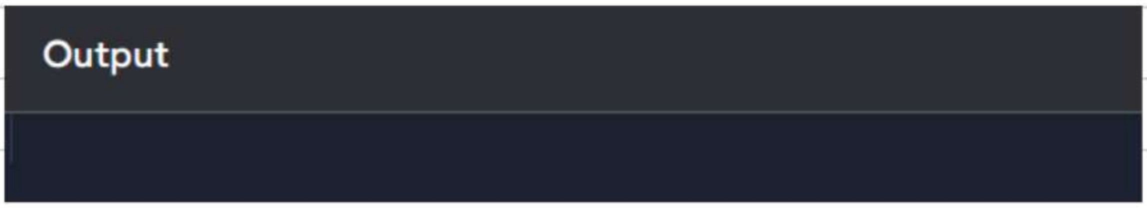
Python functions are very useful in seperating your code, making it clearer to understand and also avoiding the need to re-write code to do the same thing over and over again. Any code that you think you'll use more than once should probably be in a function. Lets see how functions work:



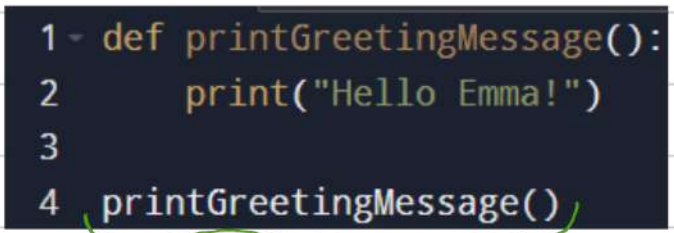
```python
def printGreetingMessage():
    print("Hello Emma!")
```

This colon states that the following indented code belongs to this function (as with if/else, for and while loops)

Functions are denoted by using def. Def is a shorthand for define, which is followed by a function name, in this case being printGreetingMessage. As with variables, the function name can be anything you like, just make sure it is consistent with the purpose of the function! This notion of functions comes from the mathematical notion of functions, which you will cross into in your maths lessons. Now, if you were to try to run the code above, you would see:

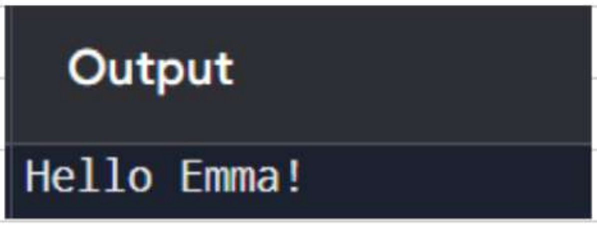


Or rather, nothing! This is because functions only run when you call them by their name, that is:



```python
def printGreetingMessage():
    print("Hello Emma!")

printGreetingMessage()
```

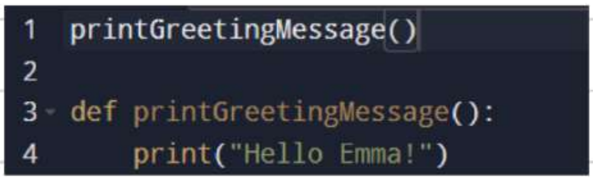
$\longrightarrow$



```
Output
Hello Emma!
```

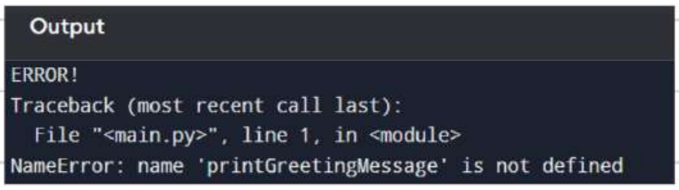
This is a function call, which tells Python - hey, I want to run the function printGreetingMessage now!

