examen

January 28, 2020

[32]:

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

def generate():
    for x in range(10):
        yield(x)
def fromiter(gen):
    a=generate()
    Z=np.fromiter(a,dtype=float, count=-1)
    return Z
if __name__=='__main__':
    with multiprocessing.Pool(3) as pool:
        a=pool.map(fromiter,(1,1,1))

        a=np.concatenate((a[0],a[1],a[2]))
        print(a)
```

```
[0. 1. 2. 3. 4. 5. 6. 7. 8. 9. 0. 1. 2. 3. 4. 5. 6. 7. 8. 9. 0. 1. 2. 3.
 4. 5. 6. 7. 8. 9.]
```

[72]:

```
#26. NumPy
a1=0
a2=1/3
a3=2/3
b1=1/3
b2=2/3
b3=1
n1=10//3
n2=10//3
n3=10-n1-n2
def p1(a,b,n):
    v1 = np.linspace(start=a, stop=b, num=n)[1:-1]
    print(v1)
```

```
procs =[multiprocessing.Process(target=p1, args=(a1,b1,n1+2)), multiprocessing.
 ↪Process(target=p1, args=(a2,b2,n2+2)), multiprocessing.Process(target=p1,␣
 ↪args=(a3,b3,n3+2))]
for proc in procs:
    proc.start()
for proc in procs:
    proc.join()
```

```
[0.08333333 0.16666667 0.25      ]
[0.41666667 0.5        0.58333333]
[0.73333333 0.8        0.86666667 0.93333333]
```

[68]:

```
#25. NumPy

import multiprocessing
import numpy as np

n3=10

def generate (a,b,t):
    for x in range (a,b,t):
        yield x

t=1
n1=10//3
n2=n1+10//3

def fromiter_ (y):
    y=generate (y[0],y[1],y[2])
    Z=np.fromiter(y, dtype=float,count=-1)
    return Z

if __name__=='__main__':
    with multiprocessing.Pool(3) as pool:
        y=pool.map(fromiter_,[(0,n1,t),(n1,n2,t),(n2,n3,t)])

y=np.concatenate((y[0],y[1],y[2]))
print(y)
```

```
[0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]
```

[27]:

```
v = np.random.random(size=[10])
print(v)
print()
v = np.sort(a=v, axis=0)
print(v)
```

```
[0.7058559  0.55877878 0.94146545 0.72653983 0.89258148 0.62626906
 0.24128436 0.92007496 0.94127867 0.14604896]

[0.14604896 0.24128436 0.55877878 0.62626906 0.7058559  0.72653983
 0.89258148 0.92007496 0.94127867 0.94146545]
```

[130]:

```
# 27.
n=10
v = np.random.random(size=[n])
print(v)
print()
n1=n//3
n2=(n//3)*2
n3=n-n1-n2
def sort (v1):
    s=np.sort(a=v1,axis=0)
    print(s)

if __name__=='__main__':
    with multiprocessing.Pool(3) as pool:
        a=pool.map(sort,(v[0:n1],v[n1:n2],v[n2:n]))

        a=np.sort((a[0],a[1],a[2]))
        print(a)
```

```
[0.3753271  0.61100631 0.93549464 0.24736914 0.22014644 0.84905844
 0.76951803 0.32594143 0.25861585 0.77673997]

[0.22014644 0.24736914 0.84905844]
[0.3753271  0.61100631 0.93549464]
[0.25861585 0.32594143 0.76951803 0.77673997]
```

```
      ␣
↪---------------------------------------------------------------------------

       TypeError                                 Traceback (most recent call␣
↪last)

       <ipython-input-130-5e0ccb305db2> in <module>
        13 if __name__=='__main__':
        14     with multiprocessing.Pool(3) as pool:
   ---> 15         a=np.sort(pool.map(sort,(v[0:n1],v[n1:n2],v[n2:n])))
        16
        17
       ~/anaconda3/lib/python3.7/site-packages/numpy/core/fromnumeric.py in␣
↪sort(a, axis, kind, order)
       932     else:
       933         a = asanyarray(a).copy(order="K")
   --> 934     a.sort(axis=axis, kind=kind, order=order)
       935     return a
       936


       TypeError: '<' not supported between instances of 'NoneType' and␣
↪'NoneType'
```

[ ]:

```
procs = [multiprocessing.Process(target=sort, args=(v[0:n1])), multiprocessing.
 ↪Process(target=sort, args=(v[n1+1:n2])), multiprocessing.
 ↪Process(target=sort, args=v[n3:n])]
for proc in procs:
    proc.start()
for proc in procs:
    proc.join()
```

[50]:

