{ "cells": [{ "cell_type": "code", "execution_count": 1, "id": "f50ed9cc-15fa-4a1b-a526-5e2702fc8a28", "metadata": {}, "outputs": [{ "name": "stdin", "output_type": "stream", "text": ["please input a: 5\n", "please input b: 8\n", "please input c: 7\n"] }, { "name": "stdout", "output_type": "stream", "text": ["-35.0\n"] }], "source": ["#flowchat\n", "str_a=input('please input a:')\n", "str_b=input('please input b:')\n", "str_c=input('please input c:')\n", "a=float(str_a)\n", "b=float(str_b)\n", "c=float(str_c)\n", "if a>b:\n", " if b>c:\n", " list=[a,b,c]\n", " else:\n", " if a>c:\n", " list=[a,c,b]\n", " else:\n", " list=[c,a,b]\n", "else:\n", " \n", " list=[c,b,a]\n", "x=list[0]\n", "y=list[1]\n", "z=list[2]\n", "rs=x+y-10*z\n", "print(rs)"] }, { "cell_type": "code", "execution_count": 5, "id": "f3496bc4-3e40-4d8e-9562-bcc82790ac2e", "metadata": {}, "outputs": [{ "name": "stdout", "output_type": "stream", "text": ["please input number,untill input 'done':\n"] }, { "name": "stdin", "output_type": "stream", "text": [" 4554\n", " 84574\n", " 965465\n", " 455685\n", " done\n"] }, { "name": "stdout", "output_type": "stream", "text": ["[17, 23, 27, 25]\n"] }], "source": ["#Continuous ceiling function\n", "import numpy as np\n", "#获取输入的元素\n", "list=[]\n", "print(\"please input number,untill input 'done':\")\n", "while True:\n", " user_input=input()\n", " if user_input=='done':\n", " break\n", " list.append(user_input)\n", "int_list=[int(x) for x in list]\n", "list2=[]#储存结果的列表\n", "#循环\n", "for i in int_list:\n", " rs=0\n", "#每迭代一次，结果加2\n", " while np.ceil(i/3)!=1:\n", " rs=rs+2\n", " i=np.ceil(i/3)\n", "#因为迭代的次数少一次，而且需要加上F（1），所以最后结果还要加3\n", " rs=rs+3\n", " list2.append(rs)\n", "print(list2)"] }, { "cell_type": "code", "execution_count": 7, "id": "4ec0aefe-27fe-4f04-b284-fec6415911ce", "metadata": {}, "outputs": [{ "name": "stdout", "output_type": "stream", "text": ["there are 1 ways to get 10\n"] }, { "ename": "KeyboardInterrupt", "evalue": "", "output_type": "error", "traceback": ["\u001b[1;31m---\u001b[0m", "\u001b[1;31mKeyboardInterrupt\u001b[0m Traceback (most recent call last)", "Cell \u001b[1;32mIn[7], line 40\u001b[0m\n\u001b[0;32m 38\u001b[0m \u001b[38;5;28;01mtry\u001b[39;00m:\n\u001b[0;32m 39\u001b[0m \u001b[38;5;28;01mwhile\u001b[39;00m \u001b[38;5;28;01mTrue\u001b[39;00m:\n\u001b[1;32m---> 40\u001b[0m out_list2\u001b[38;5;241m.\u001b[39mremove(i)\n\u001b[0;32m 41\u001b[0m \u001b[38;5;28;01mexcept\u001b[39;00m \u001b[38;5;167;01mValueError\u001b[39;00m:\n\u001b[0;32m 42\u001b[0m \u001b[38;5;28;01mpass\u001b[39;00m\n", "\u001b[1;31mKeyboardInterrupt\u001b[0m: "] }], "source": ["#Dice rolling\n", "def find_number_of_ways(x):\n", " import copy\n", " x=int(x)\n", " #通过10次循环，把所有的结果储存在列表list中\n", " list=[]\n", " for i1 in range(1,7):\n", " for i2 in range(1,7):\n", " for i3 in range(1,7):\n", " for i4 in range(1,7):\n", " for i5 in range(1,7):\n", " for i6 in range(1,7):\n", " for i7 in range(1,7):\n", " for i8 in range(1,7):\n", " for i9 in range(1,7):\n", " for i10 in range(1,7):\n", " rs=i1+i2+i3+i4+i5+i6+i7+i8+i9+i10\n", " list.append(rs)\n", " #得到列表长度\n", " l1=len(list)\n", " list2=copy.copy(list)\n", " #删除列表中所有与选中数字相同的元素，前后列表长度差即为得到该数字的途径数\n", " #list=list(filter((x)._ne_,list))\n", " try:\n", " while True:\n", " list.remove(x)\n", " except ValueError:\n", " pass\n", " l2=len(list)\n", " rs2=l1-l2\n", " return rs2,list2\n", "out_rs2,out_list2=find_number_of_ways(10)\n", "print('there are',out_rs2,'ways to get',10)\n", "#3.2由于数据太大，需要较长的运行时间\n", "number_of_ways=[]\n", "for i in range(10,61):\n", " l1=len(out_list2)\n", " try:\n", " while True:\n", " out_list2.remove(i)\n", " except ValueError:\n", " pass\n", " l2=len(out_list2)\n", " rs2=l1-l2\n", " number_of_ways.append(rs2)"] }, { "cell_type": "code", "execution_count": 11, "id": "b4b052dc-3879-428e-a5b6-615a09f4587c", "metadata": {}, "outputs": [{ "name": "stdin", "output_type": "stream", "text": ["please input a number: 4\n"] }, { "name": "stdout",

"output_type": "stream", "text": ["[6, 0, 0, 10]\n", "60.0\n", "[7.0, 16.5, 53.666666666666664, 41.25, 148.8, 388.5, 870.8571428571429, 1912.5, 3009.222222222222, 4603.5, 8560.181818181818, 19110.000000000004, 38434.692307692305, 69042.64285714287, 113592.26666666665, 327675.0, 578254.4117647059, 1223334.0000000002, 3035345.789473684, 5767162.5, 11184805.333333334, 26309718.81818182, 44496089.30434783, 72002214.37500001, 183878281.88, 281341002.57692313, 561726042.6296295, 1514742924.642857, 3091635935.7586207, 4688672627.1, 8867029252.129032, 22011707386.875, 43470275051.42424, 94994570776.58824, 163945037351.1143, 370321624627.50006, 702053032595.1082, 1490127600796.2632, 2213119558468.18, 5854899417901.875, 11263289845505.121, 22513809521122.496, 45003266625245.12, 100355424935185.56, 193904983956224.12, 335016412498004.25, 682726539255628.9, 1424967069597690.8, 2389665108400666.5, 6079859496950164.0, 9934410942729028.0, 2.3384074988269884e+16, 4.095726453570903e+16, 9.874559182975312e+16, 1.807990541315283e+17, 3.294061441733848e+17, 6.826508908856332e+17, 1.6150840042983852e+18, 2.569647082301701e+18, 6.264206841697201e+18, 1.137801222579216e+19, 2.298404805958167e+19, 4.640966564576134e+19, 8.18574268270861e+19, 1.8389984861175067e+20, 3.566370520917177e+20, 7.444775518703079e+20, 1.5668881436727407e+21, 2.5921685585317074e+21, 6.274001184383957e+21, 1.1706147901197993e+22, 2.3415067144228665e+22, 4.3989167237689845e+22, 9.291575133862436e+22, 1.4154506471321281e+23, 3.3603365709682945e+23, 7.300655923646787e+23, 1.5809029948806692e+24, 3.190645992274055e+24, 6.69442672611601e+24, 1.2596708540182064e+25, 2.4709264313099013e+25, 5.138663001928204e+25, 9.141781912133483e+25, 1.997998813405448e+26, 3.814582679193323e+26, 7.665979266264349e+26, 1.6318300517852742e+27, 3.2617633619373226e+27, 5.653259512736571e+27, 1.0665329569227895e+28, 2.335939030651544e+28, 5.196685928354974e+28, 1.0493517269176507e+29, 2.0932914516926685e+29, 4.085202129641754e+29, 8.363880248928525e+29, 1.4519773456695786e+30, 3.181931052088029e+30, 6.325576495138865e+30]\n"] }], "source": ["#Dynamic programming\n", "#4.1 random_integer\n", "import numpy as np\n", "import random\n", "N=input('please input a number:')\n", "N=int(N)\n", "\n", "list=[]\n", "i=1\n", "while i<=N:\n", " list.append(random.randint(0,10))\n", " i+=1\n", " \n", "\n", "print(list)\n", "#4.2 sum average\n", "#对于原数列，需要分别求含有1、2、...N个元素的子集的平均值\n", "#由数学知识可知，含有n个元素的子集的取法为N！/（n!*(N-n)！）\n", "#则所有含有n个元素的子集的元素数量和=子集个数*每个子集中的元素数=N！/（n!*(N-n)！）*n\n", "#对于n元素子集，由于取法不重不漏，所以每个元素出现的次数相同，都是N！/（n!*(N-n)！）*n/N\n", "#所以n元素子集平均值=（元素和）*N！/（n!*(N-n)！）/N\n", "import math\n", "sum_ave=[]\n", "sum=0\n", "int_list=[int(x) for x in list]\n", "for i in range(N):\n", " sum=sum+int_list[i]\n", "for i in range(1,N+1):\n", " sum_ave.append(sum*math.factorial(N)/math.factorial(i)/math.factorial(N-i)/N)\n", "sum_average=0\n", "for i in range(N):\n", " sum_average=sum_average+sum_ave[i]\n", "print(sum_average)\n", "#4.3\n", "total_sum_average=[]\n", "for j in range(1,101):\n", " list=[]\n", " i=1\n", " while i<=j:\n", " list.append(random.randint(0,10))\n", " i+=1\n", " \n", " sum_ave=[]\n", " sum=0\n", " int_list=[int(x) for x in list]\n", " for i in range(j):\n", " sum=sum+int_list[i]\n", " for i in range(1,j+1):\n", " sum_ave.append(sum*math.factorial(j)/math.factorial(i)/math.factorial(j-i)/j)\n", " sum_average=0\n", " for i in range(j):\n", " sum_average=sum_average+sum_ave[i]\n", " \n", " total_sum_average.append(sum_average)\n", "print(total_sum_average)"] }, { "cell_type": "code", "execution_count": 34, "id": "e468e898-87fc-4d4c-bb92-53b2b27848df",

"metadata": {}, "outputs": [{ "name": "stdin", "output_type": "stream", "text": ["please input the number of rows: 8\n", "please input the number of columns: 8\n"] }, { "name": "stdout", "output_type": "stream", "text": ["[[1. 1. 0. 0. 1. 0. 1. 0.]\n", " [0. 1. 1. 0. 1. 1. 1. 0.]\n", " [0. 0. 1. 1. 0. 1. 0. 0.]\n", " [0. 0. 1. 0. 1. 1. 1. 0.]\n", " [0. 0. 1. 1. 1. 1. 1. 1.]\n", " [1. 0. 1. 1. 0. 1. 1. 0.]\n", " [0. 0. 0. 0. 1. 1. 1. 1.]\n", " [0. 0. 0. 1. 1. 1. 1. 1.]]\n"] }], "source": ["#Path counting\n", "#5.1\n", "import numpy as np\n", "M=input('please input the number of rows:')\n", "N=input('please input the number of columns:')\n", "M=int(M)\n", "N=int(N)\n", "matrix=np.random.random((M,N))\n", "matrix=np.round(matrix)\n", "matrix[0][0]=1\n", "matrix[-1][-1]=1\n", "print(matrix)"] }, { "cell_type": "code", "execution_count": 74, "id": "6f6a7241-308f-40bf-83d7-cb30a917be26", "metadata": {}, "outputs": [{ "name": "stdout", "output_type": "stream", "text": ["0.0\n"] }], "source": ["#5.2\n", "#对于该问题，由于每次前进一步。即序列数之和每次增加1，如果我们将矩阵以副对角线方向分层，则每次跨越一层\n", "#假如原矩阵是全1矩阵，则由数学知识可知，抵达某层的方法数可以安装杨辉三角的计算规则得出\n", "#再考虑矩阵中出现0的影响：出现0说明此路不通，意味着之前到某0位置的方法全部无效，并且该位置不对此后的位置产生影响\n", "#在杨辉三角中，把相应位置变成0，可以得到相同的效果\n", "def count_path(M,N,matrix):\n", " matrix1=np.zeros((M,N))\n", " matrix1[1][0]=matrix[1][0]\n", " matrix1[0][1]=matrix[0][1]\n", " matrix1[-1][-1]=1\n", " #此时原矩阵matrix相当于布尔矩阵，只用于条件判断，计算在matrix1中进行，在对第2层赋值后，matrix1设置为全0矩阵可以避免一些bug\n", " #计算矩阵左边缘和上边缘\n", " a1=[]\n", " a2=[]\n", " for i in range(N):\n", " a1.append(matrix[0][i])\n", " for i in range(M):\n", " a2.append(matrix[i][0])\n", " index1=next((i for i,x in enumerate(a1) if x==0),False)\n", " index2=next((i for i,x in enumerate(a2) if x==0),False)\n", " if index1:\n", " for i in range(index1):\n", " matrix1[0][i]=1\n", " if index2:\n", " for i in range(index2):\n", " matrix1[i][0]=1\n", "#计算非方阵\n", " if M!=N:\n", " T=min(M,N)\n", " T2=abs(N-M)\n", " for i in range(2,T):\n", " for j in range(i):\n", " if j*(i-j)!=0:\n", " if matrix[j][i-j]!=0:\n", " matrix1[j][i-j]=matrix1[j-1][i-j]+matrix1[j][i-j-1]\n", " else:\n", " matrix1[j][i-j]=0\n", " for i in range(T,T+T2):\n", " if M>N:\n", " for j in range(i-N+1,i):\n", " if i-j==0:\n", " matrix1[j][i-j]=matrix1[j-1][i-j]\n", " else:\n", " if matrix[j][i-j]!=0:\n", " matrix1[j][i-j]=matrix1[j-1][i-j]+matrix1[j][i-j-1]\n", " else:\n", " matrix1[j][i-j]=0\n", " if MN:\n", " for j in range(i-N+1,M):\n", " if matrix[j][i-j]!=0:\n", " matrix1[j][i-j]=matrix1[j-1][i-j]+matrix1[j][i-j-1]\n", " else:\n", " matrix1[j][i-j]=0 \n", " if M