3.2_solutions

February 6, 2023

1 Solutions

1.0.1 for loops

1. let `v=['a', 'g', 'k', 'm']` be a list. Make a for loop that prints each of the elements of the list

[1]:
```
v=['a', 'g', 'k', 'm']
for e in v:
    print(e)
```

```
a
g
k
m
```

2. Make a for loop that goes from the element 6 to the element 30 (included) and add all the elements, then print the result

[3]:
```
cumul_sum=0

for i in range(6,31):
    cumul_sum = cumul_sum+i

print('result:', cumul_sum)
```

```
result: 450
```

3. Using a for loop, sum all the numbers from 1 to 1 million. Then print the result

[4]:
```
cumul_sum=0

for i in range(1,1000000+1):
    cumul_sum = cumul_sum+i

print('result:', cumul_sum)
```

```
result: 500000500000
```

4. Using a for loop, find the factorial of 10. That is, '10!=10*9*8*7*6*5*4*3*2*1'

[8]:
```
cumul_prod=1 #not zero! otherwise all products will equal zero

for i in range(1,11):
    cumul_prod = cumul_prod * i
print('result:', cumul_prod)
```

```
result: 3628800
```

5. Create a 2 dimensional numpy array `Mul[i,j]` of size 10x10, use loops to fill the array with the multiplication table. That is, the element `Mul[4,7]` for instance must be equal to 4x7=28

[12]:
```
import numpy as np  #for instance
Mul=np.zeros((10,10))

for i in range(10):     #rows
    for j in range(10):  #columns
        Mul[i,j]=i * j

print(Mul)
```

```
[[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9.]
 [ 0.  2.  4.  6.  8. 10. 12. 14. 16. 18.]
 [ 0.  3.  6.  9. 12. 15. 18. 21. 24. 27.]
 [ 0.  4.  8. 12. 16. 20. 24. 28. 32. 36.]
 [ 0.  5. 10. 15. 20. 25. 30. 35. 40. 45.]
 [ 0.  6. 12. 18. 24. 30. 36. 42. 48. 54.]
 [ 0.  7. 14. 21. 28. 35. 42. 49. 56. 63.]
 [ 0.  8. 16. 24. 32. 40. 48. 56. 64. 72.]
 [ 0.  9. 18. 27. 36. 45. 54. 63. 72. 81.]]
```

6. You have two lists of equal length. one with first names, and one with family names. For instance `names=['John', 'Emily', ...]`, `f_names=['Adams', 'Blunt',...]`. Use loops to print the i-th name and the i-th family name at each iteration. That is > John Adams, Emily Blunt, ...

[15]:
```
names=['John', 'Emily', 'Peter']
f_names=['Adams', 'Blunt','Berg']

l_list=len(names) # for instance. We know botrh are same length

for i in range(l_list):
    print(names[i], f_names[i])
```

```
John Adams
Emily Blunt
Peter Berg
```

7. You have to numpy arrays of the same length with numbers, for instance `v=[8,3,6,0]` and

`w=[3,-1,8,2]`. Compute a third array with the in which each element i of the new array is the product of the element i of array 1 and the element N-i of array 2. That is, in this example, `[8x2,3x8,6x(-1),0x3]` that is `[16,24,-6,0]`

[19]:
```
v=[8,3,6,0]
w=[3,-1,8,2]
l_list=len(v) # for instance. We know botrh are same length
prod_list=[0]* l_list #initialize a list of l_list elements; all of them zero␣
 ↪(almost like np.zeros(l_list))

for i in range(l_list):
    prod_list[i]=v[i]*w[l_list-i-1]  # do you see why you need the -1 here?


print(prod_list)
```

```
[16, 24, -6, 0]
```

8. Using loops, compute the sum all multiples of three from 3 up to 600 (included)

[25]:
```
my_range=range(3, 601, 3)  #to include 600, we put 601 as upper limit

cumul_sum=0

for i in my_range:
    print(i, end=' ')
    cumul_sum=cumul_sum + i

#new line
print('\n')
# print result
print(cumul_sum)
```

```
3 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48 51 54 57 60 63 66 69 72 75 78 81 84
87 90 93 96 99 102 105 108 111 114 117 120 123 126 129 132 135 138 141 144 147
150 153 156 159 162 165 168 171 174 177 180 183 186 189 192 195 198 201 204 207
210 213 216 219 222 225 228 231 234 237 240 243 246 249 252 255 258 261 264 267
270 273 276 279 282 285 288 291 294 297 300 303 306 309 312 315 318 321 324 327
330 333 336 339 342 345 348 351 354 357 360 363 366 369 372 375 378 381 384 387
390 393 396 399 402 405 408 411 414 417 420 423 426 429 432 435 438 441 444 447
450 453 456 459 462 465 468 471 474 477 480 483 486 489 492 495 498 501 504 507
510 513 516 519 522 525 528 531 534 537 540 543 546 549 552 555 558 561 564 567
570 573 576 579 582 585 588 591 594 597 600

60300
```

[ ]: