Modulo1_Clase1

January 31, 2021

1 TEMA 2.Manejo básico de PYTHOM II: Estructura y tipo de datos

En este capitulo aprenderemos a identificar los tipo de datos, tambien se aprendera a como declararlas y como manejar cada comando en cada tipo de dato

1.0.1 INSTALEMOS LOS PAQUETES HA UTILIZAR

[1]:
```
pip install pandas
```

```
Requirement already satisfied: pandas in c:\programdata\anaconda3\lib\site-
packages (1.1.3)
Requirement already satisfied: numpy>=1.15.4 in
c:\programdata\anaconda3\lib\site-packages (from pandas) (1.19.2)
Requirement already satisfied: pytz>=2017.2 in
c:\programdata\anaconda3\lib\site-packages (from pandas) (2020.1)
Requirement already satisfied: python-dateutil>=2.7.3 in
c:\programdata\anaconda3\lib\site-packages (from pandas) (2.8.1)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-
packages (from python-dateutil>=2.7.3->pandas) (1.15.0)
Note: you may need to restart the kernel to use updated packages.
```

[2]:
```
pip install matplotlib
```

```
Requirement already satisfied: matplotlib in c:\programdata\anaconda3\lib\site-
packages (3.3.2)
Requirement already satisfied: cycler>=0.10 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib) (0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib) (2.8.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib) (1.3.0)
Requirement already satisfied: numpy>=1.15 in c:\programdata\anaconda3\lib\site-
packages (from matplotlib) (1.19.2)
Requirement already satisfied: certifi>=2020.06.20 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib) (2020.6.20)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib) (2.4.7)
Requirement already satisfied: pillow>=6.2.0 in
```

```
c:\programdata\anaconda3\lib\site-packages (from matplotlib) (8.0.1)
Requirement already satisfied: six in c:\programdata\anaconda3\lib\site-packages
(from cycler>=0.10->matplotlib) (1.15.0)
Note: you may need to restart the kernel to use updated packages.
```

[3]:
```
pip install numpy
```

```
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-
packages (1.19.2)
Note: you may need to restart the kernel to use updated packages.
```

[4]:
```
import pandas as pd
import matplotlib as plt
import numpy as np
```

1.1 COMENTARIO

[5]:
```
!pip list
```

```
Package                           Version
--------------------------------- -------------------
alabaster                         0.7.12
anaconda-client                   1.7.2
anaconda-navigator                1.10.0
anaconda-project                  0.8.3
argh                              0.26.2
argon2-cffi                       20.1.0
asn1crypto                        1.4.0
astroid                           2.4.2
astropy                           4.0.2
async-generator                   1.10
atomicwrites                      1.4.0
attrs                             20.3.0
autopep8                          1.5.4
Babel                             2.8.1
backcall                          0.2.0
backports.functools-lru-cache     1.6.1
backports.shutil-get-terminal-size 1.0.0
backports.tempfile                1.0
backports.weakref                 1.0.post1
bcrypt                            3.2.0
beautifulsoup4                    4.9.3
bitarray                          1.6.1
bkcharts                          0.2
bleach                            3.2.1
bokeh                             2.2.3
boto                              2.49.0
Bottleneck                        1.3.2
```

brotlipy	0.7.0
certifi	2020.6.20
cffi	1.14.3
chardet	3.0.4
click	7.1.2
cloudpickle	1.6.0
clyent	1.2.2
colorama	0.4.4
comtypes	1.1.7
conda	4.9.2
conda-build	3.20.5
conda-package-handling	1.7.2
conda-verify	3.4.2
contextlib2	0.6.0.post1
cryptography	3.1.1
cycler	0.10.0
Cython	0.29.21
cytoolz	0.11.0
dask	2.30.0
decorator	4.4.2
defusedxml	0.6.0
diff-match-patch	20200713
distributed	2.30.1
docutils	0.16
entrypoints	0.3
et-xmlfile	1.0.1
fastcache	1.1.0
filelock	3.0.12
flake8	3.8.4
Flask	1.1.2
fsspec	0.8.3
future	0.18.2
gevent	20.9.0
glob2	0.7
greenlet	0.4.17
h5py	2.10.0
HeapDict	1.0.1
html5lib	1.1
idna	2.10
imageio	2.9.0
imagesize	1.2.0
importlib-metadata	2.0.0
iniconfig	1.1.1
intervaltree	3.1.0
ipykernel	5.3.4
ipython	7.19.0
ipython-genutils	0.2.0
ipywidgets	7.5.1

isort	5.6.4
itsdangerous	1.1.0
jdcal	1.4.1
jedi	0.17.1
Jinja2	2.11.2
joblib	0.17.0
json5	0.9.5
jsonschema	3.2.0
jupyter	1.0.0
jupyter-client	6.1.7
jupyter-console	6.2.0
jupyter-core	4.6.3
jupyterlab	2.2.6
jupyterlab-pygments	0.1.2
jupyterlab-server	1.2.0
keyring	21.4.0
kiwisolver	1.3.0
lazy-object-proxy	1.4.3
libarchive-c	2.9
llvmlite	0.34.0
locket	0.2.0
lxml	4.6.1
MarkupSafe	1.1.1
matplotlib	3.3.2
mccabe	0.6.1
menuinst	1.4.16
mistune	0.8.4
mkl-fft	1.2.0
mkl-random	1.1.1
mkl-service	2.3.0
mock	4.0.2
more-itertools	8.6.0
mpmath	1.1.0
msgpack	1.0.0
multipledispatch	0.6.0
navigator-updater	0.2.1
nbclient	0.5.1
nbconvert	6.0.7
nbformat	5.0.8
nest-asyncio	1.4.2
networkx	2.5
nltk	3.5
nose	1.3.7
notebook	6.1.4
numba	0.51.2
numexpr	2.7.1
numpy	1.19.2
numpydoc	1.1.0

olefile	0.46
openpyxl	3.0.5
packaging	20.4
pandas	1.1.3
pandocfilters	1.4.3
paramiko	2.7.2
parso	0.7.0
partd	1.1.0
path	15.0.0
pathlib2	2.3.5
pathtools	0.1.2
patsy	0.5.1
pep8	1.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	8.0.1
pip	20.2.4
pkginfo	1.6.1
pluggy	0.13.1
ply	3.11
prometheus-client	0.8.0
prompt-toolkit	3.0.8
psutil	5.7.2
psycopg2	2.8.6
py	1.9.0
pycodestyle	2.6.0
pycosat	0.6.3
pycparser	2.20
pycurl	7.43.0.6
pydocstyle	5.1.1
pyflakes	2.2.0
Pygments	2.7.2
pylint	2.6.0
PyNaCl	1.4.0
pyodbc	4.0.0-unsupported
pyOpenSSL	19.1.0
pyparsing	2.4.7
pyreadline	2.1
pyreadstat	1.0.8
pyrsistent	0.17.3
PySocks	1.7.1
pytest	0.0.0
python-dateutil	2.8.1
python-jsonrpc-server	0.4.0
python-language-server	0.35.1
pytz	2020.1
PyWavelets	1.1.1
pywin32	227