Functions have to be defined before they are called, that is something like this does not work:



```python
printGreetingMessage()

def printGreetingMessage():
    print("Hello Emma!")
```

$\longrightarrow$

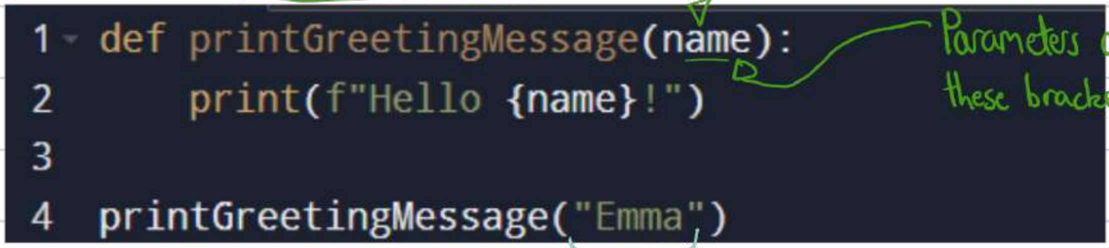


```
Output
ERROR!
Traceback (most recent call last):
  File "<main.py>", line 1, in <module>
NameError: name 'printGreetingMessage' is not defined
```

This does not work as Python reads code from top to bottom. Think of Python as reading a recipe. If the recipe says, "now add the sauce", but the sauce recipe comes later on the page, Python won't know what sauce you're talking about as it hasn't been told by that step in the recipe. Python must see the function defined before it can be used.

Functions with arguments

Functions like the one above are great, but not necessarily very useful as right now it only works for people called Emma. However, there will come a time where we will need to say hello to many people with different names - which is where the concept of function parameters and arguments come into play.

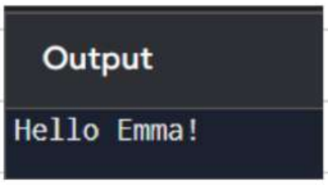


```python
def printGreetingMessage(name):
    print(f"Hello {name}!")

printGreetingMessage("Emma")
```

Parameters are always named in these brackets!

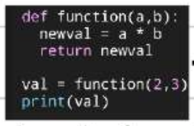
This is what we call a parameter. This is, in other words, something that this function needs to be given for it to work and complete a necessary task. This 'name' acts as a variable that we can access whilst the function is working. We will see what Python is doing behind the scenes shortly.

This is what we call an argument, the actual value that is passed to the function when called. If we run this, we get:



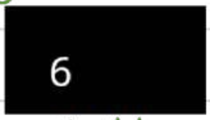
```
Output
Hello Emma!
```

Not every function we write will output things through a print statement. In fact, there will be things we don't want printed at all! In these situations, we need to use the return keyword.



```python
def function(a,b):
  newval = a * b
  return newval

val = function(2,3)
print(val)
```

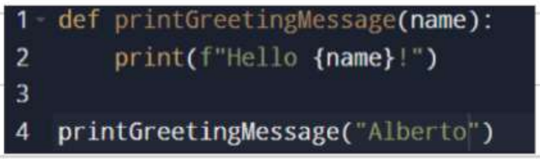
$\rightarrow$



```
6
```

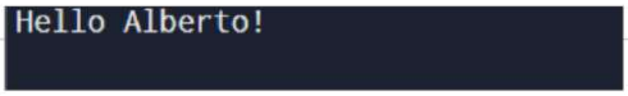
The return keyword allows us to take variables that we computed, and store them in a variable outside the function itself, this is a very useful feature that we will use very often!

If we were to change the name to something else:



```python
def printGreetingMessage(name):
    print(f"Hello {name}!")

printGreetingMessage("Alberto")
```

$\rightarrow$

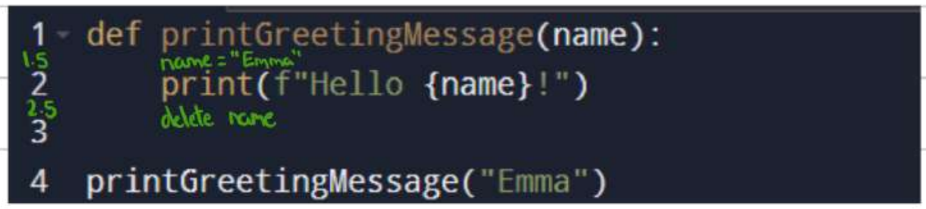


```
Hello Alberto!
```

What is happening behind the scenes?

(I didn't get to cover this in-person with you, but have a read and see if you understand!)

