Learning Aims

- Use selection to make decisions in a program using the IF statement
- Using relational and logical operators

Key terms

Selection	Controlling the flow of execution in programs using if statements.
Condition	Used to control the flow of execution in a program. A condition contains a logical expression. An expression that results in either True or False.
Relational operators	== (equal to) != (not equal to) > (greater than) < (less than) >= (greater than or equal to) <= (less than or equal to) Used in conditions, if x > 10:
Logical operators	AND → True if **both** conditions are True OR → True if **at least one** condition is True NOT → Reverses the Boolean value (True → False, False → True) Used in logic: if (x > 10) AND (y < 5):

Selection – Using the IF statement

```
if condition:
        instructions
else:
        instructions
```

```
1 answer = input("Do you like school? (y/n) ")
2
3 if answer == "y":
4     print("You answered YES!!")
5 else:
6     print("You answered NO!!")
```

Key Words	
Decision	Deciding what to do depending on certain conditions.
IF Statement	A programming construct which enables a program to take different pathways depending on particular conditions.

Quotes or No Quotes?!

How do these two pieces of code differ?

```
1 answer = input("Enter the number 10")
2 answer = int(answer)
3
4 if answer == 10:
5     print("You answered correctly!")
6 else:
7     print("You didn't enter 10!")
```

When we **compare integers**, we **do not need quotes**.

```
1 answer = input("Do you like school? (y/n) ")
2
3 if answer == "y":
4     print("You answered YES!!")
5 else:
6     print("You answered NO!!")
```

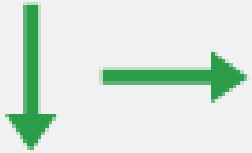
When we **compare strings**, the strings **must be inside quotes**!

A Mood Information Program!

Notice how the ELIF statements allow us to list multiple conditional statements, in a ***neat*** and ***organised*** way

```
1  number = input("How are you feeling? (1-5) ")
2
3  if number == "1":
4      print("Stay strong, things will get better!")
5  elif number == "2":
6      print("Not great today then...sorry :(")
7  elif number == "3":
8      print("Feeling OK are we?")
9  elif number == "4":
10     print("It's good that you are happy!")
11 elif number == "5":
12     print("Glad to hear you are feeling great!")
13 else:
14     print("You didn't type in a number from 1-5")
```

Flowchart symbols

Symbol	Name	Usage
	Line	Represents the flow from one component to the next
	Process	An action
	Input/ Output	An input or output
	Decision	A yes/no/true/false decision
	Terminal	The start or end of the process

Complex Boolean expression

A complex Boolean expression contains one or more of the operators and, or or not.

```
if x <= 10 or currentCharNum > lengthOfString:
```

When to use AND vs OR

Outside a range:

```
if num < 0 or  > 100:
    print ("invalid")
```

Inside a range:

```
if num > 0 and < 100:
    print ("Valid")
```

Complex Boolean expression

Consider an estate agent's program that searches through a file of house details to find ones that match a customer's requirements.

In this case the customer wants a house or flat, but it must have more than three bedrooms:

```
if rooms > 3 and (type == "House" or type == "Flat"):
    # output details
```

Notice the set of brackets around the second half of the expression.

AND takes precedence over OR so without the extra brackets the program would return all the houses with more than three bedrooms as well as any flats, whether they have more than three bedrooms or not.