

习题

1、 以下能创建形状为(2,3)的二维数组的是(**C**)

np.diag((2,3)) 生成对角矩阵(方阵)

$\begin{bmatrix} 2, 0 \\ 0, 3 \end{bmatrix}$

A.arr = np.diag((2,3))

B.arr = np.eye((2,3))

C.arr = np.ones((2,3))

D.arr = np.array([[1,4],[2,5],[3,6]])

2、Series是Pandas中重要的数据类型，如下代码最后一行的输出结果应该为（ **D** ）

```
from pandas import Series  
disc = {'a':1, 'b':2, 'c':3}  
obj_d = Series(disc, index = ['a','b','d'])  
print(obj_d['d'])
```

A. 2 B.3 C.c D.nan

5、如下代码最后一行的输出结果应该为: **B**

```
import numpy as np  
bArray = np.array([[1,2,3],[4,5,6]])  
print(bArray.ndim)
```

A.1

B.2

C.3

D.4

6、DataFrame的行索引、列索引与数据分别通过
(**D**) 这三个属性获取

A.columns, rows, data

B.rows, columns, value

C.columns, index, data

D.index, columns, values

7、文件写操作时，`writelines`方法的参数不可以是(**D**)

A、 列表

B、 元组

C、 字典

D、 整数

9、下面代码的输出结果是(**D**)

```
li = [1,[2,3],[4,[[5,6],'abc']]]  
print(li[2][1][1][2])
```

A.[5,6]

B.6

C.abc

D.c

12、关于字符串下列说法错误的是 (**B**)

A、字符应该视为长度为1的字符串

B、字符串以\0标志字符串的结束

C、既可以用单引号，也可以用双引号创建字符串

D、在三引号字符串中可包含换行回车等特殊字符

13、 以下不能创建一个字典的语句是 (**C**)

A、 `dict1 = {}`

B、 `dict2 = { 3 : 5 }`

C、 `dict3 = {[1,2,3]: "uestc"}`

D、 `dict4 = {(1,2,3): "uestc"}`

18、函数如下：

```
def showNnumber(numbers):  
    for n in numbers:  
        print(n)
```

下面那些在调用函数时会报错（ **C** ）

A、 showNnumer([2,4])

B、 showNnumber('24')

C、 showNnumber(2,4)

D、 showNumber((2,4))

19、给出如下代码`BTempStr = "Hello World"`可以输出“World”子串的是（ **B** ）

A . `print(TempStr[-5:0])`

B. `print(TempStr[-5:])`

C. `print(TempStr[-5: -1])`

D. `print(TempStr[6: -1])`

22、给出如下代码：

```
x = 3.14
```

```
print(eval('x + 10'))
```

上述代码的输出结果是 (**C**)

A. TypeError: must be str, not int

B. x+10

C. 13.14

D 3.1410

23、给出如下代码：

```
ls = ["car","truck"]
```

```
def funC(a):
```

```
    ls=[]
```

```
    ls.append(a)
```

```
    return
```

```
funC("bus")
```

```
print(ls)
```

运行结果是(**C**)

A. ['car', 'truck', 'bus']

B. ['bus']

C. ['car', 'truck']

D. []

24、下面代码的输出结果是(**A**)

```
def exchange(a,b):
```

```
    a,b = b,a
```

```
    return (a,b)
```

```
x = 10
```

```
y = 20
```

```
x,y = exchange(x,y)
```

```
print(x,y)
```

A.20 10

B.20 20

C.20,10

D.10 10

25、下面代码的输出结果是(**B**)

```
m1 = lambda x,y : (x > y) * x + (x < y) * y  
a = 10  
b = 20  
print(m1(a,b))
```

A.10

B. 20

C.30

D.200

28、 以下（ **B** ）函数可以在绘制图表时， 设置 x 轴的名称。

A. xlim()

B. xlabel()

C. xticks()

29、 pandas中不能使用哪个函数实现合并数据？ (**A**)

A. agg()函数

B. concat()函数

C. join()方法

D. merge()函数

31、以下程序的输出结果是(**D**)

```
for i in reversed(range(10)[::-2]):  
    print(i,end=",")
```

A. 0 2 4 6 8

B. 1 3 5 7 9

C. 0,2,4,6,8

D. 1,3,5,7,9,

34、表达式`list(filter(lambda x: x>5, range(10)))`的值为(**B**)

A. [1,2,3,4,5]

B. [6,7,8,9]

C. [5,6,7,8,9]

D. [6,7,8,9,10]

35、 请阅读下面一段程序：

```
arr = np.array([[11, 20, 13],[14, 25, 16],[27, 18, 9]])
```

```
print(arr[1, :1])
```

