список целочисленных значений

In [13]:

```
L = list(range(10))
L
```

Out[13]:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

список строковых значений

In [20]:

```
L = [str(i) for i in range(10)]
L
```

Out[20]:

```
['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
```

список неоднородных значений

In [21]:

```
L3 = [True, "2", 3.0, 4, []]
[type(item) for item in L3]
```

Out[21]:

```
[bool, str, float, int, list]
```

Создание массивов

In [22]:

```
import array
L = list(range(10))
A = array.array('i', L)
A
```

Out[22]:

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

In [27]:

```
import numpy as np

np.array([1, 4, 2, 5, 3])
#np.array([3.14, 4, 2, 3])
#np.array([1, 2, 3, 4], dtype='float32')
#np.array([[1.5, 2, 3], [4, 5, 6]])
```

Out[27]:

```
array(['3.14', '4', '2', '3'], dtype='<U32')
```

Другие методы создания массивов

In [30]:

```
print(np.zeros((3, 5)))
print(np.ones((2, 2, 2)))
print(np.eye(5))
print(np.empty((3, 3)))
```

```
[[0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]]
[[[1. 1.]
  [1. 1.]]

 [[1. 1.]
  [1. 1.]]]
[[1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1.]]
[[0.00000000e+000 0.00000000e+000 0.00000000e+000]
 [0.00000000e+000 0.00000000e+000 6.20546451e-321]
 [4.10463701e-288 2.29812144e-311 0.00000000e+000]]
```

Создания последовательностей чисел

In [9]:

```
np.arange(10, 30, 5)
np.arange(0, 1, 0.1)
np.linspace(0, 2, 9)  # 9 чисел от 0 до 2 включительно
```

Out[9]:

```
array([0.  , 0.25, 0.5 , 0.75, 1.  , 1.25, 1.5 , 1.75, 2.  ])
```

Атрибуты массивов

In [10]:

```
import numpy as np
np.random.seed(0) # начальное значение для целей воспроизводимости
x1 = np.random.randint(10, size=6) # одномерный массив
x2 = np.random.randint(10, size=(3, 4)) # двумерный массив
x3 = np.random.randint(10, size=(3, 4, 5)) # трехмерный массив

print("x3 ndim: ", x3.ndim)
print("x3 shape:", x3.shape)
print("x3 size: ", x3.size)
print("dtype:", x3.dtype)
```

```
x3 ndim:  3
x3 shape: (3, 4, 5)
x3 size:  60
dtype: int32
```

In [11]:

```
x = np.arange(10)
x
```

Out[11]:

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

In [ ]:

```
x1
x1[0]
x1[4]
x1[-1]
x1[-2]
```

In [ ]:

```
x2
x1[0, 0]
x1[2, 0]
x1[2, -1]
x2[0, 0] = 12
x2
x1[0] =3.14159
x2
```

In [ ]:

```
x[:5] # первые пять элементов
x[5:] # элементы после индекса 5
x[::2] # каждый второй элемент
x[1::2] # каждый второй элемент, начиная с индекса 1
x[::-1] # все элементы в обратном порядке
```

In [ ]:

```
X2
x2[:2, :3] # две строки, три столбца
x2[:3, ::2] # все строки, каждый второй столбец
x2[::-1, ::-1] # подмассивы можно инвертировать
```

Представления и копии подмассивов

In [ ]:

```
print(x2)
x2_sub = x2[:2, :2]
print(x2_sub)
x2_sub[0, 0] = 99
Print(x2_sub)
print(x2)
```

In [ ]:

```
print(x2)
x2_sub_copy = x2[:2, :2].copy()
print(x2_sub_copy)
x2_sub_copy[0, 0] = 42
print(x2_sub_copy)
print(x2)
```

Изменение формы массивов

In [ ]:

```
grid = np.arange(1, 10).reshape((3, 3))
print(grid) #поместить числа от 1 до 9 в таблицу 3 × 3
x = np.array([1, 2, 3])
Преобразование в вектор-строку с помощью reshape
x.reshape((1, 3))
# Преобразование в вектор-строку посредством newaxis
x[np.newaxis, :]
# Преобразование в вектор-столбец с помощью reshape
x.reshape((3, 1))
# Преобразование в вектор-столбец посредством newaxis
x[:, np.newaxis]
```

