Python计算生态构建

嵩 天

实例1：矩阵乘法模块的构建

嵩 天

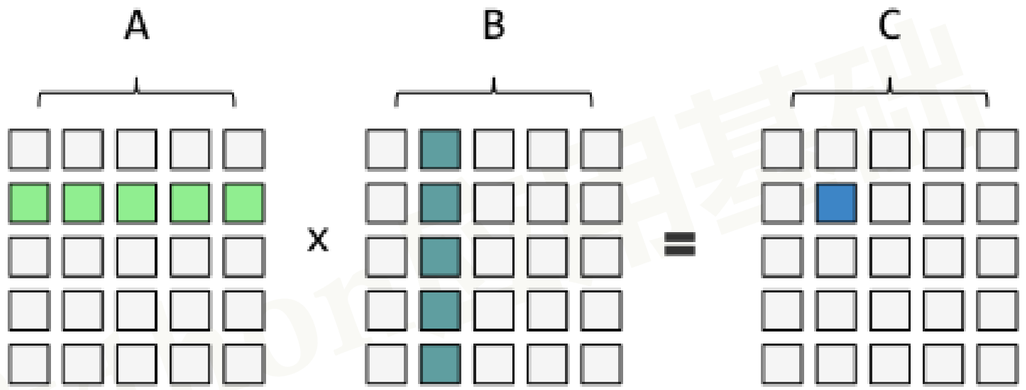
Python计算生态构建

"矩阵乘法模块的构建"需求分析

Python123

需求分析

矩阵乘法



A(nrow, nk)

B(nk, ncol)

C(nrow, ncol)

```
C[i][j] = sum(A[i][k] * B[k][j]) for k = 0 ... n
```

需求分析

矩阵乘法

- 人工智能算法的核心运算之一
- 数据分析的数据表达及运算之一
- 大规模科学计算的核心操作之一

需求分析

矩阵乘法模块

- 输入：两个矩阵及行列值
- 处理：矩阵乘法运算、矩阵各元素求和运算
- 输出：一个采用Python语言开发的Python模块

Python计算生态构建

"矩阵乘法模块的构建"实例解析

Python123

代码纵览

```python
def mxmul(mx1, mx2, nrow, nk, ncol):
    rst = [[0 for y in range(ncol)] for x in range(nrow)]
    for i in range(nrow):
        for j in range(ncol):
            for k in range(nk):
                rst[i][j] += mx1[i][k] * mx2[k][j]
    return rst

def mxsum(mx, nrow, ncol):
    s = 0
    for i in range(nrow):
        for j in range(ncol):
            s += mx[i][j]
    return s

if __name__ == "__main__":
    import time
    nrow, nk, ncol = 500, 300, 500
    mx1 = [[y for y in range(nk)] for x in range(nrow)]
    mx2 = [[y for y in range(ncol)] for x in range(nk)]
    start = time.perf_counter()
    rst = mxmul(mx1, mx2, nrow, nk, ncol)
    end =time.perf_counter()
    print("运算时间为{:.4f}s".format(end-start))
```

mxmul.py

模块的编写

代码详解

```
def mxmul(mx1, mx2, nrow, nk, ncol):
    rst = [[0 for y in range(ncol)] for x in range(nrow)]
    for i in range(nrow):
        for j in range(ncol):
            for k in range(nk):
                rst[i][j] += mx1[i][k] * mx2[k][j]
    return rst

def mxsum(mx, nrow, ncol):
    s = 0
    for i in range(nrow):
        for j in range(ncol):
            s += mx[i][j]
    return s

if __name__ == "__main__":
    import time
    nrow, nk, ncol = 500, 300, 500
    mx1 = [[y for y in range(nk)] for x in range(nrow)]
    mx2 = [[y for y in range(ncol)] for x in range(nk)]
    start = time.perf_counter()
    rst = mxmul(mx1, mx2, nrow, nk, ncol)
    end =time.perf_counter()
    print("运算时间为{:.4f}s".format(end-start))
```

