# 机器学习必备！

# 精选 Python 代码示例集

# 精选 Jupyter Notebook 快捷键

在单元格左侧单击鼠标变为命令模式（单元格左侧变为蓝色）。在单元格内侧单击变为编辑模式（单元格左侧变为绿色）。

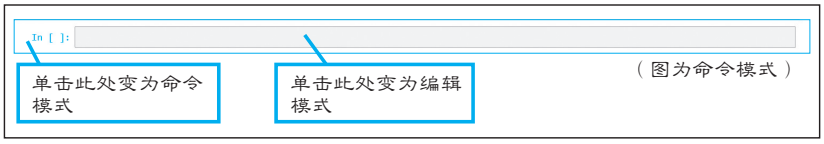



表 1　命令模式

| 键 | 说明 |
|---|---|
| [h] | 显示快捷键列表 |
| [a] | 在上方插入单元格 |
| [b] | 在下方插入单元格 |
| [x] | 删除选中的单元格 |
| [j] | 移动到下方单元格 |
| [k] | 移动到上方单元格 |
| [l] | 显示行号 |
| [y] | 变为代码模式（表达式输入模式） |
| [m] | 变为标记模式（Markdown 笔记模式） |
| [1],[2],...,[6] | 变更 Markdown 笔记模式的标题的层级 |
| [enter] | 进入编辑模式 |

表 2　编辑模式

| 键 | 说明 |
|---|---|
| [ctrl] + [enter] | 运行 |
| [shift] + [enter] | 运行，进入下方单元格，如果下方没有单元格，则在下方增加一个单元格 |
| [ctrl] + [z] | 恢复到之前的状态（undo） |
| [ctrl] + [y] | 恢复到 undo 之前的状态（redo） |
| [ctrl] + [ ] ] | 增加选中行的缩进 |
| [ctrl] + [ [ ] | 减少选中行的缩进 |
| [esc] | 进入命令模式 |

# 机器学习必备的精选 Python 代码示例集

首先将 numpy 和 matplotlib.pyplot 这两个库 import 进来。

## print 语句

以下是 print 语句的代码示例。

**In**

```
a = 1 / 3
b = 2 / 7
print(a, b)                                    # 显示变量值
```

**Out**

```
0.3333333333333333 0.2857142857142857
```

**In**

```
print(np.round(a, 2))                          # 显示到小数点后第二位为止（四舍五入）
```

**Out**

```
0.33
```

**In**

```
print('a = ' + str(a))                         # 与字符串一起显示 示例 1
```

**Out**

```
a = 0.3333333333333333
```

**In**

```
print('a = ' + str(np.round(a, 2)))            # 与字符串一起显示 示例 2
```

**Out**

```
a = 0.33
```

**In**

```
print('a = {0:.2f}, b = {1} '.format(a, b))    # 显示为指定格式
```

**Out**

```
a = 0.33, b = 0.2857142857142857
```

## 列表、向量、矩阵

以下是列表、向量、矩阵的代码示例。

**In**

```
a = [1, 2, 3]                                    # list
len(a)                                           # list 的长度
```

**Out**

```
3
```

**In**

```
a = range(5)                                     # range
list(a)                                          # 将 range 转换为 list
```

**Out**

```
[0, 1, 2, 3, 4]
```

**In**

```
a = np.array([1, 2, 3])                          # 向量（一维 ndarray）
print(a)
```

**Out**

```
[1 2 3]
```

**In**

```
a = np.array([[1, 2, 3], [4, 5, 6]])            # 矩阵（二维 ndarray）
print(a)
a.shape                                          # 矩阵的大小
```

**Out**

```
[[1 2 3]
 [4 5 6]]
(2, 3)
```

**In**

```
a = np.arange(10)                        # 元素为从 0 到 9 的向量
print(a)
```

**Out**

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

**In**

```
a = np.linspace(0, 1, 10)       # 元素为 0 和 1 之间的 10 个等间距值的向量
print(a)
```

**Out**

```
[ 0.          0.11111111  0.22222222  0.33333333  0.44444444  0.55555556
  0.66666667  0.77777778  0.88888889  1.        ]
```

**In**

```
a = np.zeros(5)                    # 元素均为 0 的向量
print(a)
```

**Out**

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

**In**

```
a = np.ones((2, 3))                # 元素均为 1 的矩阵
print(a)
```

**Out**

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

**In**

```
a = np.random.random(3)            # 元素均为 0 和 1 之间的随机数的向量
print(a)
```

**Out**

```
[ 0.22308748  0.46468634  0.30009322]
```

**In**

```
a = np.arange(1, 7)                # [1 2 3 4 5 6]
print(a.reshape(2, 3))             # 转换为 2 行 3 列的矩阵
```

**Out**

```
[[1 2 3]
 [4 5 6]]
```

**In**

```
a = np.array([[1, 2], [3, 4]])
b = np.array([[5, 6], [7, 8]])
print(np.c_[a, b])                 # 按行连接两个矩阵
print(np.r_[a, b])                 # 按列连接两个矩阵
```

**Out**

```
[[1 2 5 6]
 [3 4 7 8]]
[[1 2]
 [3 4]
 [5 6]
 [7 8]]
```

## 矩阵的运算

以下是矩阵运算的代码示例。

In
```
x = np.arange(1, 6) / 4                 # [ 0.25  0.5   0.75  1.    1.25]
np.sqrt(x)                              # 平方根
np.log(x)                               # 对数
np.round(x, 1)                          # 四舍五入，保留到小数点后第 1 位为止
np.mean(x)                              # 平均
np.std(x)                               # 标准差
np.max(x)                               # 最大值
np.min(x)                               # 最小值
```

Out
```
0.25      (# 上面单元格最后一行 np.min(x) 的结果 )
```

In
```
x = np.array([[1, 2], [3, 4], [5, 6]])
w = np.array([[1, 2], [3, 4]])
x.dot(w)                                # 矩阵的内积
```

Out
```
array([[ 7, 10],
       [15, 22],
       [23, 34]])
```

In
```
x = np.arange(10)                       # [0 1 2 3 4 5 6 7 8 9]
x[:5]                                   # 切片 从开始到 4
```

Out
```
array([0, 1, 2, 3, 4])
```

In
```
x[5:]                                   # 切片 从 5 到最后
```

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

In
```
x[3:8]                                  # 切片 从 3 到 7
```

Out
```
array([3, 4, 5, 6, 7])
```

## 控制语句

以下是控制语句的代码示例。

In
```
for i in range(3):                                    # for 语句
    print(i)
```

Out
```
0
1
2
```

In
```
a = 100
if a > 10:                                            # if 语句
   print('a is larger than 10')
else:
   print('a is not larger than 10')
```

Out
```
a is larger than 10
```

## 函数

以下是函数的代码示例。

In
```
def my_func(a, b):                                    # 定义函数
    c = a + b
    return c
```

In
```
my_func(1, 2)                                         # 运行函数
```

Out
```
3
```

## 数据的保存和读取

以下是数据的保存和读取的代码示例。

In
```
data1 = np.array([1, 2, 3])
data2 = np.array([10, 20, 30])
np.savez('datafile.npz', data1 = data1, data2 = data2)    # 数据的保存
```

**In**

```
outfile  =  np.load('datafile.npz')          # 数据的读取
print(outfile.files)                         # 显示所有保存的数据名
data1  =  outfile['data1']                   # 取出 data1
data2  =  outfile['data2']                   # 取出 data2
print(data1)
print(data2)
```

**Out**

```
['data1', 'data2']
[1 2 3]
[10 20 30]
```

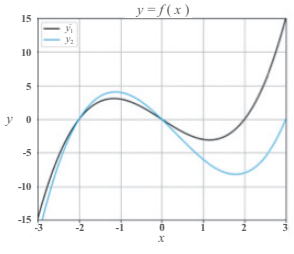
## 绘制 $y(x)$ 的图形

以下是绘制 $y(x)$ 图形的代码示例。

**In**

```
x  =  np.linspace(-3, 3, 100)                # 定义 x
y1  =  (x + 2) * x * (x - 2)                 # 定义 y1
y2  =  (x + 2) * x * (x - 3)                 # 定义 y2
plt.figure(figsize = (6, 5))                 # 创建 figure
plt.plot(x, y1, color = 'black', label = 'y1')  # 以黑色绘制 x - y1 的图形
plt.plot(x, y2, color = 'cornflowerblue', label = 'y2') # 以蓝色绘制 x - y2 的图形
plt.legend(loc = "upper left")               # 在左上角显示图例
plt.ylim(-15, 15)                            # 指定 y 的范围
plt.xlim(-3, 3)                              # 指定 x 的范围
plt.title('y = f(x)')                        # 显示标题
plt.xlabel('x')                              # 显示横轴标签
plt.ylabel('y')                              # 显示纵轴标签
plt.grid(True)                               # 显示网格
plt.show()                                   # 显示图形
```

**Out**



## 绘制 $y(x_0, x_1)$ 的图形

以下是绘制 $y(x_0, x_1)$ 图形的代码示例。

**In**

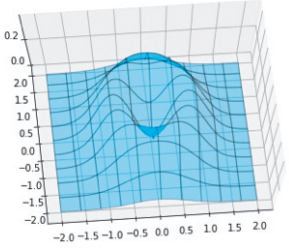
```
from mpl_toolkits.mplot3d import Axes3D      # 导入 Axes3D

def f(x0, x1):                               # 定义函数
    r  =  2 * x0**2 + x1**2
    ans  =  r * np.exp(-r)
    return ans

xn  =  50                                    # 图形的粗细
x0  =  np.linspace(-2, 2, xn)                # x0 的范围
x1  =  np.linspace(-2, 2, xn)                # x1 的范围
y  =  np.zeros((xn, xn))                     # y 的准备工作
for i0 in range(xn):
    for i1 in range(xn):
        y[i1, i0]  =  f(x0[i0], x1[i1])      # 计算 y
xx0, xx1  =  np.meshgrid(x0, x1)

plt.figure(figsize = (8, 6))                   # 创建 figure
ax  =  plt.subplot(1, 1, 1, projection = '3d') # 3D 图形显示的准备工作
ax.plot_surface(xx0, xx1, y, rstride = 5, cstride = 5, alpha = 0.3,
                color = 'blue', edgecolor = 'black')  # 显示 3D 图形
ax.set_zticks((0, 0.2))                      # 指定 z 轴的刻度
ax.view_init(75, -95)                        # 指定图形的显示角度
plt.show()                                   # 显示图形
```

**Out**




## 绘制 $y(x_0, x_1)$ 图形的等高线

可以在绘制 $y(x_0, x_1)$ 的图形的代码之后，输入以下代码，用于绘制等

高线。

**In**

```
plt.figure(1, figsize = (4, 4))                                  # figure 的准备工作
cont  =  plt.contour(xx0, xx1, y, 5, colors = 'black')  # 绘制等高线图形
cont.clabel(fmt = '%3.2f', fontsize = 8)                  # 显示等高线的高度
plt.xlabel('$x_0$', fontsize = 14)                                # 显示横轴标签
plt.ylabel('$x_1$', fontsize = 14)                                # 显示纵轴标签
plt.show()                                                                  # 显示图形
```

**Out**