pywin32-ctypes	0.2.0
pywinpty	0.5.7
PyYAML	5.3.1
pyzmq	19.0.2
QDarkStyle	2.8.1
QtAwesome	1.0.1
qtconsole	4.7.7
QtPy	1.9.0
regex	2020.10.15
requests	2.24.0
rope	0.18.0
Rtree	0.9.4
ruamel-yaml	0.15.87
scikit-image	0.17.2
scikit-learn	0.23.2
scipy	1.5.2
seaborn	0.11.0
Send2Trash	1.5.0
setuptools	50.3.1.post20201107
simplegeneric	0.8.1
singledispatch	3.4.0.3
sip	4.19.13
six	1.15.0
snowballstemmer	2.0.0
sortedcollections	1.2.1
sortedcontainers	2.2.2
soupsieve	2.0.1
Sphinx	3.2.1
sphinxcontrib-applehelp	1.0.2
sphinxcontrib-devhelp	1.0.2
sphinxcontrib-htmlhelp	1.0.3
sphinxcontrib-jsmath	1.0.1
sphinxcontrib-qthelp	1.0.3
sphinxcontrib-serializinghtml	1.1.4
sphinxcontrib-websupport	1.2.4
spyder	4.1.5
spyder-kernels	1.9.4
SQLAlchemy	1.3.20
statsmodels	0.12.0
sympy	1.6.2
tables	3.6.1
tblib	1.7.0
terminado	0.9.1
testpath	0.4.4
threadpoolctl	2.1.0
tifffile	2020.10.1
toml	0.10.1
toolz	0.11.1

```
tornado                         6.0.4
tqdm                            4.50.2
traitlets                       5.0.5
typing-extensions               3.7.4.3
ujson                           4.0.1
unicodecsv                      0.14.1
urllib3                         1.25.11
watchdog                        0.10.3
wcwidth                         0.2.5
webencodings                    0.5.1
Werkzeug                        1.0.1
wheel                           0.35.1
widgetsnbextension              3.5.1
win-inet-pton                   1.1.0
win-unicode-console             0.5
wincertstore                    0.2
wrapt                           1.11.2
xlrd                            1.2.0
XlsxWriter                      1.3.7
xlwings                         0.20.8
xlwt                            1.3.0
xmltodict                       0.12.0
yapf                            0.30.0
zict                            2.0.0
zipp                            3.4.0
zope.event                      4.5.0
zope.interface                  5.1.2
```

[6]:
```
import sys
sys.path
```

[6]:
```
['C:\\Users\\user',
 'C:\\ProgramData\\Anaconda3\\python38.zip',
 'C:\\ProgramData\\Anaconda3\\DLLs',
 'C:\\ProgramData\\Anaconda3\\lib',
 'C:\\ProgramData\\Anaconda3',
 '',
 'C:\\ProgramData\\Anaconda3\\lib\\site-packages',
 'C:\\ProgramData\\Anaconda3\\lib\\site-packages\\win32',
 'C:\\ProgramData\\Anaconda3\\lib\\site-packages\\win32\\lib',
 'C:\\ProgramData\\Anaconda3\\lib\\site-packages\\Pythonwin',
 'C:\\ProgramData\\Anaconda3\\lib\\site-packages\\IPython\\extensions',
 'C:\\Users\\user\\.ipython']
```

[7]:
```
#MOSTRAMOS EN PANTALA UN MENSAJE
print("Hola mundo")
```

```
Hola mundo
```

1.1.1 SENTENCIAS DE CONTROLES

1.1.2 if, else, elif

[8]:
```
x=2
y=0
```

[9]:
```
if x==4:
    y=5
else:
    y=2
```

[10]:
```
y
```

[10]: 2

[11]:
```
x=1
y=0
```

[12]:
```
if x==4:
    y=1
elif x==5:
    y=2
else:
    y=3
```

[13]:
```
y
```

[13]: 3

[14]:
```
list(range(0,6))
```

[14]: [0, 1, 2, 3, 4, 5]

1.1.3 bucle for

[15]:
```
lista=list(range(0,6))
lista
```

[15]: [0, 1, 2, 3, 4, 5]

[16]:
```
# HACEMOS QUE NUESTRO ITERADOR RECORRA TODA LA LISTA
for i in lista:
    print(i)
```

```
0
1
2
```

```
3
4
5
```

1.1.4 bucle while

[17]:
```
i=0
while i<10:
    print(i)
    i+=1
```

```
0
1
2
3
4
5
6
7
8
9
```

Manejo de objetos

1.1.5 Listas

Las listas contienen valores de cualquier tipo simple (numérico o no numérico), y podrían ser estructuras compuestas (lista de listas). Si usamos como referencia a una hoja de calculo con datos sobre individuos, una lista podria ser una fila que tiene los datos de los individuos.

[18]:
```
#Para declarar una lista se pone entre []
lista1=[9,8,7]
lista1
```

[18]:
```
[9, 8, 7]
```

[19]:
```
lista2=lista1
lista2
```

[19]:
```
[9, 8, 7]
```

[20]:
```
#Si quero mostar un elemento de mi lista solo pongo el monbre de mi lista
# y se abre conchetes y colocas la ubicacion de tu elemnto que quieres
#mostrar. por ejemplo:
lista1[2]=5
lista1
```

[20]:
```
[9, 8, 5]
```

[21]:
```
lista2
```

[21]:
```
[9, 8, 5]
```

[22]:
```
type(lista1)
```

[22]:
```
list
```

[23]:
```
#Mostrar los métodos de un determinado tipo
dir(lista)
```

[23]:
```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__delitem__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__gt__',
 '__hash__',
 '__iadd__',
 '__imul__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__reversed__',
 '__rmul__',
 '__setattr__',
 '__setitem__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'append',
```

```
 'clear',
 'copy',
 'count',
 'extend',
 'index',
 'insert',
 'pop',
 'remove',
 'reverse',
 'sort']
```

[24]:
```
help(list)
```

```
Help on class list in module builtins:

class list(object)
 |  list(iterable=(), /)
 |
 |  Built-in mutable sequence.
 |
 |  If no argument is given, the constructor creates a new empty list.
 |  The argument must be an iterable if specified.
 |
 |  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.
|
|  __ne__(self, value, /)
|      Return self!=value.
|
|  __repr__(self, /)
|      Return repr(self).
|
|  __reversed__(self, /)
|      Return a reverse iterator over the list.
|
|  __rmul__(self, value, /)
|      Return value*self.
|
|  __setitem__(self, key, value, /)
|      Set self[key] to value.
|
|  __sizeof__(self, /)
|      Return the size of the list in memory, in bytes.
|
|  append(self, object, /)
|      Append object to the end of the list.
|
|  clear(self, /)
```

```
 |      Remove all items from list.
 |
 |  copy(self, /)
 |      Return a shallow copy of the list.
 |
 |  count(self, value, /)
 |      Return number of occurrences of value.
 |
 |  extend(self, iterable, /)
 |      Extend list by appending elements from the iterable.
 |
 |  index(self, value, start=0, stop=9223372036854775807, /)
 |      Return first index of value.
 |
 |      Raises ValueError if the value is not present.
 |
 |  insert(self, index, object, /)
 |      Insert object before index.
 |
 |  pop(self, index=-1, /)
 |      Remove and return item at index (default last).
 |
 |      Raises IndexError if list is empty or index is out of range.
 |
 |  remove(self, value, /)
 |      Remove first occurrence of value.
 |
 |      Raises ValueError if the value is not present.
 |
 |  reverse(self, /)
 |      Reverse *IN PLACE*.
 |
 |  sort(self, /, *, key=None, reverse=False)
 |      Sort the list in ascending order and return None.
 |
 |      The sort is in-place (i.e. the list itself is modified) and stable (i.e.
the
 |      order of two equal elements is maintained).
 |
 |      If a key function is given, apply it once to each list item and sort
them,
 |      ascending or descending, according to their function values.
 |
 |      The reverse flag can be set to sort in descending order.
 |
 |  ----------------------------------------------------------------------
 |  Static methods defined here:
 |
```

```
|  __new__(*args, **kwargs) from builtins.type
|      Create and return a new object.  See help(type) for accurate signature.
|
|  ----------------------------------------------------------------------
|  Data and other attributes defined here:
|
|  __hash__ = None
```

[25]:
```
lista3=lista1.copy()
lista3
```

[25]: [9, 8, 5]

[26]:
```
lista1[1]=4
lista2
```

[26]: [9, 4, 5]

[27]:
```
lista3
```

[27]: [9, 8, 5]

[158]:
```
Estudiante=["Manuel Ponte",23,"False"]
Estudiante
```

[158]: ['Manuel Ponte', 23, 'False']

[159]:
```
Estudiante[0] # primer elemento comienza con indice '0'
```

[159]: 'Manuel Ponte'

[160]:
```
Estudiante[:2] # todo antes de índice 2
```

[160]: ['Manuel Ponte', 23]

[161]:
```
Estudiante[-2] # ultimo elementos
```

[161]: 23

Si quieres **cambiar** algun valor:

[162]:
```
Estudiante[0]='Omar Escobedo'
Estudiante
```

[162]: ['Omar Escobedo', 23, 'False']

Para **añadir** elementos:

[163]:
```
Estudiante.append('UNFV')

# Ahora tienes:
Estudiante
```

[163]:
```
['Omar Escobedo', 23, 'False', 'UNFV']
```

[164]:
```
del Estudiante[3]
Estudiante
```

[164]:
```
['Omar Escobedo', 23, 'False']
```

[165]:
```
elementsA=[11,22,33,44]
elementsB=[11,22,33,44]
```

[166]:
```
## borrar tercer elemento
del elementsA[2]
# luego:
elementsA  #
```

[166]:
```
[11, 22, 44]
```

[167]:
```
#  borrar valor '22'
# se puede eliminar por posición o por valor
elementsB.remove(22)
elementsB
```

[167]:
```
[11, 33, 44]
```

1.1.6 CADENA

[28]:
```
cadena="HOLA MUNDO"
cadena
```

[28]:
```
'HOLA MUNDO'
```

[29]:
```
type(cadena)
```

[29]:
```
str
```

[30]:
```
dir(str)
```

[30]:
```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dir__',
```

```
'__doc__',
'__eq__',
'__format__',
'__ge__',
'__getattribute__',
'__getitem__',
'__getnewargs__',
'__gt__',
'__hash__',
'__init__',
'__init_subclass__',
'__iter__',
'__le__',
'__len__',
'__lt__',
'__mod__',
'__mul__',
'__ne__',
'__new__',
'__reduce__',
'__reduce_ex__',
'__repr__',
'__rmod__',
'__rmul__',
'__setattr__',
'__sizeof__',
'__str__',
'__subclasshook__',
'capitalize',
'casefold',
'center',
'count',
'encode',
'endswith',
'expandtabs',
'find',
'format',
'format_map',
'index',
'isalnum',
'isalpha',
'isascii',
'isdecimal',
'isdigit',
'isidentifier',
'islower',
'isnumeric',
```

```
 'isprintable',
 'isspace',
 'istitle',
 'isupper',
 'join',
 'ljust',
 'lower',
 'lstrip',
 'maketrans',
 'partition',
 'replace',
 'rfind',
 'rindex',
 'rjust',
 'rpartition',
 'rsplit',
 'rstrip',
 'split',
 'splitlines',
 'startswith',
 'strip',
 'swapcase',
 'title',
 'translate',
 'upper',
 'zfill']
```

[31]:
```
help(str)
```

```
Help on class str in module builtins:

class str(object)
 |  str(object='') -> str
 |  str(bytes_or_buffer[, encoding[, errors]]) -> str
 |
 |  Create a new string object from the given object. If encoding or
 |  errors is specified, then the object must expose a data buffer
 |  that will be decoded using the given encoding and error handler.
 |  Otherwise, returns the result of object.__str__() (if defined)
 |  or repr(object).
 |  encoding defaults to sys.getdefaultencoding().
 |  errors defaults to 'strict'.
 |
 |  Methods defined here:
 |
 |  __add__(self, value, /)
 |      Return self+value.
```

```
|
|  __contains__(self, key, /)
|      Return key in self.
|
|  __eq__(self, value, /)
|      Return self==value.
|
|  __format__(self, format_spec, /)
|      Return a formatted version of the string as described by format_spec.
|
|  __ge__(self, value, /)
|      Return self>=value.
|
|  __getattribute__(self, name, /)
|      Return getattr(self, name).
|
|  __getitem__(self, key, /)
|      Return self[key].
|
|  __getnewargs__(...)
|
|  __gt__(self, value, /)
|      Return self>value.
|
|  __hash__(self, /)
|      Return hash(self).
|
|  __iter__(self, /)
|      Implement iter(self).
|
|  __le__(self, value, /)
|      Return self<=value.
|
|  __len__(self, /)
|      Return len(self).
|
|  __lt__(self, value, /)
|      Return self<value.
|
|  __mod__(self, value, /)
|      Return self%value.
|
|  __mul__(self, value, /)
|      Return self*value.
|
|  __ne__(self, value, /)
|      Return self!=value.
|
```

```
 |  __repr__(self, /)
 |      Return repr(self).
 |
 |  __rmod__(self, value, /)
 |      Return value%self.
 |
 |  __rmul__(self, value, /)
 |      Return value*self.
 |
 |  __sizeof__(self, /)
 |      Return the size of the string in memory, in bytes.
 |
 |  __str__(self, /)
 |      Return str(self).
 |
 |  capitalize(self, /)
 |      Return a capitalized version of the string.
 |
 |      More specifically, make the first character have upper case and the rest
lower
 |      case.
 |
 |  casefold(self, /)
 |      Return a version of the string suitable for caseless comparisons.
 |
 |  center(self, width, fillchar=' ', /)
 |      Return a centered string of length width.
 |
 |      Padding is done using the specified fill character (default is a space).
 |
 |  count(...)
 |      S.count(sub[, start[, end]]) -> int
 |
 |      Return the number of non-overlapping occurrences of substring sub in
 |      string S[start:end].  Optional arguments start and end are
 |      interpreted as in slice notation.
 |
 |  encode(self, /, encoding='utf-8', errors='strict')
 |      Encode the string using the codec registered for encoding.
 |
 |      encoding
 |        The encoding in which to encode the string.
 |      errors
 |        The error handling scheme to use for encoding errors.
 |        The default is 'strict' meaning that encoding errors raise a
 |        UnicodeEncodeError.  Other possible values are 'ignore', 'replace' and
 |        'xmlcharrefreplace' as well as any other name registered with
 |        codecs.register_error that can handle UnicodeEncodeErrors.
```

```
 |
 |  endswith(...)
 |      S.endswith(suffix[, start[, end]]) -> bool
 |
 |      Return True if S ends with the specified suffix, False otherwise.
 |      With optional start, test S beginning at that position.
 |      With optional end, stop comparing S at that position.
 |      suffix can also be a tuple of strings to try.
 |
 |  expandtabs(self, /, tabsize=8)
 |      Return a copy where all tab characters are expanded using spaces.
 |
 |      If tabsize is not given, a tab size of 8 characters is assumed.
 |
 |  find(...)
 |      S.find(sub[, start[, end]]) -> int
 |
 |      Return the lowest index in S where substring sub is found,
 |      such that sub is contained within S[start:end].  Optional
 |      arguments start and end are interpreted as in slice notation.
 |
 |      Return -1 on failure.
 |
 |  format(...)
 |      S.format(*args, **kwargs) -> str
 |
 |      Return a formatted version of S, using substitutions from args and
kwargs.
 |      The substitutions are identified by braces ('{' and '}').
 |
 |  format_map(...)
 |      S.format_map(mapping) -> str
 |
 |      Return a formatted version of S, using substitutions from mapping.
 |      The substitutions are identified by braces ('{' and '}').
 |
 |  index(...)
 |      S.index(sub[, start[, end]]) -> int
 |
 |      Return the lowest index in S where substring sub is found,
 |      such that sub is contained within S[start:end].  Optional
 |      arguments start and end are interpreted as in slice notation.
 |
 |      Raises ValueError when the substring is not found.
 |
 |  isalnum(self, /)
 |      Return True if the string is an alpha-numeric string, False otherwise.
 |
```

```
 |      A string is alpha-numeric if all characters in the string are alpha-
numeric and
 |      there is at least one character in the string.
 |
 |  isalpha(self, /)
 |      Return True if the string is an alphabetic string, False otherwise.
 |
 |      A string is alphabetic if all characters in the string are alphabetic
and there
 |      is at least one character in the string.
 |
 |  isascii(self, /)
 |      Return True if all characters in the string are ASCII, False otherwise.
 |
 |      ASCII characters have code points in the range U+0000-U+007F.
 |      Empty string is ASCII too.
 |
 |  isdecimal(self, /)
 |      Return True if the string is a decimal string, False otherwise.
 |
 |      A string is a decimal string if all characters in the string are decimal
and
 |      there is at least one character in the string.
 |
 |  isdigit(self, /)
 |      Return True if the string is a digit string, False otherwise.
 |
 |      A string is a digit string if all characters in the string are digits
and there
 |      is at least one character in the string.
 |
 |  isidentifier(self, /)
 |      Return True if the string is a valid Python identifier, False otherwise.
 |
 |      Call keyword.iskeyword(s) to test whether string s is a reserved
identifier,
 |      such as "def" or "class".
 |
 |  islower(self, /)
 |      Return True if the string is a lowercase string, False otherwise.
 |
 |      A string is lowercase if all cased characters in the string are
lowercase and
 |      there is at least one cased character in the string.
 |
 |  isnumeric(self, /)
 |      Return True if the string is a numeric string, False otherwise.
 |
```

```
 |      A string is numeric if all characters in the string are numeric and
there is at
 |      least one character in the string.
 |
 |  isprintable(self, /)
 |      Return True if the string is printable, False otherwise.
 |
 |      A string is printable if all of its characters are considered printable
in
 |      repr() or if it is empty.
 |
 |  isspace(self, /)
 |      Return True if the string is a whitespace string, False otherwise.
 |
 |      A string is whitespace if all characters in the string are whitespace
and there
 |      is at least one character in the string.
 |
 |  istitle(self, /)
 |      Return True if the string is a title-cased string, False otherwise.
 |
 |      In a title-cased string, upper- and title-case characters may only
 |      follow uncased characters and lowercase characters only cased ones.
 |
 |  isupper(self, /)
 |      Return True if the string is an uppercase string, False otherwise.
 |
 |      A string is uppercase if all cased characters in the string are
uppercase and
 |      there is at least one cased character in the string.
 |
 |  join(self, iterable, /)
 |      Concatenate any number of strings.
 |
 |      The string whose method is called is inserted in between each given
string.
 |      The result is returned as a new string.
 |
 |      Example: '.'.join(['ab', 'pq', 'rs']) -> 'ab.pq.rs'
 |
 |  ljust(self, width, fillchar=' ', /)
 |      Return a left-justified string of length width.
 |
 |      Padding is done using the specified fill character (default is a space).
 |
 |  lower(self, /)
 |      Return a copy of the string converted to lowercase.
 |
```

```
 |  lstrip(self, chars=None, /)
 |      Return a copy of the string with leading whitespace removed.
 |
 |      If chars is given and not None, remove characters in chars instead.
 |
 |  partition(self, sep, /)
 |      Partition the string into three parts using the given separator.
 |
 |      This will search for the separator in the string.  If the separator is
found,
 |      returns a 3-tuple containing the part before the separator, the
separator
 |      itself, and the part after it.
 |
 |      If the separator is not found, returns a 3-tuple containing the original
string
 |      and two empty strings.
 |
 |  replace(self, old, new, count=-1, /)
 |      Return a copy with all occurrences of substring old replaced by new.
 |
 |        count
 |          Maximum number of occurrences to replace.
 |          -1 (the default value) means replace all occurrences.
 |
 |      If the optional argument count is given, only the first count
occurrences are
 |      replaced.
 |
 |  rfind(...)
 |      S.rfind(sub[, start[, end]]) -> int
 |
 |      Return the highest index in S where substring sub is found,
 |      such that sub is contained within S[start:end].  Optional
 |      arguments start and end are interpreted as in slice notation.
 |
 |      Return -1 on failure.
 |
 |  rindex(...)
 |      S.rindex(sub[, start[, end]]) -> int
 |
 |      Return the highest index in S where substring sub is found,
 |      such that sub is contained within S[start:end].  Optional
 |      arguments start and end are interpreted as in slice notation.
 |
 |      Raises ValueError when the substring is not found.
 |
 |  rjust(self, width, fillchar=' ', /)
```

```
 |      Return a right-justified string of length width.
 |
 |      Padding is done using the specified fill character (default is a space).
 |
 |  rpartition(self, sep, /)
 |      Partition the string into three parts using the given separator.
 |
 |      This will search for the separator in the string, starting at the end.
If
 |      the separator is found, returns a 3-tuple containing the part before the
 |      separator, the separator itself, and the part after it.
 |
 |      If the separator is not found, returns a 3-tuple containing two empty
strings
 |      and the original string.
 |
 |  rsplit(self, /, sep=None, maxsplit=-1)
 |      Return a list of the words in the string, using sep as the delimiter
string.
 |
 |        sep
 |          The delimiter according which to split the string.
 |          None (the default value) means split according to any whitespace,
 |          and discard empty strings from the result.
 |        maxsplit
 |          Maximum number of splits to do.
 |          -1 (the default value) means no limit.
 |
 |      Splits are done starting at the end of the string and working to the
front.
 |
 |  rstrip(self, chars=None, /)
 |      Return a copy of the string with trailing whitespace removed.
 |
 |      If chars is given and not None, remove characters in chars instead.
 |
 |  split(self, /, sep=None, maxsplit=-1)
 |      Return a list of the words in the string, using sep as the delimiter
string.
 |
 |      sep
 |        The delimiter according which to split the string.
 |        None (the default value) means split according to any whitespace,
 |        and discard empty strings from the result.
 |      maxsplit
 |        Maximum number of splits to do.
 |        -1 (the default value) means no limit.
 |
```

```
 |  splitlines(self, /, keepends=False)
 |      Return a list of the lines in the string, breaking at line boundaries.
 |
 |      Line breaks are not included in the resulting list unless keepends is
given and
 |      true.
 |
 |  startswith(...)
 |      S.startswith(prefix[, start[, end]]) -> bool
 |
 |      Return True if S starts with the specified prefix, False otherwise.
 |      With optional start, test S beginning at that position.
 |      With optional end, stop comparing S at that position.
 |      prefix can also be a tuple of strings to try.
 |
 |  strip(self, chars=None, /)
 |      Return a copy of the string with leading and trailing whitespace
removed.
 |
 |      If chars is given and not None, remove characters in chars instead.
 |
 |  swapcase(self, /)
 |      Convert uppercase characters to lowercase and lowercase characters to
uppercase.
 |
 |  title(self, /)
 |      Return a version of the string where each word is titlecased.
 |
 |      More specifically, words start with uppercased characters and all
remaining
 |      cased characters have lower case.
 |
 |  translate(self, table, /)
 |      Replace each character in the string using the given translation table.
 |
 |        table
 |          Translation table, which must be a mapping of Unicode ordinals to
 |          Unicode ordinals, strings, or None.
 |
 |      The table must implement lookup/indexing via __getitem__, for instance a
 |      dictionary or list.  If this operation raises LookupError, the character
is
 |      left untouched.  Characters mapped to None are deleted.
 |
 |  upper(self, /)
 |      Return a copy of the string converted to uppercase.
 |
 |  zfill(self, width, /)
```

```
 |      Pad a numeric string with zeros on the left, to fill a field of the
given width.
 |
 |      The string is never truncated.
 |
 |  ------------------------------------------------------------------------
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs) from builtins.type
 |      Create and return a new object.  See help(type) for accurate signature.
 |
 |  maketrans(...)
 |      Return a translation table usable for str.translate().
 |
 |      If there is only one argument, it must be a dictionary mapping Unicode
 |      ordinals (integers) or characters to Unicode ordinals, strings or None.
 |      Character keys will be then converted to ordinals.
 |      If there are two arguments, they must be strings of equal length, and
 |      in the resulting dictionary, each character in x will be mapped to the
 |      character at the same position in y. If there is a third argument, it
 |      must be a string, whose characters will be mapped to None in the result.
```

