New data type: `bool`

Week 3

Boolean values

- **Boolean values** are a fundamental concept in computer science and mathematics. They represent one of two possible states: **true** or **false**.
- The statement "1 is greater than 2" is **false**
- The statement "3 is less than 5" is **true**
- **Boolean values** are essential in programming for making decisions and controlling the flow of execution.
- Boolean values are named after **George Boole** (1815-1864), whose ideas are used in modern digital computers.

New data type: Boolean (`bool`)

- In Python, Boolean values are defined by keywords `True` and `False`
- They belong to the `bool` datatype:

```
>>> a = True
>>> type(a)
<class 'bool'>
>>> b = False
>>> type(b)
<class 'bool'>
```

- `True` and `False` are really Boolean values
- **They are not** strings: `"True"` and `"False"`

Comparison operators

- **Comparison operators** are used to compare values in Python. They **return** a Boolean value based on the result of the comparison.

Operator	Interpretation
<	Smaller than
<=	Smaller than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to
!=	Not equal to

Comparison

- These operators can be applied to `int`, `float`, `str` objects. However, you cannot mix numbers and strings.
- Comparison of ***numbers*** is done "normally"
- **Strings** are compared by comparing its characters one-by-one from the beginning.
- When different characters are found, their Unicode values are compared and the character with smaller Unicode value is smaller
- For example, "apple" < "ananas" is `False`.
  - The first characters are "a" and "a", which are equal.
  - The second characters are "p" and "n". The Unicode value of "p" (112) is greater than the Unicode value of "n" (110)

Unicode values

- **Unicode** values are unique numbers assigned to each character in the Unicode standard.
- You can get the Unicode value of a character by using `ord()`:

```
>>> ord("A")
65
>>> ord("ä")
228
>>> ord("傳")
20659
```

Files: `Example1.py` (numbers); `Example2.py` (strings)

Keyword: `in`

- Python has the keyword `in` which can be used for testing whether a value is present in a sequence. It returns:
  - `True` if the value is found in a sequence
  - `False` if the value is not found if a sequence.

- A keyword with similar functionality appears in many other programming languages, though the exact syntax and usage can vary.

Examples

```
>>> "a" in "car"
True
>>> "b" in "car"
False
```

```
>>> "st" in "string"
True
```

- We will consider "`in`" also later.