函数

函数

__name__

代码详解

```
def mxmul(mx1, mx2, nrow, nk, ncol):
    rst = [[0 for y in range(ncol)] for x in range(nrow)]
    for i in range(nrow):
        for j in range(ncol):
            for k in range(nk):
                rst[i][j] += mx1[i][k] * mx2[k][j]
    return rst

def mxsum(mx, nrow, ncol):
    s = 0
    for i in range(nrow):
        for j in range(ncol):
            s += mx[i][j]
    return s

if __name__ == "__main__":
    import time
    nrow, nk, ncol = 500, 300, 500
    mx1 = [[y for y in range(nk)] for x in range(nrow)]
    mx2 = [[y for y in range(ncol)] for x in range(nk)]
    start = time.perf_counter()
    rst = mxmul(mx1, mx2, nrow, nk, ncol)
    end =time.perf_counter()
    print("运算时间为{:.4f}s".format(end-start))
```


矩阵乘，返回一个新的矩阵

即返回一个新的列表对象

代码详解

```
def mxmul(mx1, mx2, nrow, nk, ncol):
    rst = [[0 for y in range(ncol)] for x in range(nrow)]
    for i in range(nrow):
        for j in range(ncol):
            for k in range(nk):
                rst[i][j] += mx1[i][k] * mx2[k][j]
    return rst

def mxsum(mx, nrow, ncol):
    s = 0
    for i in range(nrow):
        for j in range(ncol):
            s += mx[i][j]
    return s

if __name__ == "__main__":
    import time
    nrow, nk, ncol = 500, 300, 500
    mx1 = [[y for y in range(nk)] for x in range(nrow)]
    mx2 = [[y for y in range(ncol)] for x in range(nk)]
    start = time.perf_counter()
    rst = mxmul(mx1, mx2, nrow, nk, ncol)
    end =time.perf_counter()
    print("运算时间为{:.4f}s".format(end-start))
```

矩阵累加和

返回一个新的数值对象

代码详解

```
def mxmul(mx1, mx2, nrow, nk, ncol):
    rst = [[0 for y in range(ncol)] for x in range(nrow)]
    for i in range(nrow):
        for j in range(ncol):
            for k in range(nk):
                rst[i][j] += mx1[i][k] * mx2[k][j]
    return rst

def mxsum(mx, nrow, ncol):
    s = 0
    for i in range(nrow):
        for j in range(ncol):
            s += mx[i][j]
    return s

if __name__ == "__main__":
    import time
    nrow, nk, ncol = 500, 300, 500
    mx1 = [[y for y in range(nk)] for x in range(nrow)]
    mx2 = [[y for y in range(ncol)] for x in range(nk)]
    start = time.perf_counter()
    rst = mxmul(mx1, mx2, nrow, nk, ncol)
    end =time.perf_counter()
    print("运算时间为{:.4f}s".format(end-start))
```

测试部分

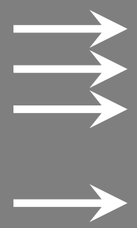
统计并输出运行时间

代码详解

```
def mxmul(mx1, mx2, nrow, nk, ncol):
    rst = [[0 for y in range(ncol)] for x in range(nrow)]
    for i in range(nrow):
        for j in range(ncol):
            for k in range(nk):
                rst[i][j] += mx1[i][k] * mx2[k][j]
    return rst

def mxsum(mx, nrow, ncol):
    s = 0
    for i in range(nrow):
        for j in range(ncol):
            s += mx[i][j]
    return s

if __name__ == "__main__":
    import time
    nrow, nk, ncol = 500, 300, 500
    mx1 = [[y for y in range(nk)] for x in range(nrow)]
    mx2 = [[y for y in range(ncol)] for x in range(nk)]
    start = time.perf_counter()
    rst = mxmul(mx1, mx2, nrow, nk, ncol)
    end =time.perf_counter()
    print("运算时间为{:.4f}s".format(end-start))
```



生成两个测试矩阵

调用函数运算结果

运行结果

nrow, nk, ncol = 50, 30, 50　　运算时间为0. 0190s

nrow, nk, ncol = 100, 60, 100　　运算时间为0. 1820s

nrow, nk, ncol = 500, 300, 500　　运算时间为19. 2137s

代码纵览


```python
import mxmul

nrow, nk, ncol = 5, 3, 5
A = [[y for y in range(nk)] for x in range(nrow)]
B = [[y for y in range(ncol)] for x in range(nk)]
C = mxmul.mxmul(A, B, nrow, nk, ncol)
print(C)
```

calmul.py

模块的使用

```
[[0, 3, 6, 9, 12], [0, 3, 6, 9, 12],
 [0, 3, 6, 9, 12], [0, 3, 6, 9, 12],
 [0, 3, 6, 9, 12]]
```


Python 123
Thank you