[32]:
```
#LOWE(...): retorna la copia del objeto cadena convertida en minuscula
cadena.lower()
```

[32]:
```
'hola mundo'
```

[33]:
```
cadena.lstrip()
```

[33]:
```
'HOLA MUNDO'
```

1.1.7 Tipos

Por defecto todas las cadenas de texto son Unicode. Así, cualquier string declarado en Python será automáticamente de tipo Unicode. Los tipos son:

1.Unicode. 2.Byte. 3.Bytearray.

Byte El tipo byte solo admite caracteres en codificación **ASCII** y, al igual que los de tipo Unicode, son inmutables.

Para declarar un string de tipo **byte**, basta con anteponer la letra b antes de las comillas:

[34]:
```
cad = b"cadena de tipo byte"
```

[35]:
```
cad
```

[35]:
```
b'cadena de tipo byte'
```

[36]:
```
type(cad)
```

[36]:
```
bytes
```

Bytearray El tipo **bytearray** es una versión mutable del tipo byte La declaración de un tipo bytearray debe hacerse utilizando la función integrada que nos ofrece el intérprete. Además, es **imprescindible indicar el tipo de codificación que deseamos emplear**. El siguiente ejemplo utiliza la codificación de los caracteres latin1 para crear un **string** de este tipo:

[37]:
```
lat = bytearray("España", #contenido
                "latin1") #tipo de formato
print(lat)
```

```
bytearray(b'Espa\xf1a')
```

[38]:
```
str = "España"
array1 = bytearray(str, 'utf-8')
print(array1)
```

```
bytearray(b'Espa\xc3\xb1a')
```

[39]:
```
bytearray("España", "utf16")
```

[39]:
```
bytearray(b'\xff\xfeE\x00s\x00p\x00a\x00\xf1\x00a\x00')
```

encode(): transforma un tipo str en tipo byte

[40]:
```
cad = "es de tipo str"
#objetounicode.encode()
cad.encode()
```

[40]:
```
b'es de tipo str'
```

decode(): transforma un tipo byte en tipo str

[41]:
```
cad = b"es de tipo byte"
cad.decode()
```

[41]:
```
'es de tipo byte'
```

Principales funciones y métodos

[42]:
```
cad = "cadena de texto de ejemplo"
len(cad) #count de los elementos del objeto cad
```

[42]:
```
26
```

[43]:
```
cad = "xyza"
cad.find("a") #find() realiza una búsqueda del elemento
```

[43]: 3

[44]:
```
cad = "Hola Mundo"
cad.replace("Hola", #objeto a reemplazar
            "Adiós") #reemplazo
```

[44]: 'Adiós Mundo'

[45]:
```
#eliminar espacios en blanco
cad = " cadena con espacios en blanco "
cad
```

[45]: ' cadena con espacios en blanco '

[46]:
```
cad.strip()
```

[46]: 'cadena con espacios en blanco'

[47]:
```
cad.lstrip()
```

[47]: 'cadena con espacios en blanco '

[48]:
```
cad.rstrip()
```

[48]: ' cadena con espacios en blanco'

[49]:
```
#convertir a mayúsculas y minúsculas
cad2 = cad.upper()
print(cad2)
```

```
CADENA CON ESPACIOS EN BLANCO
```

[50]:
```
print(cad2.lower())
```

```
cadena con espacios en blanco
```

[51]:
```
#convertir primer carácter a mayúscula
cad = "un ejemplo"
cad.capitalize()
```

[51]: 'Un ejemplo'

[52]:
```
#dividir una cadena de texto basándose en un carácter
cad = "primer valor; segundo; tercer valor"
cad
```

[52]:
```
'primer valor; segundo; tercer valor'
```

[53]:
```
cad.split(";") #split para señalar un elemento que nos ayudará a dividir el␣
↪objeto
```

[53]:
```
['primer valor', ' segundo', ' tercer valor']
```

[54]:
```
",".join("abc")
```

[54]:
```
'a,b,c'
```

Operaciones

[55]:
```
#concatenar
cad_concat = "¡Hola" + " Mundo!"
print(cad_concat)
```

```
¡Hola Mundo!
```

[56]:
```
str(23)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-56-9653f67e19ce> in <module>
----> 1 str(23)

TypeError: 'str' object is not callable
```

[57]:
```
print("Hola Mundo " * 4)
```

```
Hola Mundo Hola Mundo Hola Mundo Hola Mundo
```

[58]:
```
cad = "Nueva cadena de texto"
"d" in cad #consulta si el valor d está presente en el objeto cad
```

[58]:
```
True
```

[59]:
```
cad = "Cadenas"
print(cad[2])
```

```
d
```

[60]:
```
print(cad[:3])
```

```
Cad
```

[61]:
```
cad[-3]
```

[61]: 'n'

[62]:
```
cad[3:]
```

[62]: 'enas'

1.2 NUMEROS

[63]:
```
numero=4
numero
```

[63]: 4

[64]:
```
type(numero)
```

[64]: int

[65]:
```
type(int)
```

[65]: type

[66]:
```
isinstance(4,int)
```

[66]: True

[67]:
```
isinstance(cadena,int)
```

[67]: False

[68]:
```
numero=4
numero
```

[68]: 4

[69]:
```
type(numero)
```

[69]: int

[70]:
```
num_real=4.5
type(num_real)
```

[70]: float

[71]:
```
num_real=0.5e-7
num_real
```

[71]: 5e-08

[72]:
```
num_complejo=3.2+7j
num_complejo
```

[72]:
```
(3.2+7j)
```

[73]:
```
type(num_complejo)
```

[73]:
```
complex
```

1.2.1 Conjuntos

[74]:
```
conjunto1=set("468")
conjunto1
```

[74]:
```
{'4', '6', '8'}
```

[75]:
```
type(conjunto1)
```

[75]:
```
set
```

[76]:
```
conjunto2={1,5,6}
conjunto2
```

[76]:
```
{1, 5, 6}
```

[77]:
```
dir(set)
```

[77]:
```
['__and__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__iand__',
 '__init__',
 '__init_subclass__',
 '__ior__',
 '__isub__',
 '__iter__',
 '__ixor__',
 '__le__',
```

```
 '__len__',
 '__lt__',
 '__ne__',
 '__new__',
 '__or__',
 '__rand__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__ror__',
 '__rsub__',
 '__rxor__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__sub__',
 '__subclasshook__',
 '__xor__',
 'add',
 'clear',
 'copy',
 'difference',
 'difference_update',
 'discard',
 'intersection',
 'intersection_update',
 'isdisjoint',
 'issubset',
 'issuperset',
 'pop',
 'remove',
 'symmetric_difference',
 'symmetric_difference_update',
 'union',
 'update']
```

