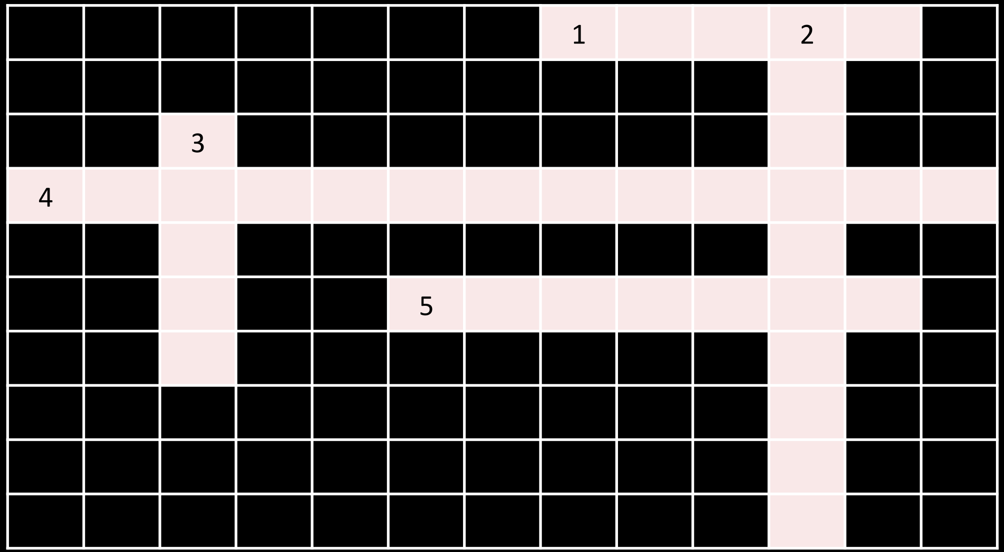
INTRODUCTION TO PROGRAMMING IN PYTHON

Selection

REVIEW - TERMINOLOGY

1. Data type for a number that includes decimal places
2. The name given to setting the value of a variable
3. The command used to get data from the user
4. The name given to joining two strings together
5. The name give to converting a variable from one data type to another



REVIEW - SOLUTION

1. Data type for a number that includes decimal places
2. The name given to setting the value of a variable
3. The command used to get data from the user
4. The name given to joining two strings together
5. The name give to converting a variable from one data type to another

							F	L	O	A	T	
										S		
		I								S		
C	O	N	C	A	T	E	N	A	T	I	O	N
		P								G		
		U			C	A	S	T	I	N	G	
		T								M		
										E		
										N		
										T		

SELECTION

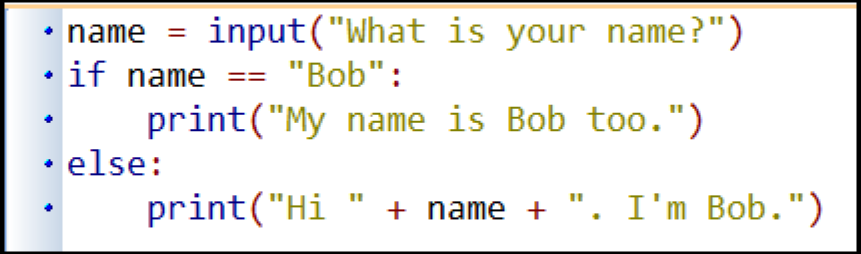
- So far the short programs we have written have run one line after the next. In programming when code is executed like this it is said to be executed in **sequence** or sequentially.
- **Selection** is the term given to code where different code may or may not be executed depending on whether a condition is met.
- This allows the user to perform different actions based on the input of the user.

IF STATEMENTS

- The key tool we use in Python to perform selection is the IF statement.
- The format for an if statement is:
  - IF condition is true
    - THEN do this
  - ELSE
    - Do this
- An example might be:
  - IF name = “Bob”
    - PRINT(“Bob is my name too.”)
  - ELSE
    - PRINT(“Hi “ + name + “. I’m Bob”)

IF SYNTAX

- The syntax for the if statement in Python is shown in the example below:



```
name = input("What is your name?")
if name == "Bob":
    print("My name is Bob too.")
else:
    print("Hi " + name + ". I'm Bob.")
```

- Some of the key things to notice:
  - If and else are written in lower case
  - To check if something is equal to a value we use double equals
  - The condition to decide what code to run is whether the name they enter is "Bob"
  - The code we want to be executed if the condition is true is indented. We could include more than one line of code here.
  - There is no condition next to else because it executes if the condition is false.
  - The code to be executed if the condition is false is also indented.

COMPARISON OPERATORS

- As seen in the example we can check if something is equal to something else using ==.
- To check if something is not equal to something we use !=.
- For integers and floats <, >, <= and >= are also useful. If we use these operators to compare strings it will compare them alphabetically.
- Note that =< and => don’t work as the order matters.