```
v = np.random.random(size=[5])
print(v)
print(v[0:5])
```

```
[0.23172924 0.65457259 0.71505857 0.87879617 0.73029249]
[0.23172924 0.65457259 0.71505857 0.87879617 0.73029249]
```

[ ]:

```
a = np.array([1,2,3,4,5])
b = np.array([1,2,3,4,5])
print(' 1')
print(np.array_equal(a,b))
print()
print(' 2')
print(np.allclose(a,b))
```

[131]:

```
# 28.  ,    2 NumPy

n=10
n1=n/2
n2=n-n1

def compare (array_list):
    equal = np.allclose(array_list[0], array_list[1])
    return equal

if __name__=='__main__':
    array1 = np.array([1,2,3,4,5])
    array2 = np.array([1,2,3,4,5])
    Len=len(array1)

    with multiprocessing.Pool(3) as pool:
        Res = pool.map(compare, [(array1[0:Len/3], array2[0:Len/3]),␣
 ↪(array1[Len/3:Len*2/3], array2[Len/3:Len*2/3]), (array1[Len*2/3:],␣
 ↪array2[Len*2/3:])])

Print(res[0] and res[1] and res[2])
```

```
       ␣
 ↪---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call␣
 ↪last)

        <ipython-input-131-50b9001a9d39> in <module>
         15
         16     with multiprocessing.Pool(3) as pool:
    ---> 17         Res = pool.map(compare, [(array1[0:Len/3], array2[0:Len/3]),␣
 ↪(array1[Len/3:Len*2/3], array2[Len/3:Len*2/3]), (array1[Len*2/3:],␣
 ↪array2[Len*2/3:])])
         18
         19 Print(res[0] and res[1] and res[2])


        TypeError: slice indices must be integers or None or have an __index__␣
 ↪method
```

[135]:
```
def compare (array_list):
    equal = np.allclose(array_list[0], array_list[1]
    return(equal)

Array1 = a()
Array2 = a()

with multiprocessing.Pool(3) as pool:
    Res = pool.map(compare, [(array1[0:Len/3], array2[0:Len/3]), (array1[Len/3:
 ↪Len*2/3], array2[Len/3:Len*2/3]), (array1[Len*2/3:], array2[Len*2/3:])])

Print(res[0] and res[1] and res[2])
```

```
          File "<ipython-input-135-710006e223b6>", line 3
```

```
        return(equal)
              ^
    SyntaxError: invalid syntax
```

[101]:

```
Z = np.zeros(10)
print(type (Z))
Z.flags.writeable = False
print(Z)
print(type(Z))
```

```
<class 'numpy.ndarray'>
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
<class 'numpy.ndarray'>
```

[106]:

```
# 29.

n0=10
n1=n/3
n2=n1+n/3
n3=n-n1-n2

def vec(n):
    Z=int(np.zeros(n))
    return Z

if __name__=='__main__':
    with multiprocessing.Pool(3) as pool:
        rez=pool.map(vec,(3,3,4))
print(res)
```

```
      ␣
↪---------------------------------------------------------------------------

        RemoteTraceback                           Traceback (most recent call␣
↪last)

        RemoteTraceback:
    """
    Traceback (most recent call last):
      File "/Users/anastasiyashabrova/anaconda3/lib/python3.7/multiprocessing/
↪pool.py", line 121, in worker
        result = (True, func(*args, **kwds))
      File "/Users/anastasiyashabrova/anaconda3/lib/python3.7/multiprocessing/
↪pool.py", line 44, in mapstar
```

```
       return list(map(*args))
     File "<ipython-input-106-fa7ba48ecc1e>", line 9, in vec
       Z=int(np.zeros(n))
   TypeError: only size-1 arrays can be converted to Python scalars
   """


   The above exception was the direct cause of the following exception:


       TypeError                                 Traceback (most recent call␣
↪last)

       <ipython-input-106-fa7ba48ecc1e> in <module>
        12 if __name__=='__main__':
        13     with multiprocessing.Pool(3) as pool:
   ---> 14         rez=pool.map(vec,(3,3,4))
        15 print(res)


       ~/anaconda3/lib/python3.7/multiprocessing/pool.py in map(self, func,␣
↪iterable, chunksize)
       266         in a list that is returned.
       267         '''
   --> 268         return self._map_async(func, iterable, mapstar, chunksize).
↪get()
       269
       270     def starmap(self, func, iterable, chunksize=None):


       ~/anaconda3/lib/python3.7/multiprocessing/pool.py in get(self, timeout)
       655             return self._value
       656         else:
   --> 657             raise self._value
       658
       659     def _set(self, i, obj):


       TypeError: only size-1 arrays can be converted to Python scalars
```

[111]:

```
#36 Pandas

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

```
test_list = 'ANASTASIYA'
test_array = np.arange(len(my_list))
dict_test = dict(zip(test_list, test_array))
my_series = pd.Series(dict_test)

my_dataframe = pd.DataFrame()

series_split = np.split(my_series, [len(my_series)//3,␣
 ↪len(my_series)-(len(my_series)//3)])
for i, sub_series in enumerate(series_split):
    globals()["sub_series%d"%i] = sub_series

def train(data):
    new_dataframe = data.to_frame().reset_index()
    return(new_dataframe)

with multiprocessing.Pool(3) as pool:
    result = pool.map(train, [sub_series0, sub_series1, sub_series2])
my_dataframe = my_dataframe.append(result)
print(my_dataframe)
```

```
  index  0
0     A  9
1     N  1
0     S  6
1     T  4
0     I  7
1     Y  8
```

[119]:

```
#37 Pandas
data = pd.read_csv('Documents//aptm1.csv', delimiter=';',nrows=20)

def ex_1(data_1):
    print('\n', ' :')
    print(data_1.dtypes)

process1 = multiprocessing.Process(target=ex_1, args=(data,))

def ex_2(data_2):
    print('\n', ':')
    print(data_2.shape)

process2 = multiprocessing.Process(target=ex_2, args=(data,))

def ex_3(data_3):
    print('\n', ' ')
    print(data_3.describe())
```

```
process3 = multiprocessing.Process(target=ex_3, args=(data,))

if __name__ == '__main__':
    process1.start()
    process1.join()
    process2.start()
    process2.join()
    process3.start()
    process3.join()
```

```
  :
dp    object
da    object
ds    object
de    object
df    object
di    object
dn    object
dtype: object

 :
(20, 7)


                 dp           da            ds           de  df            di  \
count            20           20            20           20  20            20
unique           20           20            20           20   2            20
top     0,027199074  0,020381331  -0,181954641  -0,01445709   0  -0,007267764
freq              1            1             1            1  19             1

                  dn
count             20
unique            20
top     -0,009059941
freq               1
```

[142]:

```
# 38.()     DataFrame
import random
import pandas as pd
import numpy as np
import multiprocessing

s = 'abcdefghijklmnopqrstuvwxyz'

#  DataFrame
```

```
df = pd.DataFrame(np.random.randint(low=1, high=10, size=[3,5]))

#
df = pd.concat([df, pd.DataFrame({'letter': ['a', 'b', 'c']})], axis=1)

#
df = pd.concat([df, pd.DataFrame({'r_letter': [random.choice(s) for i in␣
 ↪range(len(df))]})], axis=1)

print(' DataFrame:')
print(df)
print()

#␣
 ↪-------------------------------------------------------------------------------------------

#      ()
row, col = np.where(df.values == 5)
print('    :')
print(row)
print(col)
print()

#     (   )
if (row.size != 0) and (col.size != 0):
    print('    (, ):')
    print(df.iat[row[0], col[0]])
    print(df.iloc[row[0], col[0]])
    print()

#         ( )
if (row.size != 0) and (col.size != 0):
    print('     :')
    print(df.at[row[0], 1])
    print(df.at[row[0], 'letter'])
    print()

#       DataFrame
# (loc;at -  ()      ( ))
# (iloc, iat) -
ans1 = df.loc[df['letter'] == 'a']
print('   :')
print(ans1)
```

```
 DataFrame:
   0  1  2  3  4 letter r_letter
0  4  2  6  2  7      a        j
```

```
1  8  2  7  2  7      b        y
2  1  5  9  5  5      c        t

   :
[2 2 2]
[1 3 4]

   (, ):
5
5

    :
5
c

  :
   0  1  2  3  4 letter r_letter
0  4  2  6  2  7      a        j
```

[175]:

```
#38. Pandas
import random
import pandas as pd
import numpy as np
import multiprocessing

s = 'abcdefghijklmnopqrstuvwxyz'


#  DataFrame
df = pd.DataFrame(np.random.randint(low=1, high=10, size=[3,5]))

process1 = multiprocessing.Process(target=s)
process2 = multiprocessing.Process(target=df)
if __name__ == '__main__':
    process1.start()
    process2.start()
    process1.join()
    process2.join()

#
df = pd.concat([df, pd.DataFrame({'letter': ['a', 'b', 'c']})], axis=1)

#
df = pd.concat([df, pd.DataFrame({'r_letter': [random.choice(s) for i in␣
 ↪range(len(df))]})], axis=1)
row, col = np.where(df.values == 5)
print(' DataFrame:')
print(df)
print()

row, col = np.where(df.values == 5)
print('   :')
print(row)
print(col)
print()

#     (   )
def ind(df,col,row):
    if (row.size != 0) and (col.size != 0):
        print('   (, ):')
        print(df.iat[row[0], col[0]])
        print(df.iloc[row[0], col[0]])



#         ( )
def na_ind(df,col,row):
    if (row.size != 0) and (col.size != 0):
        print('    :')
        print(df.at[row[0], 1])
        print(df.at[row[0], 'letter'])

#       DataFrame
# (loc;at -   ()      ( ))
# (iloc, iat) -
def an1(df):
    ans1 = df.loc[df['letter'] == 'a']
    print('  :')
    print(ans1)

procs = [multiprocessing.Process(target=ind, args=(df,col,row)),␣
 ↪multiprocessing.Process(target=na_ind, args=(df,col,row)), multiprocessing.
 ↪Process(target=an1, args=(df))]
for proc in procs:
    proc.start()
for proc in procs:
    proc.join()
```

```
Process Process-382:
Traceback (most recent call last):
Process Process-383:
  File
"/Users/anastasiyashabrova/anaconda3/lib/python3.7/multiprocessing/process.py",
line 297, in _bootstrap
```

```
    self.run()
  File
"/Users/anastasiyashabrova/anaconda3/lib/python3.7/multiprocessing/process.py",
line 99, in run
    self._target(*self._args, **self._kwargs)
TypeError: 'str' object is not callable
Traceback (most recent call last):
  File
"/Users/anastasiyashabrova/anaconda3/lib/python3.7/multiprocessing/process.py",
line 297, in _bootstrap
    self.run()
  File
"/Users/anastasiyashabrova/anaconda3/lib/python3.7/multiprocessing/process.py",
line 98, in run
    if self._target:
  File "/Users/anastasiyashabrova/anaconda3/lib/python3.7/site-
packages/pandas/core/generic.py", line 1478, in __nonzero__
    .format(self.__class__.__name__))
ValueError: The truth value of a DataFrame is ambiguous. Use a.empty, a.bool(),
a.item(), a.any() or a.all().

 DataFrame:
   0  1  2  3  4 letter r_letter
0  8  3  1  8  5      a        d
1  1  3  2  2  2      b        n
2  5  4  1  1  4      c        f

   :
[0 2]
[4 0]

   (, ):
    :
5
5

Process Process-386:

3
a

Traceback (most recent call last):
  File
"/Users/anastasiyashabrova/anaconda3/lib/python3.7/multiprocessing/process.py",
line 297, in _bootstrap
    self.run()
  File
"/Users/anastasiyashabrova/anaconda3/lib/python3.7/multiprocessing/process.py",
line 99, in run
```

```
    self._target(*self._args, **self._kwargs)
TypeError: an1() takes 1 positional argument but 7 were given
```

[ ]:

```
# 39.()    DataFrame   (  )

#   ( 10 )
df = pd.read_csv('https://raw.githubusercontent.com/Grossmend/CSV/master/
 ↪titanic/data.csv', nrows=10)



#
def change_values(val):

    """     """


    #
    try:
        float(val)
    except Exception as e:
        return val

    #
    if val > 25:
        return 'High'
    elif val < 25:
        return 'Low'

col_df = df['Age'].apply(change_values)
col_df
```

[169]:

```
#39. Pandas

df=pd.read_csv('Documents//aptm1.csv', delimiter=';', nrows=10)
#
def change_values(val):
    """""   """
    #
    try:
        float(val)
    except Exception as e:
        return val

    #
    if val>-0.0025:
        return 'High'
    elif val<-0.0025:
        return 'Low'
```

```python
def col_i(c):
  changed_data = df.iloc[c].apply(change_values)
  return changed_data

if __name__ == '__main__':
  with multiprocessing.Pool(3) as p:
      result = p.map(col_i, (0,1,2))

result = pd.concat(result)
print(result)
```

```
dp     -0,07231405
da    -0,028217491
ds    -0,028349448
de    -0,014154612
df            High
di    -0,033905614
dn    -0,026440579
dp     0,001670379
da    -0,011869685
ds    -0,000477702
de     0,025875671
df            High
di     0,007988658
dn    -0,009754701
dp    -0,036687048
da    -0,014043924
ds      -0,0309552
de     -0,01445709
df            High
di    -0,037754675
dn    -0,025816403
dtype: object
```