{

"cells": [

{

"attachments": {

"image.png": {

"image/png": ""

}

},

"cell\_type": "markdown",

"metadata": {},

"source": [

"# Numpy crash course\n",

"![image.png](attachment:image.png)\n",

"\n",

"[Documentazione - tutorial](https://www.scipy-lectures.org/intro/numpy/array\_object.html)\n",

"\n",

"[Python Data Science Handbook Chapter 2](https://jakevdp.github.io/PythonDataScienceHandbook/02.00-introduction-to-numpy.html)\n",

"\n",

"### Costruzione"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"import numpy as np\n",

"\n",

"a = np.array([1, 2, 3])\n",

"a"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"a = np.array(range(10))\n",

"a"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"a = list(range(1000000))\n",

"aa = np.array(a)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"%%timeit\n",

"s = 0\n",

"for e in a:\n",

" s = s + e\n",

"#print(s)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"%%timeit\n",

"sum(a)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"%%timeit\n",

"np.sum(aa)"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"### Basi"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"a = np.array([[1, 2, 3], [4, 5, 6]])\n",

"b = np.array(['c','d','efg']) # Array di stringhe\n",

"print(a)\n",

"print(a.ndim)\n",

"print(b)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"print(a.shape, b.shape)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"print(a.dtype, b.dtype)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"len(a)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# Un esempio di array a 3 dimensioni\n",

"r, g, b, h = 1, 0.5, 0, 0.3\n",

"a = np.array([[[r,g,b,h], [r,g,b,h], [r,g,b,h]],\n",

" [[r,g,b,h], [r,g,b,h], [r,g,b,h]]])\n",

"a.shape"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"### Array 2D"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# Creo un array con 2 righe e 3 colonne\n",

"b = np.array([[1, 2, 3], [4, 5, 6]])\n",

"b"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"b.ndim"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"b.shape"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"assert b.ndim == len(\n",

" b.shape\n",

") # Questo vale sempre. Ma visto che siamo principianti del linguaggio, controlliamo."

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"(Parentesi: [cosa sono le asserzioni e quando ha senso usarle?](https://mail.python.org/pipermail/python-list/2013-November/660568.html))"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"### Costruttori utili"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"np.arange(10)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"np.arange(1, 9, 2) # start, end (exclusive), step"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"np.linspace(0, 1, 11)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"np.ones((5,)) # Nota: tupla di un elemento"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"np.random.rand(5)"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"### Tipi di dati negli array numpy\n",

"[Documentazione](https://www.scipy-lectures.org/intro/numpy/array\_object.html#basic-data-types)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"a.dtype"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"b = np.array([10, 20, 30, 40])\n",

"print(b, b.dtype)"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"### Indicizzazione 1D\n",

"[Documentazione](https://www.scipy-lectures.org/intro/numpy/array\_object.html#indexing-and-slicing)\n",

"\n",

"Possiamo indicizzare gli array 1D come se fossero liste"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"print(b[0], b[1], b[-1])"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"Slicing"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"print(b[0:3], b[:2], b[2:], b[:-2])"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"### Indicizzazione 2D"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"a = np.array([[1, 2, 3], [4, 5, 6]])\n",

"a"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"print(a[0, 1], a[-1, -2])"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"print(a[1, :], a[:, 2])"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"print(a[:, [1]], a[:, 1])"

]

},

{

"attachments": {

"image.png": {

"image/png": ""

}

},

"cell\_type": "markdown",

"metadata": {},

"source": [

"![image.png](attachment:image.png)"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"### Esercizio\n",

"Creiamo questo array:\n",

"```\n",

"[[0., 0., 0., 0., 0.],\n",

" [2., 0., 0., 0., 0.],\n",

" [0., 3., 0., 0., 0.],\n",

" [0., 0., 4., 0., 0.],\n",

" [0., 0., 0., 5., 0.],\n",

" [0., 0., 0., 0., 6.]]\n",

"```"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"### Fancy Indexing\n",

"Possiamo indicizzare un array con un array di interi\n",

"\n",

"[Documentazione](https://www.scipy-lectures.org/intro/numpy/array\_object.html#fancy-indexing)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"b = np.arange(10) \* 10\n",

"print(b)\n",

"print(b[[0,2,-1,1]])"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"b[[0, 2, -1]] = -1 # Vale anche per settare i valori, non solo per leggerli\n",

"print(b)"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"... oppure con un array di booleani (boolean mask) della stessa dimensione del mio array"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"b"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"b[[True, False, False, True, False, \n",

" False, False, False, False, False]]"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"Esempio: restituisci solo i valori maggiori di 55\n"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"print(b)\n",

"print(b > 55)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"mask = b > 55\n",

"print(mask)\n",

"print(b[mask])"

]

},

{

"attachments": {

"image.png": {

"image/png": ""

}

},

"cell\_type": "markdown",

"metadata": {},

"source": [

"Esempi di fancy indexing\n",

"\n",

"![image.png](attachment:image.png)"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"### Operazioni sugli array\n",

"[Documentazione completa](https://www.scipy-lectures.org/intro/numpy/operations.html)\n",

"\n",

"Di queste ci interessano due classi di operazioni:\n",

"- Operazioni elemento per elemento (somma, sottrazione, prodotto...)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"a = np.array([[1, 2, 3], [4, 5, 6]])\n",

"print(a)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"print(a + 1)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"print(a + 1)"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"- Riduzioni (ad esempio, `sum`, `max`, `min`)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"a"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"print(np.sum(a)) # Somma di tutti gli elementi\n",

"print(\n",

" np.sum(a, axis=0)\n",

") # Somma di ciascuna colonna (agisce lungo l'asse delle righe, ovvero l'asse 0)\n",

"print(\n",

" np.sum(a, axis=1)\n",

") # Somma di ciascuna riga (agisce lungo l'asse delle colonne, ovvero l'asse 1)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"print(np.max(a))\n",

"print(np.min(a, axis=0))"

]

}

],

"metadata": {

"kernelspec": {

"display\_name": "Python 3 (ipykernel)",

"language": "python",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.13.2"

}

},

"nbformat": 4,

"nbformat\_minor": 4

}