**Chapter 8 – strings and lists**

**8.1**

A sequence is an object that holds multiple items of data, stored one after the other. Its main types in python are: **strings, lists, tuples.**

Sequences can be mutable(lists) or immutable (strings,tuples).

**8.2**

A string is a sequence of characters, it can contain alphanumeric characters and symbols. Strings must be enclosed with single or double quotes.

**8.3**

Main operators on strings:

+ operator:

* Concatenates 2 strings of the same data type
* To concatenate different types of data, use a converter.

\* the repetition operator:

* Repeats a string a certain number of times

In operator:

* returns True if a string is contained in another string, False otherwise
* X = “rome is in the Italian capital.”
* “capital “ in x 🡪 true

Not in:

* opposite of in

Is/is not operators:

* returns True if two strings are identical, False otherwise
* is not 🡪 opposite of is

**8.4 - Lists**

A **list** is a collection of different data, each one called an element. The elements of a list are enclosed by square brackets and divided by commas.

The data types in a list do not have to be uniform

The list function converts certain object types, such as a text string, or an iterable object, into a list.

* Text = list(“milan”)
* Print(text) 🡪 [‘m’,’I’,’l’,’a’,’n’]
* Can also do this with range functions to create a list of numbers.

**8.5**

List operations:

* The + operator concatenates two lists respecting the order of insertion.
* Unlike strings, lists can be created with different data types.
* The \* operator: cities = [‘Florence’,Florence’]
  + X = cities \* 3
  + Print(x) 🡪 [‘florence’,’florence’,’florence’,]…
* in 🡪 returns True if an (entire) element is found within a list, False otherwise
* not in 🡪 opposite of in

**8.6 – Indexing**

Indexing is a way to access single element of sequence.

Graphical user interface, table, calendar

Description automatically generated

The position of each element within a sequence is identified by an integer, called index.

To access an item within a sequence the syntax is:

Nameofsequence[index]

If the sequence is mutable the same syntax can be used to modify the element of the list:

Nameofsequence [index] = newvalue

* + E.g.
  + Where capitals = [‘rome’,’london’,’berlin’]
  + Capitals[2] = ‘madrid’
  + Print(capitals)🡪 now instead of berlin you will get madrid.

While loops to show on screen all elements of a list:

Capitals = [‘rome’,’london’,’pasis’]

Index = 0

While index < 3:

Print(capitals[index])

Index = index + 1

* Print(capitals[-4]) 🡪 ‘London’
* While loops negative order:

Capitals = [‘rome’,’london’,’pasis’]

Index = -1

While index >= -3

Print(capitals[index])

Index = index – 1

Output: paris, london, rome

* To show individual characters:
* Text = ‘Python basics’
* Text[10]
* ‘I’
* Strings are immutable: can’t change individual characters of the string stored in text variable.

**8.7 - Slicing**

To select multiple elements of a sequence at the same time Python allows us to use a technique called slicing, very similar to indexing. It allows us to select a portion of the sequence (called slice) using the indexes of the elements of the sequence.

To select a slice of the sequence use: **Sequence[start:end:step]**

A picture containing table

Description automatically generated

To select and store in a variable the first six letters of the string 'PYTHON BASICS', both positive and negative indices can be used:

text = 'PYTHON BASICS'

x = text[0:6]

y = text[-13:-7]

text[2:8] Selects all the elements with index from 2 (included) to 7(element 8 excluded) and returns ‘THON B’

**Nameofsequence [startindex : endindex : step]**

The start index is included, the end index is excluded.

If start is omitted, the slice starts from index=0, if end is omitted the slice ends at the last element if step is omitted it is equal to 1.

If one parameter is negative, then they must all be or, if step is negative, start and end can be positive but they must be inverted.

To have a negative step parameter it is necessary that also the start and end parameters are negative: in this case the selection of the elements takes place from right to left, and also the step parameter, whose meaning doesn’t change, must adapt.

Alternatively, it is possible to use a negative value of the step parameter with positive start and end parameters, but in this case they must be inverted, starting from end.

**Slicing works with lists:**

* E.g.: Consider for example the following list, to which we applied a slicing with step2:
* Capitals = ['Rome', 'London', 'Berlin', 'Tokyo', 'Moscow', 'Lisbon', 'Paris']
* Capitals[0:5:2]
* Result: ['Rome', 'Berlin', 'Moscow'].
* **Replacing elements simultaneously:**
* We want to replace the first two elements of the Capitals list with 'Oslo' and 'Dublin'. With indexing, it is possible to replace the first and the second element in two steps:
* With indexing we can do:

Capitals[0] = 'Oslo'

Capitals[1] = 'Dublin'

>>>Capitals

['Oslo', 'Dublin', 'Berlin', 'Tokyo',

'Moscow', 'Lisbon', 'Paris']

* With slicing:

Capitals[0:2] = 'Oslo', 'Dublin'

Capitals[0:2] = ['Oslo', 'Dublin']

>>>capitals

['Oslo', 'Dublin', 'Berlin', 'Tokyo',

'Moscow', 'Lisbon', 'Paris']

**8.8**

**Functions and methods of strings:**

Methods are functions that belong to a certain **object**, their syntax is different than that of functions because the name of the object must always be specified.

* General syntax of a function: **function\_name(x)** where x stores a string.
* Syntax of methods: **Nameofvariable.nameofmethod(arguments)**

Main built-in **functions** for strings:

Len(): returns the length of the string, the no. of characters in the string

Max(): returns the largest value in the sequence. I strings, the maximum value is decided by

alphabetical order: a lowest, z highest

Min(): opposite of max

The main string **methods** are:

.upper(): transforms string in uppercase

.lower(): transforms string in lowercase

.capitalise(): returns string with first letter in uppercase (rest lowercase)

.strip(): returns string with all whitespace characters removed

.find(sub): finds the subrstring of the argument in the string and returns the first index.

If the substring is not found, the method returns -1

.replace(old,new): transforms the string with substring old replaced by substring new

.startswith(prefix): returns true if the string starts with the string specified in the prefix. Start and end are optional torestrict the search to a substring of the main string.

.endsith(suffix): same as startswith but this time but ends with suffix

.count(sub): returns the number of occurrences of the substring sub in the string

.split(iterable): separates a string in words. The optional argument separator can be used instead of the space. The optional argument maxsplit allows to set a maximum number of splits.

.join(iterable): concatenates all elements of an iterable object containing strings only. A string to be used as separator must be specified: separator=’’ 🡪separator.join(iterable)

**Functions and Methods of lists:**

8.9

List functions:

sorted(list): The sorted function returns a new sorted list in ascending order:

sum(list): The sum function returns the sum of all elements of a list (they must all be numbers):

8.10

The main list methods are:

.append(element)🡪appends new element to the list

.insert(index, element)🡪inserts the element in the index position

.remove(element)🡪removes the first occurrence of the element

.pop([index])🡪removes and returns the element at the index. If index is omitted it pops the last element

.extend(list2)🡪appends list2 to the list

.index(element)🡪returns the index of the first occurrence of the element

.sort()🡪sorts the list in ascending order (for descending order use .sort(reverse=True)

.reverse()🡪reverses the elements of the list

.clear()🡪removes all the elements of a list

.count(element)🡪counts the number of times element is found in the list

.copy()🡪returns a copy of the list

8.11

Traversing is sequential access to all elements of a string, list or iterable object.

8.12

Lists of lists can be created, the first index refers to the list and the second to the element of that specific list.