**Numpy作业**

1. **依照课件3中的内容给出测算三角波（triangle\_wave（））y1、y2、y3、y4四种方式的计算速度与结果比较的代码，并对其运算显示结果。**

**Code:**

from time import perf\_counter

import numpy as np

x = np.linspace(0, 2, 1000)

# Method 1

def triangle\_wave(x, c, c0, hc):

x = x - int(x) # 三角波的周期为1，因此只取x坐标的小数部分进行计算

if x >= c:

r = 0.0

elif x < c0:

r = x / c0 \* hc

else:

r = (c - x) / (c - c0) \* hc

return r

start = perf\_counter()

y1 = np.array([triangle\_wave(t, 0.6, 0.4, 1.0) for t in x])

time1 = perf\_counter()

# Method 2

triangle\_ufunc1 = np.frompyfunc(triangle\_wave, 4, 1)

y2 = triangle\_ufunc1(x, 0.6, 0.4, 1.0)

y2 = y2.astype(np.float)

time2 = perf\_counter()

# Method 3

triangle\_ufunc2 = np.frompyfunc(lambda x: triangle\_wave(x, 0.6, 0.4, 1.0), 1, 1)

y3 = triangle\_ufunc2(x)

y3 = y3.astype(np.float)

time3 = perf\_counter()

# Method 4

triangle\_ufunc3 = np.vectorize(triangle\_wave, otypes=[np.float])

y4 = triangle\_ufunc3(x, 0.6, 0.4, 1.0)

time4 = perf\_counter()

print("y1与y2:", np.all(y1 == y2))

print("y2与y3:", np.all(y2 == y3))

print("y3与y4:", np.all(y3 == y4))

print("time of y1:", time1 - start)

print("time of y2:", time2 - time1)

print("time of y3:", time3 - time2)

print("time of y4:", time4 - time3)

**Result:**

**y1与y2: True**

**y2与y3: True**

**y3与y4: True**

**time of y1: 0.00023108299956220435**

**time of y2: 0.0002329580001969589**

**time of y3: 0.00019433399984336575**

**time of y4: 0.00017283300030612736**

**2、arr11 = 5-np.arange(1,13).reshape(4,3), 计算所有元素及每一列的和；对每一个元素、每一列求累积和；计算每一行的累计积；计算所有元素的最小值；计算每一列的最大值；计算所有元素、每一行的均值；计算所有元素、每一列的中位数；计算所有元素的方差，每一行的标准差。**

**Code:**

import numpy as np

arr11 = 5-np.arange(1,13).reshape(4,3)

print("array:",arr11)

print("-------------"\*3)

print("\n")

# 所有元素的和

print("所有元素的和:",np.sum(arr11))

# 计算每一列的和

print("计算每一列的和:",np.sum(arr11,axis=0))

# 对每一个元素求累积和

print("对每一个元素求累积和:",np.cumsum(arr11))

# 对每一列求累积和

print("对每一列求累积和:",np.cumsum(arr11,axis=0))

# 计算每一行的累计积

print("计算每一行的累计积:",np.cumprod(arr11,axis=1))

# 计算所有元素的最小值

print("计算所有元素的最小值:",np.min(arr11))

# 计算每一列的最大值

print("计算每一列的最大值:",np.max(arr11,axis=0))

# 计算所有元素的均值

print("计算所有元素的均值:",np.mean(arr11))

# 计算每一行的均值

print("计算每一行的均值:",np.mean(arr11,axis=1))

# 计算所有元素的中位数

print("计算所有元素的中位数:",np.median(arr11))

# 计算每一列的中位数

print("计算每一列的中位数:",np.median(arr11,axis=0))

# 计算所有元素的方差

print("计算所有元素的方差:",np.var(arr11))

# 计算所有元素的方差，每一行的标准差

print("计算所有元素的方差，每一行的标准差:",np.std(arr11,axis=1))