执行上述程序后， 最终输出的结果为（ **B** ）。

A. 14

B. [14]

C. [14, 25]

D. [11,14]

37、已知列表对象x = ['11', '2', '3'], 则表达式max(x) 的值为(**A**)。

A.'3'

B.'11'

C.3

D.11

列表推导式/列表生成式

(1) 构建如下列表:

[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]

[i for i in range(0,19,2)]

[i*2 for i in range(10)]

列表推导式/列表生成式

(2) 构建如下列表:

`[(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 6)]`

`[(i,i+1) for i in range(6)]`

列表推导式/列表生成式

(3) 50以内能被3整除的数的平方

```
[i*i for i in range(50) if i%3==0]
```


列表推导式/列表生成式

(4) 构建一个列表，列表里面是三种不同尺寸的T恤衫，每个尺寸都有两个颜色（列表里面的元素为元组类型）。

```
colors = ['black', 'white']
```

```
sizes = ['S', 'M', 'L']
```

```
[(x,y) for x in colors for y in sizes]
```

列表推导式/列表生成式

(6) 定义一个列表推导式，把字典

`d = {'x': 'A', 'y': 'B', 'z': 'C' }`

转为： `['x=A','y=B','z=C']`的形式

`[f'{k}={v}' for k,v in d.items()]`

`['{}={}'].format(k,v) for k,v in d.items()]`

`[k+'='+v for k,v in d.items()]`

列表推导式/列表生成式

（8）取出所有带a且长度大于3的字符串组成的列表：

```
list1 = ['52pojie', 'OVVO', 'asd', 'sdf', 'freg',  
'sfag', 'refv', 'aser', 'sdfr']
```

```
[x for x in list1 if 'a' in x and len(x)>3]
```

列表推导式/列表生成式

(9) 应用列表推导式，去除列表中成绩小于70的字典项：

```
dict_list = [{"科目": "政治", "成绩": 98},  
             {"科目": "语文", "成绩": 77},  
             {"科目": "数学", "成绩": 99},  
             {"科目": "历史", "成绩": 65}]
```

```
[d for d in dict_list if d['成绩'] >= 70]
```

程序填空

1、三名同学的四门课的成绩如下，请：（1）将第3个同学的第一门课的成绩改为60；（2）将所有同学第3门课成绩设置为未知：

```
from numpy import array, nan as NA
import pandas as pd
```

```
arr = np.array([[66,75,80,71],[88,92,86,75],[53,67,65,71]])
```

```
print(arr)
```

```
arr[2,0]=60
```

```
df = pd.DataFrame(arr)
```

```
df.iloc[:,2]=NA
```

```
print(df)
```

- 2、将df中的三名同学的四门课数据按以下方式填充：
- （1）第一列的NA值用70填充；第二列的NA值用80填充，第三列的NA值用60填充；
 - （2）第四列用前一个同学的同列分数填充。

```
from numpy import array, nan as NA
import pandas as pd
```

```
arr = array([[NA,NA,80,71],[88,92,NA,75],[53,80,65,NA]])
df = pd.DataFrame(arr,index=['A','B','C'],columns=[1,2,3,4])
```

```
_____ df = df.fillna({1:70,2:80,3:60})
_____ df[4] = df[4].fillna(method='ffill')
print(df)
```

3、创建一个长度为10，位于[1,100]区间的整数随机数组a并将最大值替换为0，然后打印a。

```
import numpy as np
```

```
a = np.random.randint(1,101,10)
```

```
a[n.argmax()] = 0
```

```
print(a)
```

4、创建一个3*2的[0,10)范围内的随机整数数组n，交换两行的元素后打印n。

```
import numpy as np
```

```
n = np.random.randint(0,10,(3,2))
```

```
n[[0,1]] = n[[1,0]]
```

```
print(n)
```


5、将数组 `np.arange(20)` 转变为 4 行 5 列的二维数组，并交换第 1 列和第 2 列的值。

```
import numpy as np
```

```
n = np.arange(20).reshape(4,5)
```

```
n = n[:,[1,0,2,3]] 或 n[:,[0,1]] = n[:,[1,0]]
```

```
print(n)
```

6、寻找数组 `np.random.randint(1,10,size=(3,3))` 中所有的奇数，并将所有奇数替换为 0.

```
import numpy as np
```

```
n = np.random.randint(1,10,size=(3,3))
```


```
n[n%2==1] = 0
```

```
print(n)
```