[78]:
```
help(set)
```

```
Help on class set in module builtins:

class set(object)
 |  set() -> new empty set object
 |  set(iterable) -> new set object
 |
 |  Build an unordered collection of unique elements.
 |
 |  Methods defined here:
```

```
|
|  __and__(self, value, /)
|      Return self&value.
|
|  __contains__(...)
|      x.__contains__(y) <==> y in x.
|
|  __eq__(self, value, /)
|      Return self==value.
|
|  __ge__(self, value, /)
|      Return self>=value.
|
|  __getattribute__(self, name, /)
|      Return getattr(self, name).
|
|  __gt__(self, value, /)
|      Return self>value.
|
|  __iand__(self, value, /)
|      Return self&=value.
|
|  __init__(self, /, *args, **kwargs)
|      Initialize self.  See help(type(self)) for accurate signature.
|
|  __ior__(self, value, /)
|      Return self|=value.
|
|  __isub__(self, value, /)
|      Return self-=value.
|
|  __iter__(self, /)
|      Implement iter(self).
|
|  __ixor__(self, value, /)
|      Return self^=value.
|
|  __le__(self, value, /)
|      Return self<=value.
|
|  __len__(self, /)
|      Return len(self).
|
|  __lt__(self, value, /)
|      Return self<value.
|
|  __ne__(self, value, /)
|      Return self!=value.
```

```
|
|  __or__(self, value, /)
|      Return self|value.
|
|  __rand__(self, value, /)
|      Return value&self.
|
|  __reduce__(...)
|      Return state information for pickling.
|
|  __repr__(self, /)
|      Return repr(self).
|
|  __ror__(self, value, /)
|      Return value|self.
|
|  __rsub__(self, value, /)
|      Return value-self.
|
|  __rxor__(self, value, /)
|      Return value^self.
|
|  __sizeof__(...)
|      S.__sizeof__() -> size of S in memory, in bytes
|
|  __sub__(self, value, /)
|      Return self-value.
|
|  __xor__(self, value, /)
|      Return self^value.
|
|  add(...)
|      Add an element to a set.
|
|      This has no effect if the element is already present.
|
|  clear(...)
|      Remove all elements from this set.
|
|  copy(...)
|      Return a shallow copy of a set.
|
|  difference(...)
|      Return the difference of two or more sets as a new set.
|
|      (i.e. all elements that are in this set but not the others.)
|
|  difference_update(...)
```

```
|      Remove all elements of another set from this set.
|
|  discard(...)
|      Remove an element from a set if it is a member.
|
|      If the element is not a member, do nothing.
|
|  intersection(...)
|      Return the intersection of two sets as a new set.
|
|      (i.e. all elements that are in both sets.)
|
|  intersection_update(...)
|      Update a set with the intersection of itself and another.
|
|  isdisjoint(...)
|      Return True if two sets have a null intersection.
|
|  issubset(...)
|      Report whether another set contains this set.
|
|  issuperset(...)
|      Report whether this set contains another set.
|
|  pop(...)
|      Remove and return an arbitrary set element.
|      Raises KeyError if the set is empty.
|
|  remove(...)
|      Remove an element from a set; it must be a member.
|
|      If the element is not a member, raise a KeyError.
|
|  symmetric_difference(...)
|      Return the symmetric difference of two sets as a new set.
|
|      (i.e. all elements that are in exactly one of the sets.)
|
|  symmetric_difference_update(...)
|      Update a set with the symmetric difference of itself and another.
|
|  union(...)
|      Return the union of sets as a new set.
|
|      (i.e. all elements that are in either set.)
|
|  update(...)
|      Update a set with the union of itself and others.
```

```
 |
 |  ----------------------------------------------------------------------
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs) from builtins.type
 |      Create and return a new object.  See help(type) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  __hash__ = None
```

[79]:
```
print(conjunto1)
print(conjunto2)
```

```
{'4', '8', '6'}
{1, 5, 6}
```

[80]:
```
conjunto1.intersection(conjunto2)
#La interseccion no se da ya que el conjunto 1 los numeros lo esta
#leyendo como una cadena string y en cambio el conjunto 2 lo lee como
#variables numericas
```

[80]:
```
set()
```

[81]:
```
conjunto1={4,6,8}
conjunto1
```

[81]:
```
{4, 6, 8}
```

[82]:
```
conjunto1.intersection(conjunto2)
```

[82]:
```
{6}
```

[83]:
```
Estudiante=["Brayan",23,"False"]
Estudiante
```

[83]:
```
['Brayan', 23, 'False']
```

[84]:
```
Estudiante[0]
```

[84]:
```
'Brayan'
```

[85]:
```
Estudiante[0:2]
```

[85]:
```
['Brayan', 23]
```

si queremos obtener los elementos de DERECHA A IZQUIERDA

[86]:
```
Estudiante[-1]
```

[86]:
```
'False'
```

[87]:
```
Estudiante[-2]
```

[87]:
```
23
```

[88]:
```
#Si queremos agregar un nuevo elemento a mi lista usamos "append()"
#pero este comando lo agregara al final
Estudiante.append("Domingo")
Estudiante
```

[88]:
```
['Brayan', 23, 'False', 'Domingo']
```

[89]:
```
#Si queremos eliminar un elemento de la lista usamos "pop()"
#este comando elimina el ultimo elemento
Estudiante.pop()
Estudiante
```

[89]:
```
['Brayan', 23, 'False']
```

[90]:
```
dir(list)
```

[90]:
```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__delitem__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__gt__',
 '__hash__',
 '__iadd__',
 '__imul__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
```

```
 '__lt__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__reversed__',
 '__rmul__',
 '__setattr__',
 '__setitem__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'append',
 'clear',
 'copy',
 'count',
 'extend',
 'index',
 'insert',
 'pop',
 'remove',
 'reverse',
 'sort']
```

[91]:
```
help(list)
```

```
Help on class list in module builtins:

class list(object)
 |  list(iterable=(), /)
 |
 |  Built-in mutable sequence.
 |
 |  If no argument is given, the constructor creates a new empty list.
 |  The argument must be an iterable if specified.
 |
 |  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.
|
|  __ne__(self, value, /)
|      Return self!=value.
|
|  __repr__(self, /)
|      Return repr(self).
|
|  __reversed__(self, /)
```

```
|      Return a reverse iterator over the list.
|
|  __rmul__(self, value, /)
|      Return value*self.
|
|  __setitem__(self, key, value, /)
|      Set self[key] to value.
|
|  __sizeof__(self, /)
|      Return the size of the list in memory, in bytes.
|
|  append(self, object, /)
|      Append object to the end of the list.
|
|  clear(self, /)
|      Remove all items from list.
|
|  copy(self, /)
|      Return a shallow copy of the list.
|
|  count(self, value, /)
|      Return number of occurrences of value.
|
|  extend(self, iterable, /)
|      Extend list by appending elements from the iterable.
|
|  index(self, value, start=0, stop=9223372036854775807, /)
|      Return first index of value.
|
|      Raises ValueError if the value is not present.
|
|  insert(self, index, object, /)
|      Insert object before index.
|
|  pop(self, index=-1, /)
|      Remove and return item at index (default last).
|
|      Raises IndexError if list is empty or index is out of range.
|
|  remove(self, value, /)
|      Remove first occurrence of value.
|
|      Raises ValueError if the value is not present.
|
|  reverse(self, /)
|      Reverse *IN PLACE*.
|
|  sort(self, /, *, key=None, reverse=False)
```

```
 |      Sort the list in ascending order and return None.
 |
 |      The sort is in-place (i.e. the list itself is modified) and stable (i.e.
the
 |      order of two equal elements is maintained).
 |
 |      If a key function is given, apply it once to each list item and sort
them,
 |      ascending or descending, according to their function values.
 |
 |      The reverse flag can be set to sort in descending order.
 |
 |  ----------------------------------------------------------------------
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs) from builtins.type
 |      Create and return a new object.  See help(type) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  __hash__ = None
```

