Advanced School in Artificial Intelligence

Introduction to Python

Ing. Zese Riccardo

riccardo.zese@unife.it

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Regione Emilia-Romagna

FERRARIAE UNIVERSITAS · 13 91 · EX LABORE FRUCTUS

Università degli Studi di Ferrara

Outline

- Introduction to Python
- Introduction to Neural Networks
- Convolutional NN
- Recurrent NN
- Autoencoders and self supervised learning

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Outline

- Introduction to Python
- Introduction to Neural Networks
- Convolutional NN
- Recurrent NN
- Autoencoders and self supervised learning

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Operators

- `+` sum
- `-` subtraction
- `*` multiplication
- `/` division (59/60 = 0.9833, while 59/60.0 = 0.9833)
- `//` integer division (2.5//2 = 1.0)
- `**` exponentiation
- `%` module (10%3 = 1)

- Python follows the following precedence:

1. parentheses, 2. exponentiation, 3. multiplication and division, 4. addition and subtraction

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Boolean Operators

- `==` equal
- `!=` different
- `<` lower than
- `<=` lower or equal than
- `>` greater than
- `>=` greater or equal than

They all return a Boolean

- `and`
- `or`
- `not`

In Python `0`, `""` (empty string), `False` and `None` are considered false. Everything else is considered true.
Function `bool` returns the Boolean value of an object.

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Operators with strings, comments and assignments

- `+` concatenation
  - `'Hello' + ' World' = 'Hello World'`
- `*` repetition
  - `'Hi'*3 = 'HiHiHi' = 'Hi' + 'Hi' + 'Hi'`
  - `'Ba' + 'na'*2 = 'Banana'`

- `#` in-line comment
  - `x = 10 # this is an assignment`

- `=` assignment, can be multiple
  - `x = 10 # this is an assignment`
  - `x, y, z = 10, 20, 30 # this is a multiple assignment`

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Multiline comments

```
#This is a long comment
#and it extends
#to multiple lines
```

Another way of doing this is to use triple quotes, either ''' or """.

```
"""This is also a
perfect example of
multi-line comments"""
```

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Type conversion

- **`int(a,base)`** : converts any data type to integer. 'base' specifies the base in which string is if the data type is a string, for example **`int('1101',2)`**
- **`float()`** : converts any data type to a floating point number
- **`str()`** : converts an integer into a string.

- **`ord()`** : converts a character to (integer) unicode representation.
- **`hex()`** : converts an integer to a hexadecimal string.
- **`oct()`** : converts an integer to an octal string
- **`tuple()`** : converts to a tuple.
- **`set()`** : returns the type after converting to set.
- **`list()`** : converts any data type to a list type.
- **`dict()`** : converts a tuple (key,value) into a dictionary.
- **`complex(real,imag)`** : converts two real numbers to a complex number.

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Functions

- Like in every programming language, functions are a sequence of statements executable by using the name of the function and that possibly returns a value.
- We have already seen many functions: input, print, type, ...
- Functions accept 0 or more arguments and can be used with objects of different types, using the syntax **`function(arguments)`**

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Functions

- Functions accept 0 or more arguments and can be used with objects of different types, using the syntax **`function(arguments)`**

```
>>> len("Python") # with a string
6
>>> len([1,2,3]) # with an array
3
>>> len({'a':3,'b':5}) # with a dictionary
2
```

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Methods

- Methods are similar to functions but are linked to the type of object.
- They can be called on objects via the dot notation.

```
>>> s = "Python"
>>> s.upper() # this methods returns "PYTHON"
>>> s.lower() # this method returns "python"
```

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Strings

- To declare a string is sufficient to assign a text enclosed in quotation marks to a new variable.
  - It is possible to use single quotation marks (character ') or double quotation marks (character ").
  - One can always use escape characters
  - `"This is Riccardo's string" = 'This is Riccardo\'s string'`
  - Long strings can be divided on more lines using **triple quotation**
  - One can also compare strings using relational operators.

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Strings

- **Indexing:** use of an index to indicate a character of a string. Index must be integer

```
>>> s = 'Python'
>>> s[0]  # element in pos 0 (first)
'P'
>>> s[5]  # element in pos 5 (sixth)
'n'
>>> s[-1]  # element in pos -1 (last)
'n'
>>> s[-4]  # element in pos -4 (fourth to last)
't'
```

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Strings

- **Slicing:** the possibility to obtain sub-strings by means of indexing

```
>>> s = 'Python'
>>> s[0:2]  # substring with elements from 0 (included) to 2 (excluded)
'Py'
>>> s[:2]   # from the beginning to 2 (excluded)
'Py'
>>> s[3:5]  # from index 3 (incl.) to 5 (excl.)
'ho'
>>> s[4:]   # from index 4 (incl.) to the last
'on'
>>> s[-2:]  # from index -2 (incl.) to the last
'on'
```

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Strings

- **Test of inclusion:** by means of operators in and not in

```
>>> s = 'Python'
>>> 'P' in s  # checks if character 'P' is in the string s
True
>>> 'x' in s  # character 'x' is not in s, returns False
False
>>> 'x' not in s  # "not" to perform inverse of inclusion test
True
>>> 'Py' in s  # checks if substring 'Py' is in string s
True
>>> 'py' in s  # the check is case-sensitive
False
```

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Strings

- **Length of a string:** use of the function **`len`**

```
>>> len("Python")
6
>>> s = "string"
>>> len(s)
6
```

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Strings

- The operator + can be used to concatenate strings

```
"Python" + " is a string of " + str(6) + "characters"
```

Remember to convert other types to string

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Strings

- Other interesting methods:
  - **`find(substring)`**: gives the index of the first character in a matching of the substring from the left or -1 if no such character exists.
  - **`rfind(substring)`**: same as above, but from the right.
  - **`find(substring,i,j)`**: same as **`find()`**, but looks only in **`string[i:j]`**.
  - **`split()`**: divides a string into a list of substrings according to white spaces or a string passed as argument.

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Lists

- Lists are denoted by:

```
list_name = [el0, el1, ..., eln]
```

- Indexing works like in strings:
  - Index goes from 0 to the length of the list – 1

```
list_name[0],   list_name[1],   list_name[2],
list_name[3]
```

- Lists are **mutable**

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Lists

- You can also have an empty list: `[]`.
- You can index into lists from the back.
  - `list_name[-i]` returns the ith element from the back.
- Lists are **heterogeneous**:
  - The elements in a list can be of different types, can have integers and strings.
  - Can even have lists themselves.

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Lists: Functions

- Lists come with lots of useful functions and methods.
  - **`len(list_name)`**, as with strings, returns the length of the list.
  - **`min(list_name)`** and **`max(list_name)`** return the min and max so long as they are well defined.
  - **`sum(list_name)`** returns the sum of elements so long as they're numbers.
    - **Not** defined for lists of strings.

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Lists: Methods

- **`append(value)`** – adds value to the end of the list.
- **`sort()`** - sorts the list so long as this is well defined. (need consistent notions of > and ==)
- **`insert(index, value)`** – inserts the element value at the index specified.
- **`remove(value)`** – removes the first instance of value.
- **`count(value)`** – counts the number of instances of value in the list.
- **`index(value)`** – Return zero-based index in the list of the first item equals to value. Raises a ValueError if there is no such item.

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

List slicing

- Slicing works like in strings.
- **`y=x[i:j]`** gives us a list **`y`** with the elements from **`i`** to **`j-1`** inclusive.
  - **`x[:]`** makes a list that contains all the elements of the original → **copy of a list!**
  - **`x[i:]`** makes a list that contains the elements from **`i`** to the end.
  - **`x[:j]`** makes a list that contains the elements from the beginning to **`j-1`**.
- **`y`** is a **new list**, so that it is not aliased with **`x`**.

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Lists

- As lists are mutable, nested lists can cause situations in which aliasing is non-obvious.
- Be wary of slicing lists that contain mutable elements.
- While the slicing creates new lists, the items in the list are aliases to the original elements.
- So you might be changing both lists when you think you're only changing one.

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

If statement

```
if a == 10:
    # statements
elif a < 0:
    # statements
elif a > 10:
    # statements
else:
    # statements
```

- Every elif and else are optional.
- The code must be indented. Python does not use parentheses to enclose statements but uses indentation by means of **tabulation** (or spaces, usually 4)

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Loops

- **While loop**

```
while condition:
    # statements
```

- There is also **for loop** but it is quite different than what seen in other languages. Let's see first arrays and lists.

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Looping over Lists: for loop

- We want to perform a similar operation to every element of a list.
- Python allows us to do this using for loops

```
for item in list:
    # statements
```

- This is equivalent to:

```
item = list[0]
block
item = list[1]
block
...
```

Do you remember the **`for(element : list)`** of Java?

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

For loop and lists

- **ATTENTION!**

```
for item in list:
    # statements
```

- Here, `item` is **immutable**, so we can't alter the `list` elements.
- If we want to alter the list elements, we need to refer to the **indices** of the list.

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

For loop with indices

- To do that, we use the `range()` function.
- `range(i)` returns an ordered list of integers ranging from 0 to i-1.
- `range(i,j)` returns an ordered list of ints ranging from i to j-1 inclusive.
- `range(i,j,k)` returns a list of integers ranging from i to j-1 with a step of k between ints.
  - So `range(i,k)==range(i,k,1)`

- To modify a list element by element we use:

```
for i in range(len(list)):
    # statements
```

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

More about range()

- The function range returns an **immutable** sequence of numbers.
- It can be used to initialize a list, but not directly: `lst = range(10)` does not create a list so we cannot assign values to `lst` → `lst[2] = 2` will rise an error
- We must use constructors

```
lst = list(range(10))
```

