CH40208: TOPICS IN COMPUTATIONAL CHEMISTRY

PYTHONIC LOGIC AND LOOPS

LOGICAL OPERATORS

- Python and Jupiter Notebook can be used as a simple calculator
- Let's make or code more intelligent!
- To do this we can use *Boolean logic*; `True` or `False` questions
- Python is able to assess the truth of a particular operation

LOGICAL OPERATORS

Some logical operators

Name	Equals	Less than	Less than or equal	Greater than	Greater than or equal	Not equal
Operator	==	<	<=	>	>=	!=

LOGICAL OPERATORS

DEMO

FLOW CONTROL

- We are then able to use this Boolean logic to *control* the path that the code will follow
- To do this we use `if` statements; these ask `if x is True`?
  - Note the the `is True` part is often implicit
- The `if` statement is often accompanied by an `else`; which is the path taken when `x is False`
- The third modifier in an `if` statement is the `elif` (short for else if); this offers an alternate path to follow

FLOW CONTROL

DEMO

MORE LOGICAL OPERATORS

- Logical operators can be extended to include those which link two statements
- These are the AND and OR operators; which are foundational to computational logic

The results of an AND operation

Input A	Input B	Logic	Output
True	False	AND	False
True	True	AND	True
False	False	AND	False

The results of an OR operation

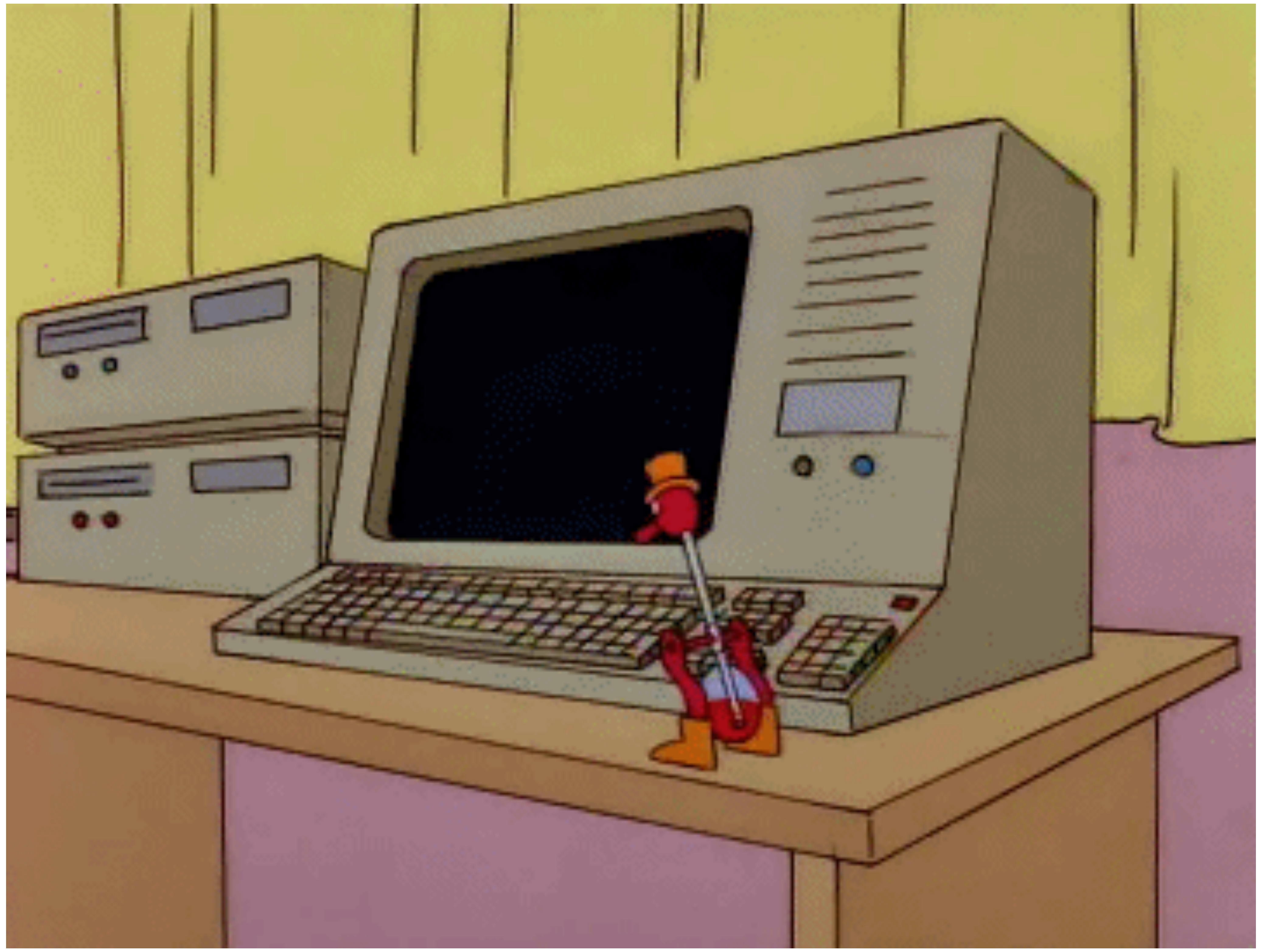
Input A	Input B	Logic	Output
True	False	OR	True
True	True	OR	True
False	False	OR	False

MORE LOGICAL OPERATORS

DEMO

LOOPS

- One of the most powerful tools of a computer is to perform repetitive tasks



LOOPS

- Loops allow us to ask the computer to perform the same (or a very similar) task multiple times over
- Python has two types of loop
  - `for` loops iterate over a given list
  - `while` loops repeat as long as a logical operation is True
- Generally it is safer to use a for loop than a while loop; with the while loop it is more easy to cause an infinite loop
- It may be desirable to escape from a loop, or to skip to the next iteration; for this there are the `break` and `continue` commands

LOOPS

DEMO

PROBLEMS

- There are two problems to tackle this week, which can be found on the handout
- **Remember** to first determine the algorithm that you will use (ideally write it down)
- Only once you have an algorithm in mind (or on paper), should you start to code