2.5: Python in-built Documentation

PyDOC is a documentation tool for Python. It can be used both to access to the documentation of the modules that come with Python, and the documentation of third party modules.

In Windows, go to the icon "Module Docs" documentation of the standard library and "Python Manuals" to view the tutorial, referrals, and other more extensive documentation.

To use PyDOC in Linux:

```
pydoc ./modulo.py
```

To show the documentation of `modulo.py` in the current directory.

In Linux, the documentation of libraries can be seen throught the *browser* by using:

```
pydoc -p 8000
```

At the address **http://localhost:8000/ <http://localhost:8000/>**. To run the graphical version of PyDOC type:

```
pydoc -g
```

PyDOC uses the module *Doc Strings* to generate the documentation.

Besides that, it is also possible to consult the documentation on the interpreter itself, with the function `help()`.

Example:

```
help(list)
```

```
Help on class list in module builtins:

class list(object)
 |  list() -> new empty list
 |  list(iterable) -> new list initialized from iterable's items
 |
 |  Methods defined here:
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __contains__(self, key, /)
 |      Return key in self.
```

```
|
|  __delitem__(self, key, /)
|      Delete self[key].
|
|  __eq__(self, value, /)
|      Return self==value.
|
|  __ge__(self, value, /)
|      Return self>=value.
|
|  __getattribute__(self, name, /)
|      Return getattr(self, name).
|
|  __getitem__(...)
|      x.__getitem__(y) <==> x[y]
|
|  __gt__(self, value, /)
|      Return self>value.
|
|  __iadd__(self, value, /)
|      Implement self+=value.
|
|  __imul__(self, value, /)
|      Implement self*=value.
|
|  __init__(self, /, *args, **kwargs)
|      Initialize self.  See help(type(self)) for accurate signature.
|
|  __iter__(self, /)
|      Implement iter(self).
|
|  __le__(self, value, /)
|      Return self<=value.
|
|  __len__(self, /)
|      Return len(self).
|
|  __lt__(self, value, /)
|      Return self<value.
|
|  __mul__(self, value, /)
|      Return self*value.n
|
|  __ne__(self, value, /)
|      Return self!=value.
|
|  __new__(*args, **kwargs) from builtins.type
|      Create and return a new object.  See help(type) for accurate signature.
|
|  __repr__(self, /)
|      Return repr(self).
|
|  __reversed__(...)
|      L.__reversed__() -- return a reverse iterator over the list
|
```

```
|  __rmul__(self, value, /)
|      Return self*value.
|
|  __setitem__(self, key, value, /)
|      Set self[key] to value.
|
|  __sizeof__(...)
|      L.__sizeof__() -- size of L in memory, in bytes
|
|  append(...)
|      L.append(object) -> None -- append object to end
|
|  clear(...)
|      L.clear() -> None -- remove all items from L
|
|  copy(...)
|      L.copy() -> list -- a shallow copy of L
|
|  count(...)
|      L.count(value) -> integer -- return number of occurrences of value
|
|  extend(...)
|      L.extend(iterable) -> None -- extend list by appending elements from the iterable
|
|  index(...)
|      L.index(value, [start, [stop]]) -> integer -- return first index of value.
|      Raises ValueError if the value is not present.
|
|  insert(...)
|      L.insert(index, object) -- insert object before index
|
|  pop(...)
|      L.pop([index]) -> item -- remove and return item at index (default last).
|      Raises IndexError if list is empty or index is out of range.
|
|  remove(...)
|      L.remove(value) -> None -- remove first occurrence of value.
|      Raises ValueError if the value is not present.
|
|  reverse(...)
|      L.reverse() -- reverse *IN PLACE*
|
|  sort(...)
|      L.sort(key=None, reverse=False) -> None -- stable sort *IN PLACE*
|
|  ----------------------------------------------------------------------
|  Data and other attributes defined here:
|
|  __hash__ = None
```

Which shows the Python list documentation.