**Result:**

**array: [[ 4 3 2]**

**[ 1 0 -1]**

**[-2 -3 -4]**

**[-5 -6 -7]]**

**---------------------------------------**

**所有元素的和: -18**

**计算每一列的和: [ -2 -6 -10]**

**对每一个元素求累积和: [ 4 7 9 10 10 9 7 4 0 -5 -11 -18]**

**对每一列求累积和: [[ 4 3 2]**

**[ 5 3 1]**

**[ 3 0 -3]**

**[ -2 -6 -10]]**

**计算每一行的累计积: [[ 4 12 24]**

**[ 1 0 0]**

**[ -2 6 -24]**

**[ -5 30 -210]]**

**计算所有元素的最小值: -7**

**计算每一列的最大值: [4 3 2]**

**计算所有元素的均值: -1.5**

**计算每一行的均值: [ 3. 0. -3. -6.]**

**计算所有元素的中位数: -1.5**

**计算每一列的中位数: [-0.5 -1.5 -2.5]**

**计算所有元素的方差: 11.916666666666666**

**计算所有元素的方差，每一行的标准差: [0.81649658 0.81649658 0.81649658 0.81649658]**

**3、在数组[1, 2, 3, 4, 5]中每相邻两个数字中间插入两个0。**

**Code：**

import numpy as np

# 生成 1-5 的数组

array1 = np.arange(1, 6)

# 生成需要插入的 0 数组

array2 = np.zeros(5)

# 先将 0 数组堆叠到 1-5 数组下面，在进行维度变换，之后在展平

array3 = np.ravel(np.vstack((array1,array2)).swapaxes(0,1))

print(array3)

**Result：**

**[1. 0. 2. 0. 3. 0. 4. 0. 5. 0.]**

**4、归一化，将矩阵规格化到0～1，即最小的变成0，最大的变成1，最小与最大之间的等比缩放。试对Z = np.random.random((5,5))进行归一化。**

**Code:**

import numpy as np

Z = np.random.random((5,5))

range = np.max(Z) - np.min(Z)

Z1 = np.divide(Z-np.min(Z),np.max(Z))

print(Z1)

**Result:**

**[[0.86086256 0.65635864 0.4222716 0.78079641 0.92074663]**

**[0.57244005 0.52199128 0.36904469 0.14576818 0.36633288]**

**[0.63764378 0.75605212 0. 0.00450319 0.22361107]**

**[0.01177594 0.54696339 0.94498489 0.61877451 0.25088646]**

**[0.837705 0.98399876 0.31418745 0.33334667 0.57055454]]**

**5、找出数组中与给定值最接近的数（通用方法）。（例：任一数组Z=array([[0,1,2,3],[4,5,6,7]])，给任一定值z=5.1，如何找出Z中的5）**

**Code:**

*import* numpy *as* np  
*# 给定的数字*c = 5.1  
Z = np.array([[0,1,2,3],[4,5,6,7]])  
indx = np.argmin(np.abs(Z-c))  
num = np.ravel(Z)[indx]  
print(num)

**Result:**

**5**

**6、解方程：3x + 6y -5z = 12；x-3y+2z = -2；5x -y +4z = 10。**

**Code：**

*import* numpy *as* np  
  
*# 输入系数矩阵*A = np.array([[3, 6, -5], [1, -3, 2], [5, -1, 4]])  
*# 输入方程组的向量*b = np.array([12, -2, 10])  
y = np.linalg.solve(A, b)  
print(y)

**Result:**

**[1.75 1.75 0.75]**

**7、参见课件4第45页，对g(y)在100个切比雪夫节点之上分別使用Polynomial（Polynomial.fit）和Chebyshev（Chebyshev.fit）进行插值，在[-1,1]区间上取1000个等距点对误差进行比较。g(x)= sin(z\*\*2) + sin(z)\*\*2, 其中z=(x - 1) \* 5。**