[92]:
```
#Si queremos agregar un elemto a mi lista en cualquier posicion
#usamos el comando "insert"
Estudiante.insert(1,"Smit")
Estudiante
```

[92]:
```
['Brayan', 'Smit', 23, 'False']
```

1.2.2 Vectores

[93]:
```
import numpy as np
```

[94]:
```
vector1=np.array([1,2,3])
vector1
```

[94]:
```
array([1, 2, 3])
```

[95]:
```
#Que tipo es el vector1
type(vector1)
```

[95]:
```
numpy.ndarray
```

[96]:
```
vector2=np.array([1,"2",3])
vector2
```

[96]:
```
array(['1', '2', '3'], dtype='<U11')
```

[97]:
```
#Que tipo es el vector2
type(vector2)
```

[97]:
```
numpy.ndarray
```

[98]:
```
#Modemos modificar los elementos del vector
vector1[1]=1.5
vector1
#al cambiar el elemesto 2 por 1.5 se mostrara la parte entera de 1.5
# que es 1 debido a que sus elementos son enteros
```

[98]:
```
array([1, 1, 3])
```

[99]:
```
# en cambio en el vector dos al modificar su elemento 2 por 1.5 si se
#mostrara el 1.5 debido a que 2 es de tipo string
vector2[1]=1.5
vector2
```

[99]:
```
array(['1', '1.5', '3'], dtype='<U11')
```

Tuplas

[100]:
```
#Las Tuplas siempre va en parentesis
EstudianteTupla=("carloS",25,"False")
EstudianteTupla
```

[100]:
```
('carloS', 25, 'False')
```

[101]:
```
#Una caracteristicas de Tuplas es que no son modificables
EstudianteTupla[2]=2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-101-76d103107836> in <module>
      1 #Una caracteristicas de Tuplas es que no son modificables
----> 2 EstudianteTupla[2]=2

TypeError: 'tuple' object does not support item assignment
```

[102]:
```
#Pero si podemos mostra cualquier elemento que elijamos por ejemplo
EstudianteTupla[0]
```

[102]:
```
'carloS'
```

[103]:
```
#Que tipo es la Tupla
type(EstudianteTupla)
```

[103]:
```
tuple
```

[104]:
```
t=(1,3,1,5,1)
t
```

[104]:
```
(1, 3, 1, 5, 1)
```

[105]:
```
#El comando  "count()" cuenta cuantos elementos que elijas se repite
#por ejemplo
t.count(1)
#Podemos ver que el elemento 1 se repite 3 veces y eso se mostrara
```

[105]:
```
3
```

[106]:
```
#El comando "len()" nos da la cantidad de elemtos de mi Tupla por ejemplo
len(t)
#Podemos ver que la Tupla "t" tiene 5 elemento y eso se mostrara
```

[106]:
```
5
```

[107]:
```
help(t.count)
```

```
Help on built-in function count:

count(value, /) method of builtins.tuple instance
    Return number of occurrences of value.
```

1.2.3 Diccionario

[108]:
```
#¿Porqué es importante diccionario?Para general un DataFrame con el modulo
#panda hay diferentes manera, pero una de las maneras es mediante un
#diccionario
#los diccionarios se trabajara con {}
#Los diccionarios responde a un key(es el campo por ejamplo: Nombre Edad,etc)
#y valve(son los elementosde ese campo)
#Diccionario {key:valve}
EstudianteDict={"Nombres":"Juan",
                "Edad":23,
                "Masculino":True}
EstudianteDict
```

[108]:
```
{'Nombres': 'Juan', 'Edad': 23, 'Masculino': True}
```

[109]:
```
#En un diccionario para mostar los elementos no se pide como las lista que
#se pone entre [] y colocas el numero donde esta ubicada tu elemento sino
#que se ponde entre llaves y eliges el campo que quieres. Veamos
EstudianteDict[0]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-109-9d3df3ef7343> in <module>
      2 #se pone entre [] y colocas el numero donde esta ubicada tu elemento sino
      3 #que se ponde entre llaves y eliges el campo que quieres. Veamos
----> 4 EstudianteDict[0]

KeyError: 0
```

[110]:
```
#Mostremos el campo "Nombres"
EstudianteDict["Nombres"]
```

[110]:
```
'Juan'
```

[111]:
```
#Mostremos el campo "Edad"
EstudianteDict["Edad"]
```

[111]:
```
23
```

[112]:
```
#Que tipo es "EstudianteDict"
type(EstudianteDict)
```

[112]:
```
dict
```

[113]:
```
dir(dict)
```

[113]:
```
['__class__',
 '__contains__',
 '__delattr__',
 '__delitem__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
```

```
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__reversed__',
 '__setattr__',
 '__setitem__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'clear',
 'copy',
 'fromkeys',
 'get',
 'items',
 'keys',
 'pop',
 'popitem',
 'setdefault',
 'update',
 'values']
```

[114]:
```
EstudianteDict1={"Nombres":{"Carlos","Sandra","Smit","Gloria"},
                 "Edad":{23,22,26,27},
                 "Sexo":{"Masculino","Femenino","Masculino","Femenino"}}
EstudianteDict1
```

[114]:
```
{'Nombres': {'Carlos', 'Gloria', 'Sandra', 'Smit'},
 'Edad': {22, 23, 26, 27},
 'Sexo': {'Femenino', 'Masculino'}}
```

[115]:
```
EstudianteDict1["Nombres"]
```

[115]:
```
{'Carlos', 'Gloria', 'Sandra', 'Smit'}
```

[116]:
```
EstudianteDict1["Edad"]
```

[116]:
```
{22, 23, 26, 27}
```

[117]:
```
EstudianteDict1["Sexo"]
```

[117]:
```
{'Femenino', 'Masculino'}
```

1.2.4 DataFrames

Los *Data Frames* pueden interpretarse como estructuras compuestas en base a las simples. Python requiere que llamemos al paquete pandas para usar DFs:

[118]:
```
#Generando un DF mediante un diccionario
nombre=["Manuel","Antuane","Kevin","Keiko"]
edad=[23,22,23,21]
pais=["Perú","Ecuador","Mexico","Perú"]
educacion=["Bach","Lic","Lic","Bach"]
```

[119]:
```
data={"nombre":nombre,
      "edad":edad,
      "pais":pais,
      "educacion":educacion}
data
```

[119]:
```
{'nombre': ['Manuel', 'Antuane', 'Kevin', 'Keiko'],
 'edad': [23, 22, 23, 21],
 'pais': ['Perú', 'Ecuador', 'Mexico', 'Perú'],
 'educacion': ['Bach', 'Lic', 'Lic', 'Bach']}
```

[120]:
```
estudiantes=pd.DataFrame(data)
estudiantes
```

[120]:

	nombre	edad	pais	educacion
0	Manuel	23	Perú	Bach
1	Antuane	22	Ecuador	Lic
2	Kevin	23	Mexico	Lic
3	Keiko	21	Perú	Bach

1.3 Generando un DF mediante una lista de lista

[121]:
```
fila1=["Manuel",23,"Perú","Bach"]
fila2=["Antuane",22,"Ecuador","Lic"]
fila3=["Kevin",23,"Mexico","Lic"]
fila4=["Keiko",21,"Perú","Bach"]
```

[122]:
```
listofRows=[fila1,fila2,fila3,fila4]
listofRows
```

[122]:
```
[['Manuel', 23, 'Perú', 'Bach'],
 ['Antuane', 22, 'Ecuador', 'Lic'],
 ['Kevin', 23, 'Mexico', 'Lic'],
 ['Keiko', 21, 'Perú', 'Bach']]
```

[123]:
```
estudiantes2=pd.
 ↪DataFrame(listofRows,columns=["nombres","edad","pais","educacion"])
estudiantes2
```