Слияние массивов

In [ ]:

```
x = np.array([1, 2, 3])
y = np.array([3, 2, 1])
np.concatenate([x, y])
z = [99, 99, 99]
print(np.concatenate([x, y, z]))
grid = np.array(        [[1, 2, 3],
                         [4, 5, 6]])
# слияние по первой оси координат
np.concatenate([grid, grid])
# слияние по первой оси координат
np.concatenate([grid, grid])
```

In [33]:

```
x = np.array([1, 2, 3])
grid = np.array([[9, 8, 7],[6, 5, 4]])
# Объединяет массивы по вертикали
np.vstack([x, grid])
# Объединяет массивы по горизонтали
y = np.array([[99], [99]])
np.hstack([grid, y])
```

Out[33]:

```
array([[ 9,  8,  7, 99],
       [ 6,  5,  4, 99]])
```

Сравнение скорости выполнения циклов в Python и функций в Numpy

In [31]:

```
import numpy as np
np.random.seed(0)
def compute_reciprocals(values):
    output = np.empty(len(values))
    for i in range(len(values)):
        output[i] = 1.0 / values[i]
    return output
values = np.random.randint(1, 10, size=5)
compute_reciprocals(values)
```

Out[31]:

```
array([0.16666667, 1.        , 0.25      , 0.25      , 0.125     ])
```

In [32]:

```
big_array = np.random.randint(1, 100, size=1000000)
%timeit compute_reciprocals(big_array)
```

```
1.79 s ± 11.1 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

In [33]:

```
%timeit (1.0 / big_array)
```

```
3.93 ms ± 101 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

In [ ]:

```
```

Средний рост президентов США

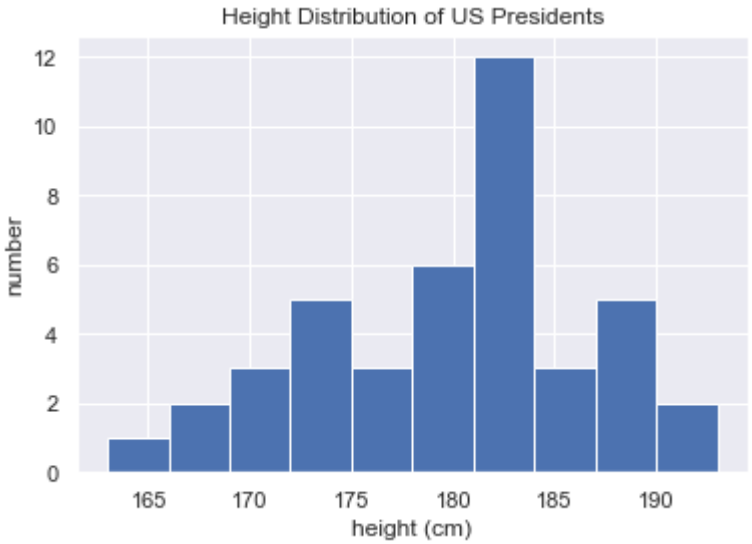
In [34]:

```
import pandas as pd
data = pd.read_csv('president_heights.csv', sep=';')
heights = np.array(data['height(cm)'])
print(heights)
print("Mean height: ", heights.mean())
print("Standard deviation:", heights.std())
print("Minimum height: ", heights.min())
print("Maximum height: ", heights.max())
print("25th percentile: ", np.percentile(heights, 25))
print("Median: ", np.median(heights))
print("75th percentile: ", np.percentile(heights, 75))
```

```
[189 170 189 163 183 171 185 168 173 183 173 173 175 178 183 193 178 173
 174 183 183 168 170 178 182 180 183 178 182 188 175 179 183 193 182 183
 177 185 188 188 182 185]
Mean height:  179.73809523809524
Standard deviation: 6.931843442745892
Minimum height:  163
Maximum height:  193
25th percentile:  174.25
Median:  182.0
75th percentile:  183.0
```

In [37]:

```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn; seaborn.set() # задает стиль графика
plt.hist(heights)
plt.title('Height Distribution of US Presidents') # Распределение роста
plt.xlabel('height (cm)') # Рост, см
plt.ylabel('number'); # Количество
```



In [ ]: