Session 2: If statements

See here for last week's content.

Clarifications from last session

In the last session we introduced two ways of printing strings and variables using `+` and `,` respectively

```
average = 3.0
print("The average is "+str(average)) # The average is 3.0
print("The average is", average) # The average is 3.0
```

Both works perfectly fine but to prevent confusion, we would use `+` from now on.

If statements

We often want our programs to adapt their behaviour based on the input they receive. For instance, lets say we want to create a program, which tells our user if the number they input is small (e.g. less than 10). This is what the next program does:

```
number = int(input("Please enter a number: "))
if number < 10:
  print(number + " is a small number.")
```

We could also have our program say something if the number is large as well:

```
number = int(input("Please enter a number: "))
if number < 10:
  print(number + " is a small number.")
else:
  print(number + " is a large number.")
```

We can read the above as

> if "this condition is true" then "execute this code", otherwise "execute this other code"

We can generalise the first example as:

```
if condition_is_true:
  execute_this_code
```

And the second one as:

```
if condition_is_true:
  execute_this_code
else:
  this_code_is_executed_otherwise
```

Notice that we write "then" as `:`. The most important thing to notice is that we execute the indented code *only* when the condition holds.

We can write any number of statements inside the **body** of the `if` statement, i.e. statements that are indented (press `TAB`) will only be executed if the condition of the `if` statement holds.

There are several important concepts to note here:

- If we want to compare two values (e.g. the value the user entered and a fixed string) we use `==` instead of `=` (remember, `=` means *becomes*)
- We can compare numbers and strings using the comparison operators `==`, `!=` (not equal), `<=` (less than or equal), `>=` (greater than or equal), `<` (less than), and `>` (greater than)
- `if` statements can be nested

`elif`

We can also abbreviate the following:

```
number = int(input("Please enter a number: "))
if number < 10:
  print("This number is small.")
else:
  if number >= 10 and number < 100:
    print("This number is not small, but not large either.")
  else:
    print("This number is large.")
```

to

```
number = int(input("Please enter a number: "))
if number < 10:
  print("This number is small.")
elif number >= 10 and number < 100:
  print("This number is not small, but not large either.")
else:
  print("This number is large.")
```

You can have as many `elif` branches as you like, but only one `if` branch and one `else` branch, which must appear at the beginning and end respectively.

`and` `or` `not`

Here we used the keyword `and` to mean that we want both `number >= 10` and `number < 100` to be true for that branch to be executed. You could use `or` and `not` in expressions, e.g.

```
number = int(input("Please enter a number: "))
if not number < 10:
  # Will be executed if number is not less than 10 (i.e. if number >= 10)
  print("The number is not less than 10.")

if number == 3 or number == 4:
  # Will be executed if number is equal to 3 or equal to 4
  # Note that comparison is done with == rather than =
  print("The number is either 3 or 4.")
```

If you are using multiple different connectives, you should include parentheses to make it clear how the expression should be interpreted. E.g.

```
number = int(input("Please enter a number: "))
if ((1 <= number and number < 10) or number == 42) and (not number == 3):
  print("This was the number: " + str(number))
```

This code will only print `number` if it is one of the numbers [0, 1, 2, 4, 5, 6, 7, 8, 9, 42].

Note: there is an order of operations on logical connectives as well (which you may wish to google); however, it is recommended that you insert parentheses if you are using multiple different connectives, like in the example above.

It's important to remember that an `else` statement (or an `elif`) statement is associated with the closest `if` statement above it, which has the same indentation (same number of `TAB`s before it). So consider the program:

```
name = input("Please enter a name: ")
if name == "Mark":
  print("Oh, hi Mark.")
if name == "Johnny":
  print("Oh, hey Johnny, what's up?")
else:
  print("Who is this?")
```

You might think that if the name input is 'Mark' it will print 'Oh, hi Mark.' if it's Johnny, it will print 'Oh, hey Johnny, what's up?', and otherwise it will print 'Who is this?'. But actually, if the name is Mark, it will print both 'Oh, hi Mark' and 'Who is this?'. This is because the `else` statement is only associated with the `if` just above it. The first `if` is completely independent. You might find it helpful to think about the program split as so:

```
name = input("Please enter a name: ")
```

```
```

```
if name == "Mark":
  print("Oh, hi Mark.")
```

```
if name == "Johnny":
  print("Oh, hey Johnny, what's up?")
else:
  print("Who is this?")
```

The correct way to write the above program is using an `elif`:

```
name = input("Please enter a name: ")
if name == "Mark":
  print("Oh, hi Mark.")
elif name == "Johnny":
  print("Oh, hey Johnny, what's up?")
else:
  print("Who is this?")
```

Exercise

Easy

1. **One detector:** create a program, which takes an input (an int) and outputs "This number is 1" if the input was 1 and "This input was not 1" otherwise.

