CERTIFIED ASSOCIATE IN PYTHON PROGRAMMING

BY: IMRAN

Comparing strings

- Python's strings **can be compared using the same set of operators** which are in use in relation to numbers.
- Take a look at these operators - they can all compare strings, too:
  - ==
  - !=
  - >
  - >=
  - <
  - <=
- There is one "but" - the results of such comparisons may sometimes be a bit surprising. Don't forget that Python is not aware (it cannot be in any way) of subtle linguistic issues - it just **compares code point values**, character by character.

Comparing strings

- Two strings are equal when they consist of the same characters in the same order. By the same fashion, two strings are not equal when they don't consist of the same characters in the same order.

- Both comparisons give True as a result:
- 'alpha' == 'alpha'
- 'alpha' != 'Alpha'


Comparing strings

- When you compare two strings of different lengths and the shorter one is identical to the longer one's beginning, the **longer string is considered greater**.
- 'alpha' < 'alphabet'
- String comparison is always case-sensitive (**upper-case letters are taken as lesser than lower-case**).
- 'beta' > 'Beta'

Comparing strings

- Even **if a string contains digits only, it's still not a number**. It's interpreted as-is, like any other regular string, and its (potential) numerical aspect is not taken into consideration in any way.

- '10' == '010'
- '10' > '010'
- '10' > '8'
- '20' < '8'
- '20' < '80'


Comparing strings

- **Comparing strings against numbers is generally a bad idea.**
- The only comparisons you can perform with impunity are these symbolized by the == and != operators. The former always gives False, while the latter always produces True.
- '10' == 10
- '10' != 10
- '10' == 1
- '10' != 1
- '10' > 10


Sorting

- Comparing is closely related to sorting (or rather, sorting is in fact a very sophisticated case of comparing).
- This is a good opportunity to show you two possible ways to **sort lists containing strings**. Such an operation is very common in the real world - any time you see a list of names, goods, titles, or cities, you expect them to be sorted.
- In general, Python offers two different ways to sort lists.
- The first is implemented as **a function named sorted()**.
- The function takes one argument (a list) and **returns a new list**, filled with the sorted argument's elements. (Note: this description is a bit simplified compared to the actual implementation - we'll discuss it later.)
- The original list remains untouched.

Sorting

- Comparing is closely related to sorting (or rather, sorting is in fact a very sophisticated case of comparing).
- This is a good opportunity to show you two possible ways to **sort lists containing strings**. Such an operation is very common in the real world - any time you see a list of names, goods, titles, or cities, you expect them to be sorted.
- In general, Python offers two different ways to sort lists.
- The first is implemented as **a function named sorted()**.
- The function takes one argument (a list) and **returns a new list**, filled with the sorted argument's elements. (Note: this description is a bit simplified compared to the actual implementation - we'll discuss it later.)
- The original list remains untouched.

Sorting

```
# Demonstrating the sorted() function:

first_greek = ['omega', 'alpha', 'pi', 'gamma']
first_greek_2 = sorted(first_greek)
print(first_greek)
print(first_greek_2)
print()


# Demonstrating the sort() method:
second_greek = ['omega', 'alpha', 'pi', 'gamma']
print(second_greek)
second_greek.sort()
print(second_greek)
```


Strings vs. numbers

- There are two additional issues that should be discussed here: **how to convert a number (an integer or a float) into a string, and vice versa**. It may be necessary to perform such a transformation. Moreover, it's a routine way to process input/output data.
- The number-string conversion is simple, as it is always possible. It's done by a function named str().
- itg = 13
- flt = 1.3
- si = str(itg)
- sf = str(flt)

- print(si + ' ' + sf)

Strings vs. numbers

- The reverse transformation (string-number) is possible when and only when the string represents a valid number. If the condition is not met, expect a `ValueError` exception.
- Use the int() function if you want to get an integer, and float() if you need a floating-point value.
- si = '13'
- sf = '1.3'
- itg = int(si)
- flt = float(sf)

- print(itg + flt)