1、写程序运行结果：

```
import csv
with open('e:\\1.csv') as file:
    csv_reader = csv.reader(file, delimiter='\\t')
    headers = next(csv_reader)
    print(headers)
    for row in csv_reader:
        print(row)
```

```
['name', 'score']
['Mike', '61']
['hero', '89']
['trump', '90']
```

 1.csv - 记事本

文件(E) 编辑(E) 格式(O) 查看

name	score
Mike	61
hero	89
trump	90

2、写程序运行结果：

```
import pandas as pd
import numpy as np
```

```
df = pd.DataFrame([[1,2], [3,5]], columns=list('AB'))
print(df)
print('*****')
df1 = df.apply(np.square)
print(df1)
print('*****')
print(df.apply(np.mean))
```

```
   A  B
0  1  2
1  3  5
*****
   A  B
0  1  4
1  9 25
*****
A    2.0
B    3.5
dtype: float64
```

3、写程序运行结果：

```
import pandas as pd
```

```
ser = pd.Series(range(1, 6), index=[5, 3, 0, 4, 2])
```

```
print(ser.sort_index())
```

```
0      3
2      5
3      2
4      4
5      1
dtype: int64
```

4、写程序运行结果：

```
from numpy import nan as NA
import pandas as pd
```

```
data = {'grammar': ['Python', 'C', 'Java', 'C', NA, 'SQL'],\
        'score': [1.0, 2.0, NA, 4.0, 5.0, 6.0]}
df = pd.DataFrame(data)
print(df)
```

	grammar	score
0	Python	1.0
1	C	2.0
2	Java	NaN
3	C	4.0
4	NaN	5.0
5	SQL	6.0

填空：提取含有字符串C的行

```
result = df[df['grammar'] == 'C']
```

```
print(result)
```

	grammar	score
0	Python	1.0
1	C	2.0
2	Java	NaN
3	C	4.0
4	NaN	5.0
5	SQL	6.0

	grammar	score
1	C	2.0
3	C	4.0

填空：原地修改第二列列名为**popularity**

`df.rename(columns={'score': 'popularity'}, inplace=True)`

`print(df)`

	grammar	score
0	Python	1.0
1	C	2.0
2	Java	NaN
3	C	4.0
4	NaN	5.0
5	SQL	6.0

	grammar	popularity
0	Python	1.0
1	C	2.0
2	Java	NaN
3	C	4.0
4	NaN	5.0
5	SQL	6.0

填空：统计grammar列中每种编程语言出现的次数

```
print(df['grammar'].value_counts())
```

	grammar	score
0	Python	1.0
1	C	2.0
2	Java	NaN
3	C	4.0
4	NaN	5.0
5	SQL	6.0

C	2
Python	1
Java	1
SQL	1
Name: grammar, dtype: int64	

填空：提取**popularity**列中值大于3的行

```
result = df[df['popularity'] > 3]
```

```
print(result)
```

	grammar	score
0	Python	1.0
1	C	2.0
2	Java	NaN
3	C	4.0
4	NaN	5.0
5	SQL	6.0

	grammar	popularity
3	C	4.0
4	NaN	5.0
5	SQL	6.0

填空： 输出grammar列从前往后去除重复值后的数据帧

```
print( df.drop_duplicates('grammar', keep='first') )
```

	grammar	popularity
0	Python	1.0
1	C	2.0
2	Java	NaN
4	NaN	5.0
5	SQL	6.0

填空： 输出popularity列的平均值

```
print(      df['popularity'].mean()      )
```

3.6

填空：将DataFrame保存到df.csv

df.to_csv('df.csv')

填空： 对数据按照"**popularity**"列值由大到小进行排序

```
df.sort_values('popularity', inplace=True, ascending=False)
```

```
print(df)
```

	grammar	popularity
6	Perl	7.0
5	SQL	6.0
4	NaN	5.0
3	C	4.0
1	C	2.0
0	Python	1.0
2	Java	NaN

填空： 将"Grammar"列NA填充为"R"

```
df['grammar']=df['grammar'].fillna('R')
```

```
print(df)
```

	grammar	popularity
6	Perl	7.0
5	SQL	6.0
4	R	5.0
3	C	4.0
1	C	2.0
0	Python	1.0
2	Java	NaN

填空：添加一列： `len_str`，其值为`grammar`列每个字符串的长度。

```
df['len_str'] = df['grammar'].apply(len)
```

```
print(df)
```

	grammar	popularity	len_str
6	Perl	7.0	4
5	SQL	6.0	3
4	R	5.0	1
3	C	4.0	1
1	C	2.0	1
0	Python	1.0	6
2	Java	NaN	4