2. **Subtraction:** create a program, which takes two integers, and outputs the result of the larger number subtracting the smaller number.

```
first = ________(input("Input first number: "))
second = ________(input("Input second number: "))
if ________ < ________:
    #swap the two numbers
answer = first - second
print("The answer of "+str(first)+"-"+str(second)+" is " + str(answer))
```

```
Samples:
Input first number: 4
Input second number: 3
The answer of 4-3 is 1

Input first number: 3
Input second number: 4
The answer of 4-3 is 1
```

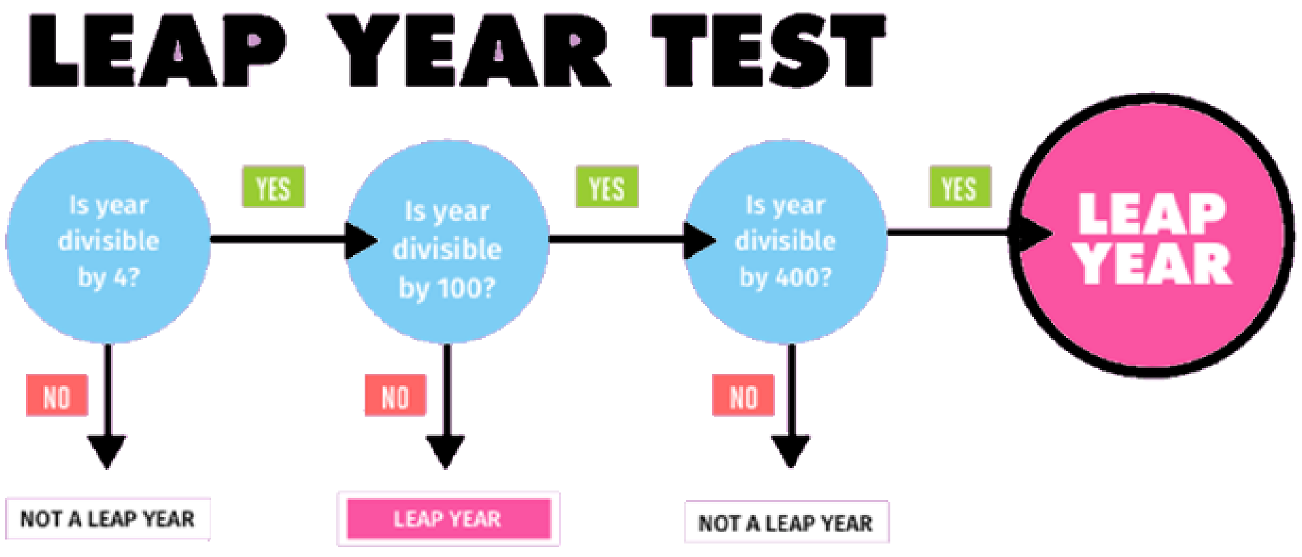
Medium

3. **Award**: Students can only get an award if they get a score of 60 or more and get an "A" for conduct. Create a program, which takes one integer (a score) and one string (a conduct grade), and tell whether the student could get an award.

```
Samples:
Enter score: 88
Enter conduct grade: A
Congratulations! You get an award.

Enter score: 88
Enter conduct grade: B
Your conduct grade is not A.

Enter score: 40
Enter conduct grade: A
Your score is not high enough.

Enter score: 40
Enter conduct grade: C
Your score is not high enough and your conduct grade is not A.
```

Hard

4. **Leap Year:** a. create a program, which determines whether a year is a leap year.



(*You will need to use the operator %. a % b gives the remainder when a is divided by b*)

```
Samples:
Enter year: 2022
2022 is not a leap year

Enter year: 2020
2020 is a leap year

Enter year: 2000
2000 is a leap year

Enter year: 1900
1900 is not a leap year
```

b. Try to achieve the same result by only using one if else block.