[123]:

	nombres	edad	pais	educacion
0	Manuel	23	Perú	Bach
1	Antuane	22	Ecuador	Lic
2	Kevin	23	Mexico	Lic
3	Keiko	21	Perú	Bach

[124]:
```
type(estudiantes)
```

[124]:
```
pandas.core.frame.DataFrame
```

[125]:
```
#Verificando los tipos de datos de nuestro DF
estudiantes.dtypes
```

[125]:
```
nombre       object
edad          int64
pais         object
educacion    object
dtype: object
```

[126]:
```
#Las dimenciones de nuestra DF
estudiantes.shape
```

[126]:
```
(4, 4)
```

[127]:
```
#Obteniendo el número de filas
NroFilas=estudiantes.shape[1]
NroFilas
```

[127]:
```
4
```

[128]:
```
#Obteniendo el número de columnas
NroColumnas=estudiantes.shape[1]
NroColumnas
```

[128]:
```
4
```

[129]:
```
estudiantes
```

[129]:

	nombre	edad	pais	educacion
0	Manuel	23	Perú	Bach
1	Antuane	22	Ecuador	Lic
2	Kevin	23	Mexico	Lic
3	Keiko	21	Perú	Bach

[130]:
```
edad1=estudiantes["edad"]
edad1
```

[130]:
```
0    23
1    22
2    23
3    21
Name: edad, dtype: int64
```

[131]:
```
type(edad1)
```

[131]:
```
pandas.core.series.Series
```

[132]:
```
edad2=estudiantes["edad"].values
edad2
```

[132]:
```
array([23, 22, 23, 21], dtype=int64)
```

[133]:
```
type(edad2)
```

[133]:
```
numpy.ndarray
```

[134]:
```
edad3=estudiantes.edad
edad3
```

[134]:
```
0    23
1    22
2    23
3    21
Name: edad, dtype: int64
```

[135]:
```
type(edad3)
```

[135]:
```
pandas.core.series.Series
```

[136]:
```
#Obteniendo las matrices de las columnas
#columnas=estudiantes.columns
columnas=list(estudiantes.columns)
columnas
```

[136]:
```
['nombre', 'edad', 'pais', 'educacion']
```

[137]:
```
estudiantes
```

[137]:

	nombre	edad	pais	educacion
0	Manuel	23	Perú	Bach
1	Antuane	22	Ecuador	Lic

```
2    Kevin    23   Mexico       Lic
3    Keiko    21     Perú      Bach
```

[138]:
```
columnasseleccionadas=['nombre','educacion']
columnasseleccionadas
```

[138]:
```
['nombre', 'educacion']
```

[139]:
```
estudiantes[columnasseleccionadas]
```

[139]:
```
    nombre educacion
0   Manuel      Bach
1  Antuane       Lic
2    Kevin       Lic
3    Keiko      Bach
```

utilizando .loc y .iloc

[140]:
```
#.Loc a nivel de locación, se nombra la variable
varselected=['edad','educacion']
varselected
```

[140]:
```
['edad', 'educacion']
```

[141]:
```
estudiantes.loc[:,'edad':'educacion']
```

[141]:
```
   edad     pais educacion
0    23     Perú      Bach
1    22  Ecuador       Lic
2    23   Mexico       Lic
3    21     Perú      Bach
```

[142]:
```
#i.loc a nivel de indices, se coloca el respectivo indice
estudiantes.iloc[:,1:4]
```

[142]:
```
   edad     pais educacion
0    23     Perú      Bach
1    22  Ecuador       Lic
2    23   Mexico       Lic
3    21     Perú      Bach
```

[143]:
```
estudiantes.iloc[:,1:3]
```

[143]:
```
   edad     pais
0    23     Perú
1    22  Ecuador
2    23   Mexico
```

```
3    21     Perú
```

[144]:
```
estudiantes
```

[144]:
```
    nombre  edad     pais educacion
0   Manuel    23     Perú      Bach
1  Antuane    22  Ecuador       Lic
2    Kevin    23   Mexico       Lic
3    Keiko    21     Perú      Bach
```

[145]:
```
#seleccionando filas
estudiantes.iloc[[2,3],:]
```

[145]:
```
  nombre  edad    pais educacion
2  Kevin    23  Mexico       Lic
3  Keiko    21    Perú      Bach
```

[146]:
```
estudiantes.iloc[2,3]
```

[146]:
```
'Lic'
```

1.4 Generando cambios en el DF

[147]:
```
estudiantes2noedu=estudiantes2.iloc[:,0:3]
estudiantes2noedu
```

[147]:
```
   nombres  edad     pais
0   Manuel    23     Perú
1  Antuane    22  Ecuador
2    Kevin    23   Mexico
3    Keiko    21     Perú
```

[148]:
```
estudiantescopia=estudiantes.copy()
estudiantescopia
```

[148]:
```
    nombre  edad     pais educacion
0   Manuel    23     Perú      Bach
1  Antuane    22  Ecuador       Lic
2    Kevin    23   Mexico       Lic
3    Keiko    21     Perú      Bach
```

[149]:
```
estudiantescopia.iloc[0,1]=25
estudiantescopia
```

[149]:
```
    nombre  edad     pais educacion
0   Manuel    25     Perú      Bach
1  Antuane    22  Ecuador       Lic
```

```
2    Kevin    23   Mexico       Lic
3    Keiko    21     Perú      Bach
```

[150]:
```
#Para eliminar una columna por nombre de variable
estudiantescopia.drop(labels='edad',#
                      axis=1,#axis=0 por filas y axis=1 por columnas
                      inplace=True)
estudiantescopia
```

[150]:
```
    nombre     pais educacion
0   Manuel     Perú      Bach
1  Antuane  Ecuador       Lic
2    Kevin   Mexico       Lic
3    Keiko     Perú      Bach
```

[151]:
```
#Para eliminar una columna por indice de variable
estudiantescopia.drop(labels=estudiantescopia.columns[2],
                      axis=1,
                      inplace=True)
estudiantescopia
```

[151]:
```
    nombre     pais
0   Manuel     Perú
1  Antuane  Ecuador
2    Kevin   Mexico
3    Keiko     Perú
```

1.4.1 Consultas al DataFrame

[152]:
```
estudiantes2
```

[152]:
```
   nombres  edad     pais educacion
0   Manuel    23     Perú      Bach
1  Antuane    22  Ecuador       Lic
2    Kevin    23   Mexico       Lic
3    Keiko    21     Perú      Bach
```

[153]:
```
#¿Quien es el estudiante mas joven?
estudiantes2[estudiantes.edad==min(estudiantes2.edad)].nombres
```

[153]:
```
3    Keiko
Name: nombres, dtype: object
```

[154]:
```
#¿quienes son estudiantes mas veteranos?
estudiantes2[estudiantes2.edad==max(estudiantes2.edad)].nombres
```

[154]:
```
0    Manuel
2     Kevin
Name: nombres, dtype: object
```

[155]:
```
#¿que estudiantes no son Peruanos?
estudiantes2[estudiantes2.pais!="Perú"].nombres
```

[155]:
```
1    Antuane
2      Kevin
Name: nombres, dtype: object
```

[156]:
```
#otra forma: Mediante el uso de .isin
paises=['Ecuador','Mexico']
estudiantes2[estudiantes2.pais.isin(paises)].nombres
```

[156]:
```
1    Antuane
2      Kevin
Name: nombres, dtype: object
```

[157]:
```
#Funcion para ordenar un DF
tosort=['educacion','edad']
ascendente=[True,False]
estudiantes2.sort_values(by=tosort,ascending=ascendente)
```

[157]:

	nombres	edad	pais	educacion
0	Manuel	23	Perú	Bach
3	Keiko	21	Perú	Bach
2	Kevin	23	Mexico	Lic
1	Antuane	22	Ecuador	Lic

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: