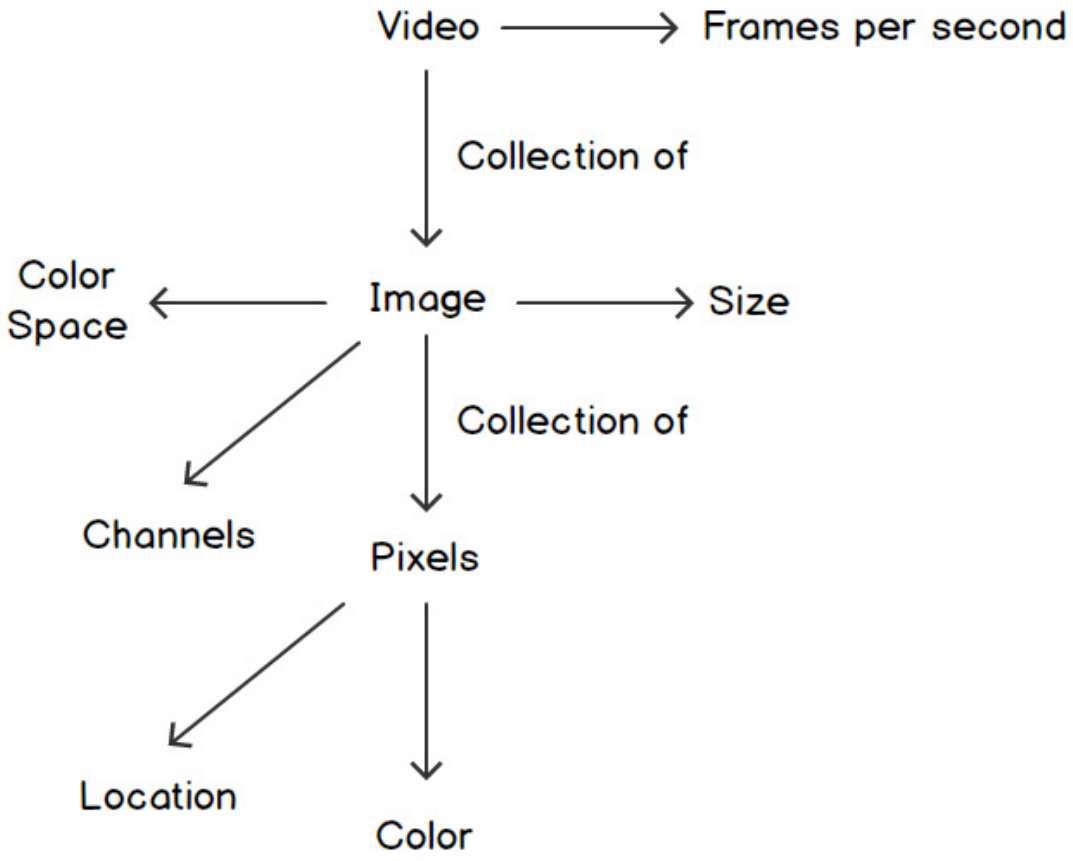
实验：创建 NumPy 数组

实验概要

图像 （Image）由称为 像素 （Pixel）的较小成分组成。 视频 （Video）由多个 帧 （Frame）组成，每个帧都不过是一幅图像。下图使我们对视频，图像和像素的各个组成部分有一个概要性了解：



实验目标

在本实验中，我们将获得一些使用各种 NumPy 函数的实际经验，这些函数用于创建 NumPy 数组并获取它们的形状。我们将使用 NumPy 的 0、1 和 rand 函数来创建数组。我们还将查看它们的数据类型和形状。

1. 导入 NumPy 模块

深入了解图像和像素的细节之前，让我们先了解一下先决条件，从 NumPy 数组开始。这背后的原因是 OpenCV 中的图像只是 NumPy 数组。简单回顾一下， NumPy 是一个用于数值计算的 Python 模块（库），以其高速计算而闻名。

首先，我们需要使用

```
import NumPy as np
```

命令导入 NumPy 模块。这里， np 被用作别名。这意味我们可以使用 np.function_name -> 替代 numpy.function_name 。

我们来看看创建 NumPy 数组的四种方法:

- 全 0 填充数组： `np.zeros`
- 全 1 填充数组： `np.ones`
- 随机数填充数组： `np.random.rand`
- 指定值填充数组： `np.array`

我们先看看 `np.zeros` 和 `np.ones` 命令，函数有两个重要的参数：

- 数组的形状。对于一个二维数组，这就是(行数，列数)。
- 元素的数据类型。默认情况下，NumPy 使用 `浮点数` 作为其数据类型。对于图像，我们将使用不带正负号的 8 位整数（unsigned 8-bit integers） `np.uint8` 。这背后的原因是 unsigned 8-bit integers 的范围是 0 到 255，这与像素值的范围相同。

譬如：我们要创建一个大小是 `4x3` 的全 `0` 填充数组，我们可以通过使用 `np.zeros(4,3)` 来实现。类似地，如果我们想创建一个全 `1` 填充的 `4x3` 数组，我们可以使用 `np.ones(4,3)` 。

对于 `np.random.rand` 函数而言，只需要确定数组的形状，对于二维数据，实现方式就是 `np.random.rand(number_of_rows, number_of_columns)`

对于 `np.array` 我们提供数据作为第一个参数，数据类型作为第二个参数。

有了 NumPy 数组之后，就可以使用 `npArray.shape` 查找数组的形状，其中 `npArray` 是 NumPy 数组的名称。

我们也可以使用 `npArray.dtype` 显示数组中元素的数据类型。

In [1]:

```
import numpy as np # 导入NumPy模块
```

2. 创建一个 5 行 6 列全 0 填充的数组

In [2]:

```
# 创建一个5行6列的2D NumPy数组，用零填充
npArray = np.zeros((5,6))
```

In [3]:

```
print(npArray) # 输出刚刚创建的数组
```

```
[[0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]]
```

In [4]:

```
print(npArray.dtype) # 输出数组元素的数据类型
```

```
float64
```

In [5]:

```
print(npArray.shape) # 输出数组的形状
```

```
(5, 6)
```

In [6]:

```
# 输出数组中的行数
print("数组中的行数(row) = {}".format(npArray.shape[0]))
# 输出数组中的列数
print("数组中的列数(columns) = {}".format(npArray.shape[1]))
```

```
数组中的行数(row) = 5
数组中的列数(columns) = 6
```

3. 指定数组数据类型

注意，我们刚刚创建的数组使用浮点数作为数据类型。 让我们用另一个数据类型 —— 不带正负号的 8 位整数（unsigned 8-bit integers），创建一个新数组，并找出其数据类型和形状。

In [7]:

```
# 指定数组数据类型为np.uint8
npArray = np.zeros((5,6), dtype=np.uint8)
```

In [8]:

```
print(npArray) # 输出刚刚创建的数组
```

```
[[0 0 0 0 0 0]
 [0 0 0 0 0 0]
 [0 0 0 0 0 0]
 [0 0 0 0 0 0]
 [0 0 0 0 0 0]]
```

In [9]:

```
print(npArray.dtype) # 输出数组元素的数据类型
```

```
uint8
```

In [10]:

```
print(npArray.shape) # 输出数组的形状
```

```
(5, 6)
```

In [11]:

```
# 输出数组中的行数
print("数组中的行数(row) = {}".format(npArray.shape[0]))
# 输出数组中的列数
print("数组中的列数(columns) = {}".format(npArray.shape[1]))
```

```
数组中的行数(row) = 5
数组中的列数(columns) = 6
```

4. 创建一个 5 行 6 列全 1 填充的数组

现在，我们将使用前面看到的其他命令来创建大小相同的数组，即（5,6），并且具有相同的数据类型（ np.uint8 ）。 让我们创建一个全 1 填充的数组。

In [12]:

```
# 创建一个全1填充的数组，形状大小为（5,6），数据类型为（np.uint8）
npArray = np.ones((5,6), dtype=np.uint8)

print(npArray) # 输出刚刚创建的数组
```

```
[[1 1 1 1 1 1]
 [1 1 1 1 1 1]
 [1 1 1 1 1 1]
 [1 1 1 1 1 1]
 [1 1 1 1 1 1]]
```

In [13]:

```
print(npArray.dtype) # 输出数组元素的数据类型
```

```
uint8
```

In [14]:

```
print(npArray.shape) # 输出数组的形状
```

```
(5, 6)
```

In [15]:

```
# 输出数组中的行数
print("数组中的行数(row) = {}".format(npArray.shape[0]))
# 输出数组中的列数
print("数组中的列数(columns) = {}".format(npArray.shape[1]))
```

```
数组中的行数(row) = 5
数组中的列数(columns) = 6
```

5. 创建随机数填充数组

我们将创建一个用随机数填充的数组。 请注意，在构建由随机数填充的数组时，我们无法指定数据类型。

In [16]:

```
# 创建一个随机数填充的数组，形状大小同样是（5,6）
npArray = np.random.rand(5,6)
```

In [17]:

```
print(npArray) # 输出刚刚创建的数组
```

```
[[0.76493029 0.4244641  0.96668388 0.87860886 0.00654919 0.46697776]
 [0.20844501 0.40487349 0.51814253 0.06393495 0.21362829 0.64340518]
 [0.8480972  0.65643116 0.08282768 0.09566009 0.90510305 0.74211234]
 [0.64052005 0.85885759 0.48681204 0.73636393 0.65345601 0.2980164 ]
 [0.37037381 0.28148836 0.15755969 0.32690712 0.56888984 0.16034325]]
```

In [18]:

```
print(npArray.dtype) # 输出数组元素的数据类型
print(npArray.shape) # 输出数组的形状
```

```
float64
(5, 6)
```

In [19]:

```
# 输出数组中的行数
print("数组中的行数(row) = {}".format(npArray.shape[0]))
# 输出数组中的列数
print("数组中的列数(columns) = {}".format(npArray.shape[1]))
```

```
数组中的行数(row) = 5
数组中的列数(columns) = 6
```

6. 创建指定值填充数组

最后，让我们创建一个如下图所示的数组：

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30

创建和打印数组的代码如下：

In [20]:

```
# 创建指定值填充数组
npArray = np.array([[1,2,3,4,5,6],
                    [7,8,9,10,11,12],
                    [13,14,15,16,17,18],
                    [19,20,21,22,23,24],
                    [25,26,27,28,29,30]],
                   dtype=np.uint8)

print(npArray) # 输出刚刚创建的数组
```

```
[[ 1  2  3  4  5  6]
 [ 7  8  9 10 11 12]
 [13 14 15 16 17 18]
 [19 20 21 22 23 24]
 [25 26 27 28 29 30]]
```

In [21]:

```
print(npArray.dtype) # 输出数组元素的数据类型
```

```
uint8
```

In [22]:

```
print(npArray.shape) # 输出数组的形状
```

```
(5, 6)
```

In [23]:

```
# 输出数组中的行数
print("数组中的行数(row) = {}".format(npArray.shape[0]))
# 输出数组中的列数
print("数组中的列数(columns) = {}".format(npArray.shape[1]))
```

```
数组中的行数(row) = 5
数组中的列数(columns) = 6
```

实验小结

在本实验中，我们看到了如何使用不同的函数创建 NumPy 数组，如何指定其数据类型以及如何显示其形状。