**Code：**

*import* numpy *as* np  
*from* numpy.polynomial *import* Chebyshev  
*from* numpy.polynomial *import* Polynomial  
  
*def* g(x):  
 z = (x-1)\*5  
 y = np.sin(z\*\*2)+np.sin(z)\*\*2  
 *return* y  
  
  
*def* main():  
 n = 100  
 x = Chebyshev.basis(n).roots()  
 xd = np.linspace(-1,1,1000)  
  
 c1 = Chebyshev.fit(x, g(x), n-1, domain=[-1,1])  
 c2 = Polynomial.fit(x, g(x), n, domain=[-1,1])  
 print("---不同差值方法的最大误差---")  
 print("多项式差值的最大误差：", np.max(abs(g(xd)-c2(xd))))  
 print("Chebyshev差值的最大误差：", np.max(abs(g(xd)-c1(xd))))

**Result：**

**---不同差值方法的最大误差---**

**多项式差值的最大误差： 1.1937392163216494**

**Chebyshev差值的最大误差： 6.4757785800395595e-09**

**8、试用bincount()函数替代histogram()函数完成统计男青少年年龄和身高的例子的计算（数据见height.csv）**

**Code:**

*import* numpy *as* np  
*import* pandas *as* pd  
  
data = pd.read\_csv("height.csv")  
dt = np.array(data, dtype="int64")  
age = np.bincount(dt[:, 0])  
height = np.bincount(dt[:, 1])  
  
age\_dict = {}  
height\_dict ={}  
*for* i *in* range(len(age)):  
 *if* age[i] != 0:  
 age\_dict[f"{i+1}"] = age[i]  
  
*for* i *in* range(len(height)):  
 *if* height[i] != 0:  
 height\_dict[f"{i+1}"] = height[i]  
  
print("年龄统计结果：\n", age\_dict)  
print("身高统计结果：\n", height\_dict)

**Result:**

**年龄统计结果：**

**{'8': 10, '9': 6, '10': 7, '11': 9, '12': 5, '13': 10, '14': 9, '15': 3, '16': 11, '17': 6, '18': 4, '19': 11, '20': 8}**

**身高统计结果：**

**{'122': 1, '124': 2, '126': 1, '127': 3, '128': 1, '129': 1, '131': 1, '132': 3, '133': 1, '134': 1, '135': 2, '136': 1, '138': 2, '140': 1, '141': 2, '143': 2, '144': 4, '145': 1, '146': 2, '147': 1, '148': 2, '149': 2, '152': 2, '154': 4, '155': 1, '156': 2, '157': 1, '161': 2, '162': 3, '164': 2, '165': 2, '166': 1, '167': 1, '170': 4, '171': 2, '172': 2, '173': 5, '174': 8, '175': 5, '176': 9, '177': 4, '179': 2}**

**9、使用二项分布进行赌博计算. 同时抛弃5枚硬币，如果正面朝上少于3枚，则输掉8元，否则就赢8元。如果手中有1000元作为赌资，请问赌博10000次后可能会是什么情况呢？(参见课件)**

**Code:**

*import* matplotlib.pyplot *as* plt  
*import* numpy *as* np  
  
cash = np.zeros(10000)  
cash[0] = 1000  
  
outcome = np.random.binomial(5, 0.5, 10000)  
  
*for* i *in* range(1, 10000):  
 *if* outcome[i] < 3:  
 cash[i] = cash[i - 1] - 8  
 *elif* outcome[i] <= 5:  
 cash[i] = cash[i - 1] + 8  
 *else*:  
 print(np.max(outcome), np.min(outcome))  
 *raise* AssertionError("Unexpected outcome ", outcome[i])  
  
plt.plot(np.arange(len(cash)), cash)  
plt.xlabel("Times of gamble")  
plt.ylabel("Money remained")  
plt.title("Simulate Gamble")  
plt.show()  
print(cash[-1])

**Result:**

**1184.0**

****