Behind the scenes, Python is creating and deleting a variable name (this could be any name) within the brackets and assigning it the value we gave as an argument. In other words:

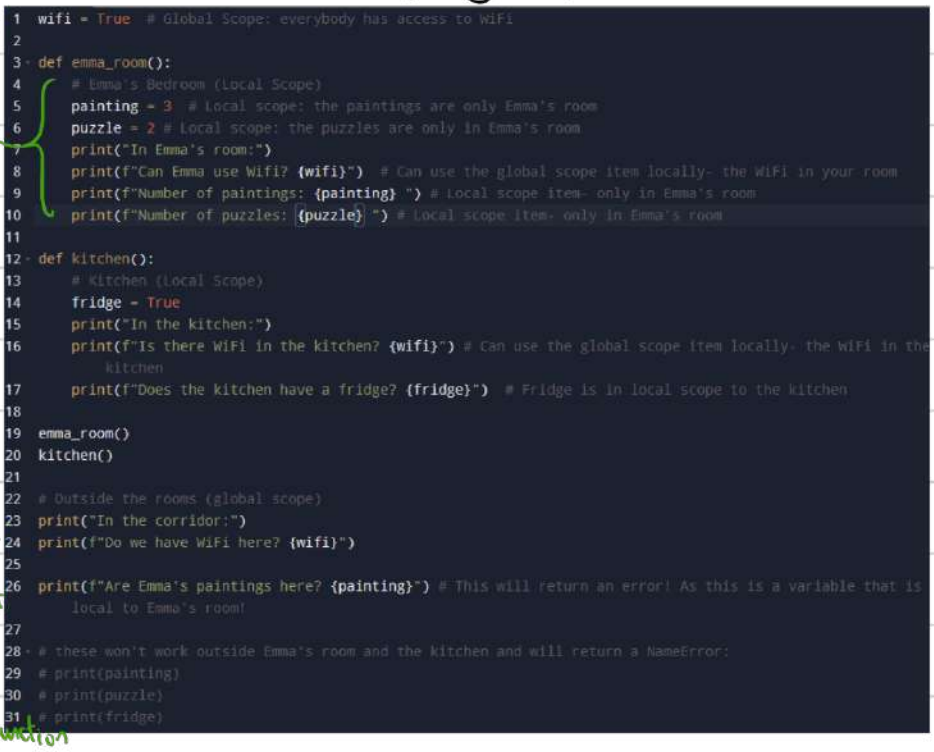


```
1   def printGreetingMessage(name):
1.5     name = "Emma"
2       print(f"Hello {name}!")
2.5     delete name
3
4   printGreetingMessage("Emma")
```

You may have wondered, why does Python always 'delete' these things like the name. This is due to the concept of scope. Scope is the part of the code where a variable exists and can be used. Think of it as rooms in a house - some things are only available in certain rooms (like a bed in a bedroom, or a fridge in a kitchen) and other things are available everywhere (like WiFi is available everywhere, the heating/radiators too). What does this look like in code? Let's see!