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Nested Lists

- Because lists are heterogeneous, we can have lists of lists.
- This is useful if we want matrices, or to represent a grid or higher dimensional space.
- We then reference elements by `list_name[i][j]` if we want the jth element of the ith list.
- So then naturally, if we wish to loop over all the elements we need nested loops:

```
for item in list:
    for item2 in item:
        # statements
```

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Tuples

- If we need **immutable lists** we can resort to **tuples**

```
tuple_name=(item0,item1,item2,...)
```

  - Note the different use of paratheses!
- Items can be referenced as in lists

```
tuple_name[0], tuple_name[1],...
```

- Single element tuple must be defined using a comma to avoid ambiguity

`(8+3)` vs. `(8+3,)`

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Strings

- Strings can be considered tuples of individual characters since they are immutable.
- We have:
  - Indexing and slicing
  - Strings are not heterogenous, they can only contain characters.
  - Functions **`min()`** and **`max()`** defined on strings, but **`sum()`** is not.

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Sets

- Python also includes a data type for **sets**.
- A set is an unordered collection with no duplicate elements.
- Basic uses include membership testing and eliminating duplicate entries.
- Set objects also support mathematical operations like union, intersection, difference, and symmetric difference.
- Curly braces or the `set()` function can be used to create sets.
- Note: to create an empty set you have to use set(), not {}; the latter creates an empty **dictionary** (we will see it in a couple of slides).

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Sets

```
>>> basket = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'}
>>> print(basket)  # show that duplicates have been removed
{'orange', 'banana', 'pear', 'apple'}
>>> 'orange' in basket  # fast membership testing
True
>>> 'crabgrass' in basket
False
```

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Sets

```
>>> a = set('abracadabra')
>>> b = set('alacazam')
>>> a                  # unique letters in a
{'a', 'r', 'b', 'c', 'd'}
>>> a - b              # letters in a but not in b
{'r', 'd', 'b'}
>>> a | b              # letters in a or b or both
{'a', 'c', 'r', 'd', 'b', 'm', 'z', 'l'}
>>> a & b              # letters in both a and b
{'a', 'c'}
>>> a ^ b              # letters in a or b but not both
{'r', 'd', 'b', 'm', 'z', 'l'}
```

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Dictionaries

- Lists can be only accessed using the index.
- Sometimes it is useful to access data indexed by meaningful information instead of an integer index.
  - Remember maps in Java
- In one sentence, dictionaries are (key, value) pairs. Sometimes they are called, as in Java, maps.

```
{key0 : val0, key1 : val1, ..., keyn : valn}
```

- Dictionaries are of type **dict**. Since they have a type, they can be assigned to a variable.
- To refer to a value associated with a key in a dictionary we use `dictionary_name[key]`

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Dictionaries

- Dictionaries are unsorted.
- Dictionary keys must be immutable, but the values can be anything.
- Cannot be **`None`**.
- Once you've created a dictionary you can add key-value pairs by assigning the value to the key.

```
dictionary_name[key] = value
```

- Keys must be unique.

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Dictionary methods

- **`len(dict_name)`** works in the same way as it does for strings and lists.
- **`+`** and **`*`** are **not** defined for dictionaries.
- **`dict.keys()`** - returns the keys in some order.
- **`dict.values()`** - returns the values in some order.
- **`dict.items()`** - returns the (key, value) pairs in some order.

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Dictionary methods

- **`key in dict`** - returns **`True`** iff the dictionary has the key in it.
- **`dict.get(key)`** – returns the value that is paired with the key, or **`None`** if no such key exists.
  - The function is defined as **`get(key, d=None)`** it returns **`d`** rather than **`None`** (if given) if no such key exists → **`get(key,0)`** returns **`0`** if **`key`** is not found
- **`dict.clear()`** - removes all the key-value pairs from the dictionary.

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Dictionary methods

- **`dict.copy()`** - copy the entire dictionary.
  - Be wary if the dictionary has mutable objects.
  - Can have the same issue as with nested lists.
- **`dict.update(dict_name)`** - adds the key-value pairs in **`dict_name`** to **`dict`**.
- **`dict.pop(key)`** – removes and returns the key-value pair indexed by the key.
- **`dict.popitem()`** – removes and returns one **`(key, value)`** pair.
- To remove items from a dictionary one can use the **`del`** **function** or **`del`** **directive**

  `del(dict_name[key])`

  `del` **`dict_name[key]`**

  Attention: **`del dict_name`** will delete the variable which cannot be further called until a new declaration.

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Looping over dictionaries.

```
for key in d:
     print(key, d[keys])

for val in d.values():
     print(val)

for key,val in d.items():
     print(key, val)
```

- However, the order is still arbitrary.
- How can we make the loop ordered?

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021

Looping over dictionaries

How can we make the loop ordered?

```
dict_keys = dict_name.keys()


dict_keys.sort()


for key in dict_keys:
    print(key, dict_name[key])
```

Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021