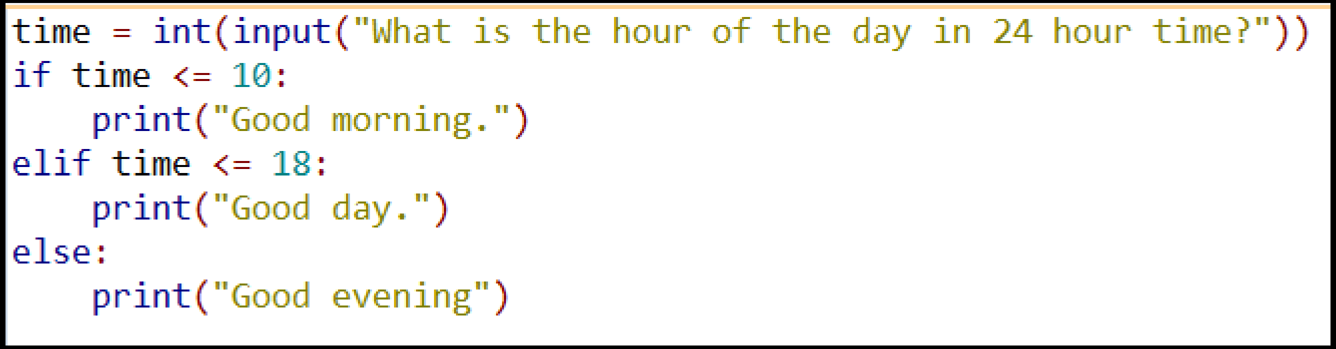
IF SYNTAX EXERCISE

- Open the file if_syntax_task.py and fix the syntax.
- Remember what to look for:
  - If and else are written in lower case
  - Acceptable comparison operator (i.e. one of: ==, !=,<,>,<=,>=)
  - Lines starting with if or else end with a colon (:)
  - All code to be executed if true is indented
  - All code to be executed in the case of else is indented

- If you are finished early open a blank Python program and create a program that asks the user what their favourite sport or tv show is and give a different response depending whether their favourite is the same as yours.

ELSE IF

- Sometimes you may want to check more than one condition and respond differently for each. This is where an if else is useful as in the example below:



```python
time = int(input("What is the hour of the day in 24 hour time?"))
if time <= 10:
    print("Good morning.")
elif time <= 18:
    print("Good day.")
else:
    print("Good evening")
```

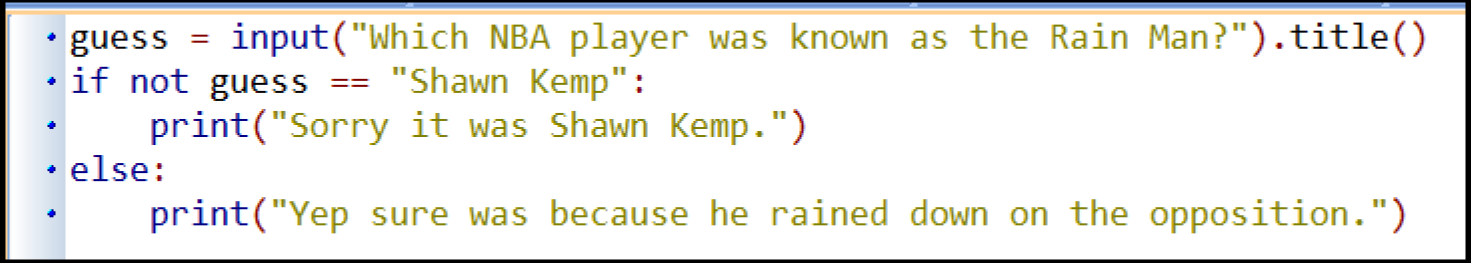
- Some points to be aware of:
  - The conditions are checked in order and as soon as one is true it is executed and the statement finished

ELSE IF SYNTAX EXERCISE

- Open the else_if_syntax_task.py file and correct the syntax.
- Remember what to look for:
  - If, elif and else are written in lower case
  - Acceptable comparison operators (i.e. one of: ==, !=,<,>,<=,>=)
  - Lines starting with if, else if or else end with a colon (:)
  - All code to be executed if true is indented
  - All code to be executed else if true is indented
  - All code to be executed in the case of else is indented
  - If and elif lines have a condition else don’t
- If finished early adapt your earlier code or create new code so you have a question that asks the user what their favourite sport or tv show is and provides several different answers based on what the user enters.

BOOLEAN OPERATORS

- The Boolean operators AND, OR and NOT can be used to create more complex conditions.
- The NOT operator is pretty self explanatory. In the example below it checks if the answer is anything other than the one given.



```
guess = input("Which NBA player was known as the Rain Man?").title()
if not guess == "Shawn Kemp":
    print("Sorry it was Shawn Kemp.")
else:
    print("Yep sure was because he rained down on the opposition.")
```

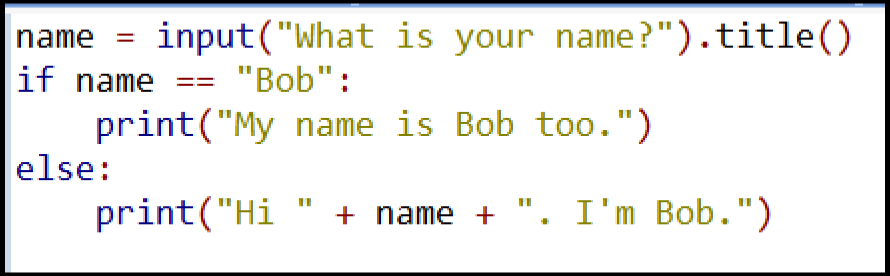
BOOLEAN OPERATORS

- The OR operator is used when more than one condition leads to the same outcome.

- The NOT operator is pretty self explanatory. In the example below it checks if the answer is anything other than the one given.

ANOTHER USEFUL STRING TRICK

- You can change the case of a python string with a couple of useful commands:
  - By adding .upper(), .lower(), or .title() to the end of your input command you will force the data to be stored as all capitals, all lowercase or with the first letter of each word capitalised respectively.
  - This means that when you are asking the user for input and don’t know if they will use capitals or not you can force it to the way you want so you only need to check one response like in the example below.



```
name = input("What is your name?").title()
if name == "Bob":
    print("My name is Bob too.")
else:
    print("Hi " + name + ". I'm Bob.")
```