Scope: Global and local scope

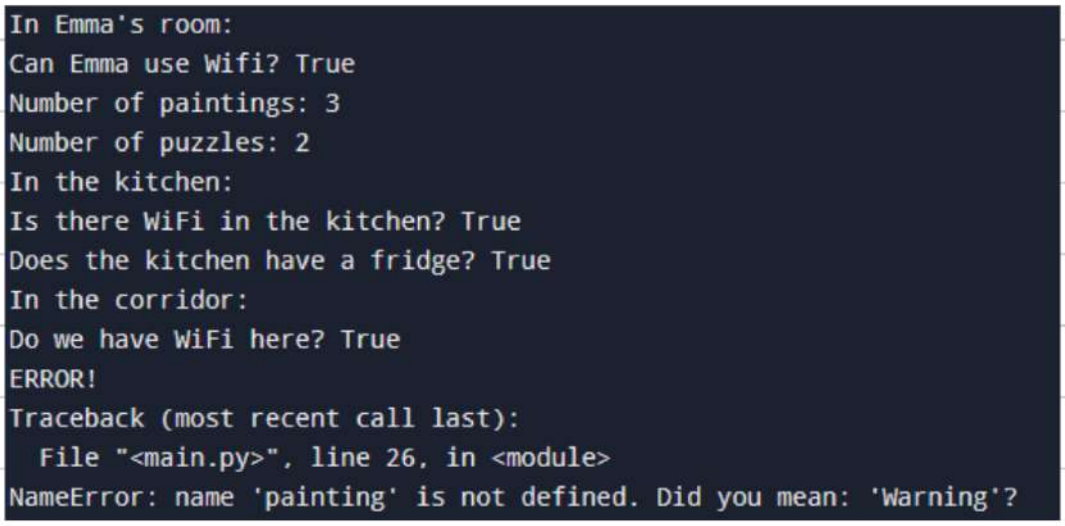
There are two types of scope - global and local scope. Global scope is variables defined outside of all functions that can be used anywhere in the file after they're defined, whilst local scope is for variables defined inside a function, that can only be used inside that function. Let's see the following example:



```
wifi = True  # Global Scope: everybody has access to WiFi

def emma_room():
    # Emma's Bedroom (Local Scope)
    painting = 3  # Local scope: the paintings are only Emma's room
    puzzle = 2 # Local scope: the puzzles are only in Emma's room
    print("In Emma's room:")
    print(f"Can Emma use Wifi? {wifi}")  # Can use the global scope item locally- the WiFi in your room
    print(f"Number of paintings: {painting} ") # Local scope item- only in Emma's room
    print(f"Number of puzzles: {puzzle} ") # Local scope item- only in Emma's room

def kitchen():
    # Kitchen (Local Scope)
    fridge = True
    print("In the kitchen:")
    print(f"Is there WiFi in the kitchen? {wifi}") # Can use the global scope item locally- the WiFi in the
        kitchen
    print(f"Does the kitchen have a fridge? {fridge}")  # Fridge is in local scope to the kitchen

emma_room()
kitchen()

# Outside the rooms (global scope)
print("In the corridor:")
print(f"Do we have WiFi here? {wifi}")

print(f"Are Emma's paintings here? {painting}") # This will return an error! As this is a variable that is
    local to Emma's room!

# these won't work outside Emma's room and the kitchen and will return a NameError:
# print(painting)
# print(puzzle)
# print(fridge)
```

Indentation matters!! It is a good way of understanding and seeing the scope of things - the variable wifi is global as there is no indentation, but painting is local to the emma_room() function

This is what happens when we run it:

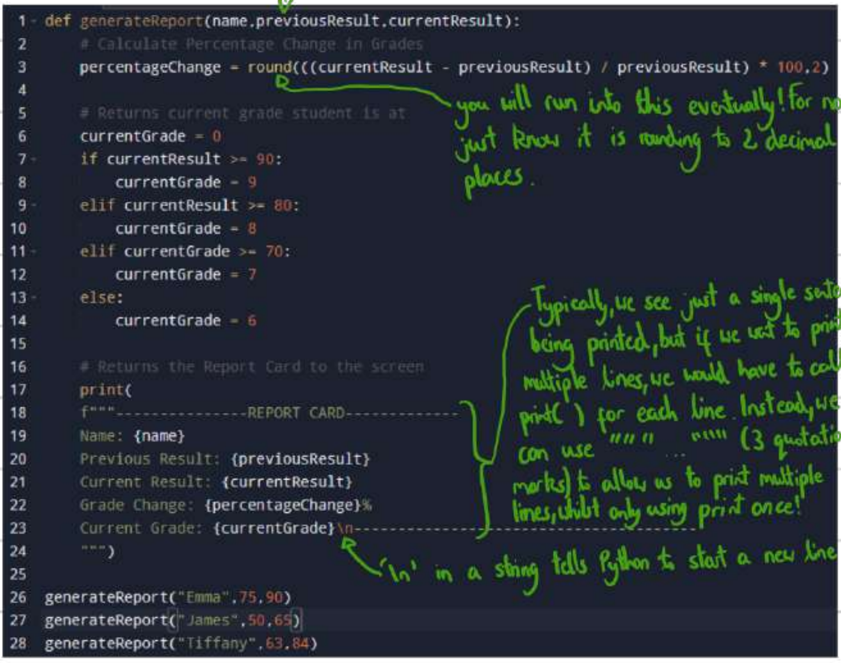


```
In Emma's room:
Can Emma use Wifi? True
Number of paintings: 3
Number of puzzles: 2
In the kitchen:
Is there WiFi in the kitchen? True
Does the kitchen have a fridge? True
In the corridor:
Do we have WiFi here? True
ERROR!
Traceback (most recent call last):
  File "<main.py>", line 26, in <module>
NameError: name 'painting' is not defined. Did you mean: 'Warning'?
```

As you can see, everything worked just fine until we got to the line of printing the number of paintings. Since the number of paintings is defined locally in the function emma_room(), we have no way of accessing this anywhere else and so Python returned us an error. This concept applies with function parameters and arguments, with the parameter being in a local scope.

The following example shows how we can use multiple parameters in one function (you can use as many as you desire):

to add more parameters, just add a comma and give it a name!



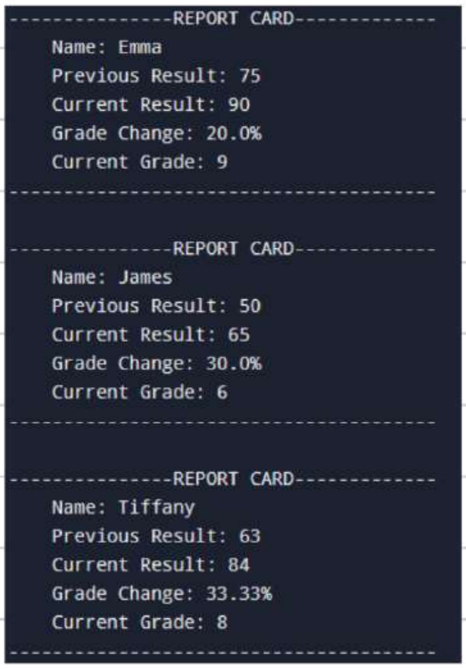
```python
def generateReport(name,previousResult,currentResult):
    # Calculate Percentage Change in Grades
    percentageChange = round(((currentResult - previousResult) / previousResult) * 100,2)

    # Returns current grade student is at
    currentGrade = 0
    if currentResult >= 90:
        currentGrade = 9
    elif currentResult >= 80:
        currentGrade = 8
    elif currentGrade >= 70:
        currentGrade = 7
    else:
        currentGrade = 6

    # Returns the Report Card to the screen
    print(
    f"""--------------REPORT CARD------------
    Name: {name}
    Previous Result: {previousResult}
    Current Result: {currentResult}
    Grade Change: {percentageChange}%
    Current Grade: {currentGrade}\n------------------------------------
    """)

generateReport("Emma",75,90)
generateReport("James",50,65)
generateReport("Tiffany",63,84)
```

you will run into this eventually! For now, just know it is rounding to 2 decimal places.

Typically, we see just a single sentence being printed, but if we want to print multiple lines, we would have to call print( ) for each line. Instead, we can use """ """ (3 quotation marks) to allow us to print multiple lines, whilst only using print once!

'\n' in a string tells Python to start a new line

$\longrightarrow$



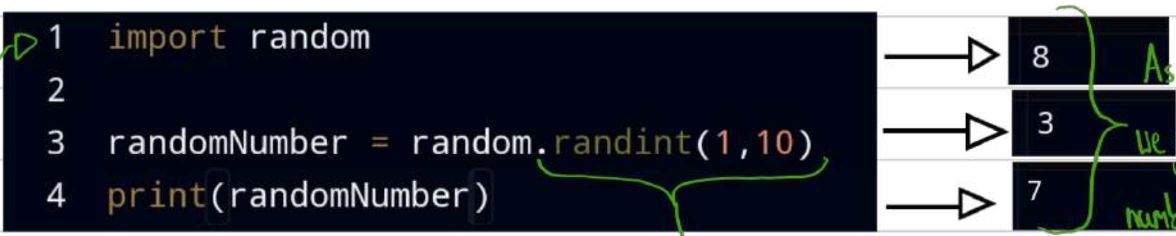
```
--------------REPORT CARD------------
    Name: Emma
    Previous Result: 75
    Current Result: 90
    Grade Change: 20.0%
    Current Grade: 9
------------------------------------

--------------REPORT CARD------------
    Name: James
    Previous Result: 50
    Current Result: 65
    Grade Change: 30.0%
    Current Grade: 6
------------------------------------

--------------REPORT CARD------------
    Name: Tiffany
    Previous Result: 63
    Current Result: 84
    Grade Change: 33.33%
    Current Grade: 8
------------------------------------
```

Libraries

Libraries in Python are a collection of pre-written code that you can reuse in your own program so that you don't have to write everything from scratch. Their purpose is to save you time from having to write already existing code and to decrease the potential number of mistakes that can occur in a program. There are libraries to do just about everything you can imagine! From using turtle to draw, to detecting animals in an image, this is all done through libraries! Using libraries is very easy, let's see how we can use them:

Example 1: Generating random 'things'

One very cool use is the random module. As the name suggests, this module is concerned with all things random! If you want to generate a random number, or generate a random string, (which can be very useful in things like password generators) using the random library is the way to do so! Lets see how it works:



```
import random

randomNumber = random.randint(1,10)
print(randomNumber)
```

```
8
```

```
3
```

```
7
```

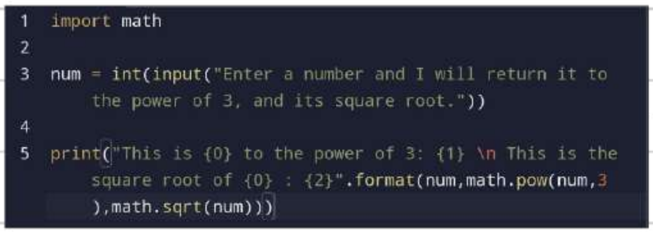
As you can see, we got a different number each time!
This use of a dot may look weird, we will look into that shortly!

In order to use a library, we import a module within it. Think of this as a library being a book, and each chapter within the book being a module. That is, many modules (chapters) make up one library (a book). We do this by writing 'import moduleName'. This essentially tells Python, "Hey! There is this specific code somebody made and I'm going to need it - so fetch it for me please!". When we want to import something, this is the FIRST thing that should be in the file.

Note: The terms 'library' and 'module' are interchangeable, especially in the context of Python. However, strictly speaking, there is a difference between the two.

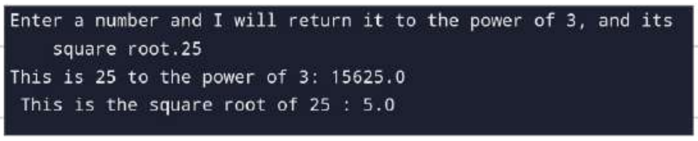
Example 2: Mathematical operations

The math module is concerned with doing all sorts of maths operations - many of which you will not have ran into yet, but will eventually. This example shows it in action, computing the cubic power of a number (that is, a number multiplied by itself 3 times) as well as the square root of a number:



```python
import math

num = int(input("Enter a number and I will return it to the power of 3, and its square root."))

print("This is {0} to the power of 3: {1} \n This is the square root of {0} : {2}".format(num,math.pow(num,3),math.sqrt(num)))
```

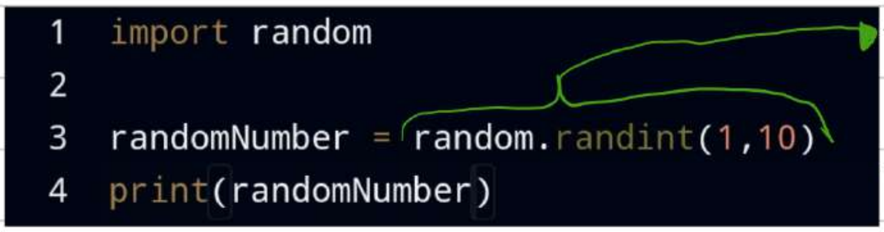
→



```
Enter a number and I will return it to the power of 3, and its
    square root.25
This is 25 to the power of 3: 15625.0
 This is the square root of 25 : 5.0
```

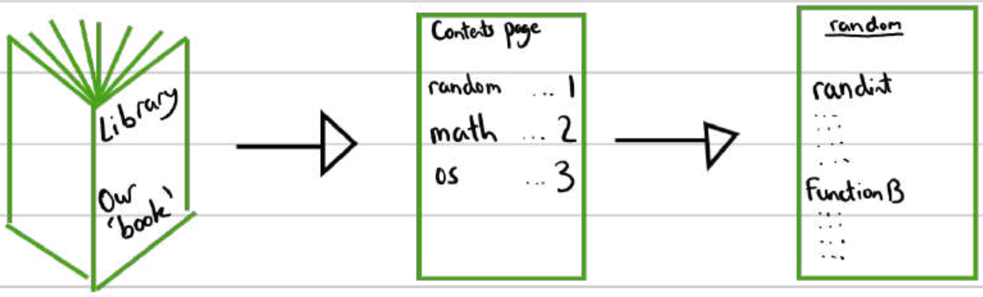
What is that weird '.pow' and '.sqrt' stuff?

To understand what this '.' stuff is, we need to better understand programming languages as a whole. There are various different 'types' of programming languages, all which require you to write your code in a different way - if you decide to progress with Computer Science onto A-Levels, you will know this better. However, for now, you need to know Python is an Object-Oriented Programming Language. What this means essentially is that everything in Python is an object, and that these objects have attributes (variables that store some data) and methods (functions). Lets use a previous example to describe this:



```python
import random

randomNumber = random.randint(1,10)
print(randomNumber)
```

This is essentially saying, from the random module, use the randint function. This diagram may help to understand:



Library: stdlib → Our chapters = modules → Heading A = randint
/Book name = stdlib
(This is not important)

Your Excercises

{ You don't need to complete/attempt all these questions by the next time we meet! I have given more questions than usual so you can really get a hang of all the concepts we've learnt!

Exc 1: When will I become 100? (Testing the use of imports, if statements, and functions)

Create a program that asks the user to enter their age. Print out a message to tell them the year that they will turn 100. If they're already 100 or more, print "incompatible age". Try to use the datetime module for this!

Exc 2: Dice Game (Testing the use of imports, if statements, and functions)

Create a function roll_dice() that returns a random number between 1 and 6. Simulate a game where the player has 3 rolls. If they roll a 6 at any time, they win.

Exc 3: Divisible by 2 (Testing the use of functions, and while loops)

Write a function div_by_2(number) that returns the number of times that a number can be divided by 2 before it is less than or equal to 10. For example:

```
div_by_2(100)=4   as   100/2 = 50    is 50≤10? No, continue
                       50/2 = 25     is 25≤10? No, continue
                       25/2 = 12.5   is 12.5≤10? No, continue
                       12.5 = 6.25   is 6.25≤10? Yes, stop!
```

Exc 4: Even Numbers (Testing the use of functions, loops, and the modulo '%' operator)

Write a function even_upto_X(value) that takes an integer number and outputs all the even numbers from 0 until that number. For example:

even_upto_X(10) will return

Challenge: Rock, Paper, Scissors! (Testing the use of functions, imports and if statements)

Make a simple rock, paper, scissors game between you and the computer. You will need to ask the user for their choice, and then randomly select either rock, paper, or scissors as the computer's choice. Output both the individual's choice and the computer's choice, followed by who won. Remember the rules:

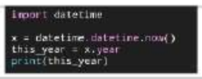
- Rock beats scissors
- Scissors beats paper
- Paper beats rock

Top tip: the internet is your best friend! Do not be afraid to search on it how to do certain things - what matters most is that you know WHAT to do, rather than the syntax on HOW to do it.

Need some help? Hints are on the next page!

Exc 1

Hint 1: The datetime module is really useful for this. The following code will let you get the year



```
import datetime

x = datetime.datetime.now()
this_year = x.year
print(this_year)
```

Hint 2: To calculate the year of the 100th birthday, we just have to subtract the age from the current year and add 100! In other words, (current-year - age)+100

Exc 2

Hint 1: You will need the random library for this. Remember random.randint()? It will be very useful here.

Exc 3

Hint 1: You need to use a while loop here. You need to apply a similar logic to the guessing game so that the while loop keeps looping. Your while condition will need to say: 'while the number is greater than or equal to 10'

Exc 4

Hint 1: A for loop of the form: for i in range(number) would be useful here. We can use the 'i' from the loop to check each number in the range.

Hint 2: The modulo operator will be very useful here!

Challenge

This one is pretty difficult, give it your best go but dont be disheartened if you dont understand. Here is a breakdown of the problem:

1) We need to ask the user for their choice, so the program should start with an input statement

2) We need to randomly choose between rock, paper or scissors for the computer. Perhaps the random library will be useful, as well as using a function to store this.
   ↳ Perhaps we can have a variable called option. We can then have another variable called random_number which stores a number between 1 and 3. If random_number ==1, we can set option = "rock", If random_number ==2, we can set option = "paper" and 3 will be option = "scissor". We have to then *return* this value, which we do 'return option'

3) Then, we need to use if statements to compare the user's input to the computer generated one to determine a winner