Day 2 - REPASO

INTRODUCTION TO PYTHON

Python is a high level programming language that is very powerful

BASICS

OUTPUT TEXT

```
print("Hello")
```

Your program can output simple text Strings by putting them in between Quotes

OUTPUT NUMBERS

```
print(1234)
```

Numbers can be typed in without anything to enclose them

VARIABLES

Naming variables then setting them to a value using = means we can use the name later to call the value that we set

You can change the value of a variable but not the **type** of data

STORE & USE TEXT

```
varHi = "Hi"
print(varHi)
```

OTHER VARIABLE TYPES

```
varChar = 'a'
varString = "Hi"
varInt = 21
varReal = 7.3
varBool = True
```

= changes the value of a variable

SELECTION

Making your program decide between different code to run is the basis of all computer programs

IF STATEMENTS

```
if(grade>20):
  print("Pass")
else:
  print("Fail")
```

This works by working out the condition in the brackets and deciding if it's True or False. If True then it runs the first line of code, if False it runs the code under the **else** construct

If you have more than one potential answer then you can use **elif** to add more conditions

ELSE IF

```
if(grade>20):
  print("Pass")
elif(grade>10):
  print("Resit")
else:
  print("Fail")
```

Always make sure to use a normal **else** at the end to catch any states you haven't considered

TABS show which line of code is run by each construct

CONDITIONS

Conditions are the different ways that you can compare variables in an IF statement

EXAMPLE CONDITIONS

```
if(value==10)
if(this<=that)
if(x>z)
if(hour!=now)
```

== compares two variables to see if they have the same value

You can use **>**, **<**, **>=** and **<=** and these work just like they do in Maths

!= means not equal to, it checks that one variable is not the same as another

ITERATION

Iteration is where you tell the program to **loop** around some code whilst a condition is True

WHILE LOOP

```
count = 0
while(count<5):
  print(count)
  count=count+1
```

The while loop keeps executing its code untilt the condition becomes true. In this case it will keep running the two lines of code as long as count is less than 5, the moment it hits 5 the exection here is complete

FOR LOOP

```
count=0
for count in range(1,10):
  print(count)
```

This loop makes it easier to iterate over a range of values without having to write code to change that value ourselves, making the likelihood of an infinite loop much smaller

NESTING

You can place any construct inside another, this is called **nesting** and allows us to do anything

```
count=0
for count in range(1,10):
  if(count%2==0):
    print("Even")
  else
    print("Odd")
```

Here we've got the same count from 1 to 10 as before only this time we have a nested **if** that uses **modulo (%)** to calculate if the number is even or odd

ARRAYS

Instead of storing each small piece of data as a seperate variable we can use a data structure, called arrays, to let us store many piece of data under a single variable name

CREATE AN ARRAY

```
names = ["Naomi", "Jack", "Jessica", "Andy"]
```

This creates a array like the one shown on the right, where the names are in order

0 is always the first element in an array last element is n-1

	NAMES
0	Naomi
1	Jack
2	Jessica
3	Andy

MANIPULATING ARRAYS

```
print(names[2])
```

Will print out the third item in the array

```
names[3]= "Andrew"
```

Will change the value of the fourth item in the array to "Andrew"

```
for count in len(names):
  print(names[count])
```

Will move through every item in the array and print it out to the console

LEN(ARRAY)

Returns the amount of items in the array