test2

July 17, 2021

1 Lists and Arrays in Python

[1]:
```
import array
import numpy as np
```

1.1 Python List

[2]:
```
my_list = [True, "Hello", 42.0, 420, None]

print([type(val) for val in my_list])
```

```
[<class 'bool'>, <class 'str'>, <class 'float'>, <class 'int'>, <class
'NoneType'>]
```

1.2 Python Array

[3]:
```
my_array_range = list(range(10))
my_array = array.array('i', my_array_range)

print(my_array)
```

```
array('i', [0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

1.2.1 Array Dtypes

Type code	C Type	Python Type	Minimum size in bytes
'b'	signed char	int	1
'B'	unsigned char	int	1
'u'	wchar_t	Unicode character	2
'h'	signed short	int	2
'H'	unsigned short	int	2
'i'	signed int	int	2
'I'	unsigned int	int	2
'l'	signed long	int	4
'L'	unsigned long	int	4
'q'	signed long long	int	8
'Q'	unsigned long long	int	8

‘f’	float	float	4
‘d’	double	float	8

[4]:
```
my_array_range = list(range(10))
my_array = array.array('d', my_array_range)

print(my_array)
```

```
array('d', [0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0])
```

1.3 Numpy Array

NumPy is the fundamental package for scientific computing in Python.
It is a Python library that provides a multidimensional array object, various derived objects, and an assortment of routines for fast operations on arrays.

At the core of the NumPy package, is the ndarray object.
This encapsulates n-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance.

[5]:
```
def array_info(array: np.ndarray) -> None:
    print(f"ndim: {array.ndim}")
    print(f"shape: {array.shape}")
    print(f"size: {array.size}")
    print(f"dtype: {array.dtype}")
    print(f"values:\n{array}\n")
```

[6]:
```
my_np_array = np.array([1, 4, 2, 5, 3])
array_info(my_np_array)
```

```
ndim: 1
shape: (5,)
size: 5
dtype: int64
values:
[1 4 2 5 3]
```

[7]:
```
my_np_array = np.array([3.14, 4, 2, 3])
array_info(my_np_array)
```

```
ndim: 1
shape: (4,)
size: 4
dtype: float64
values:
[3.14 4.   2.   3.  ]
```

[8]:
```
my_np_array = np.array([1, 2, 3, 4], dtype='float32')
array_info(my_np_array)
```

```
ndim: 1
shape: (4,)
size: 4
dtype: float32
values:
[1. 2. 3. 4.]
```

1.4 Intrinsic Arrays

[9]:
```
my_np_array = np.zeros(shape=10, dtype=int)
array_info(my_np_array)
```

```
ndim: 1
shape: (10,)
size: 10
dtype: int64
values:
[0 0 0 0 0 0 0 0 0 0]
```

[10]:
```
my_np_array = np.ones(shape=(3, 5), dtype=float)
array_info(my_np_array)
```

```
ndim: 2
shape: (3, 5)
size: 15
dtype: float64
values:
[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
```

[11]:
```
my_np_array = np.full(shape=(3, 5), fill_value=3.14)
array_info(my_np_array)
```

```
ndim: 2
shape: (3, 5)
size: 15
dtype: float64
values:
[[3.14 3.14 3.14 3.14 3.14]
 [3.14 3.14 3.14 3.14 3.14]
 [3.14 3.14 3.14 3.14 3.14]]
```

[12]:
```
my_np_array = np.arange(start=0, stop=20, step=2)
array_info(my_np_array)
```

```
ndim: 1
shape: (10,)
size: 10
dtype: int64
values:
[ 0  2  4  6  8 10 12 14 16 18]
```

[13]:
```
my_np_array = np.linspace(start=0, stop=1, num=5)
array_info(my_np_array)
```

```
ndim: 1
shape: (5,)
size: 5
dtype: float64
values:
[0.   0.25 0.5  0.75 1.  ]
```

[14]:
```
my_np_array = np.eye(N=3)
array_info(my_np_array)
```

```
ndim: 2
shape: (3, 3)
size: 9
dtype: float64
values:
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

[15]:
```
my_np_array = np.diag([1, 2, 3])
array_info(my_np_array)
```

```
ndim: 2
shape: (3, 3)
size: 9
dtype: int64
values:
[[1 0 0]
 [0 2 0]
 [0 0 3]]
```

1.5 Random Functions

[16]:
```
my_np_array = np.random.randint(low=0, high=10, size=(3, 3))
array_info(my_np_array)
```

```
ndim: 2
shape: (3, 3)
size: 9
dtype: int64
values:
[[6 3 0]
 [4 0 1]
 [3 4 0]]
```

[17]:
```
my_np_array = np.random.random(size=(3, 3))
array_info(my_np_array)
```

```
ndim: 2
shape: (3, 3)
size: 9
dtype: float64
values:
[[0.98289302 0.76638992 0.75550304]
 [0.57653147 0.8354745  0.90016072]
 [0.92805314 0.98848567 0.90530133]]
```