.1) Write a Pandas program to create a dataframe from a dictionary and display it

```
import pandas as pd
df = pd.DataFrame({'X':[78,85,96,80,86], 'Y':[84,94,89,83,86],'Z':[86,97,96,72,83]});
print(df)
```

```
    X   Y   Z
0  78  84  86
1  85  94  97
2  96  89  96
3  80  83  72
4  86  86  83
```

2) Write a Pandas program to create and display a DataFrame from a specified dictionary data which has the index labels.

```
import pandas as pd
import numpy as np

exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthe
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)
print(df)
```

```
       name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
d      James    NaN         3      no
e      Emily    9.0         2      no
f    Michael   20.0         3     yes
g    Matthew   14.5         1     yes
h      Laura    NaN         1      no
i      Kevin    8.0         2      no
j      Jonas   19.0         1     yes
```

3)Write a Pandas program to get the first 3 rows of a given DataFrame

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

```
exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthe
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)
print("First three rows of the data frame:")
print(df.iloc[:3])
```

```
First three rows of the data frame:
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
```

4) Write a Pandas program to select the rows where the number of attempts in the examination is greater than 2.

```
import pandas as pd
import numpy as np

exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthe
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts' : [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)
print("Number of attempts in the examination is greater than 2:")
print(df[df['attempts'] > 2])
```

```
Number of attempts in the examination is greater than 2:
      name  score  attempts qualify
b     Dima    9.0         3      no
d    James    NaN         3      no
f  Michael   20.0         3     yes
```

5) Write a Pandas program to select the rows the score is between 15 and 20 (inclusive)

```
import pandas as pd
import numpy as np
exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthe
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
```

```
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)
print("Rows where score between 15 and 20 (inclusive):")
print(df[df['score'].between(15, 20)])
```

```
Rows where score between 15 and 20 (inclusive):
        name  score  attempts qualify
c  Katherine   16.5         2     yes
f    Michael   20.0         3     yes
j      Jonas   19.0         1     yes
```

6) Write a Pandas program to calculate the mean of all students' scores. Data is stored in a dataframe.

```
import pandas as pd
import numpy as np
exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthe
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)
print("\nMean score for each different student in data frame:")
print(df['score'].mean())
```

```
Mean score for each different student in data frame:
13.5625
```

7) Write a Pandas program to append a new row 'k' to data frame with given values for each column. Now delete the new row and return the original DataFrame.

```
import pandas as pd
import numpy as np
exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthe
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data , index=labels)
print("Original rows:")
print(df)
print("\nAppend a new row:")
df.loc['k'] = [1, 'Suresh', 'yes', 15.5]
```

```
print("Print all records after insert a new record:")
print(df)
print("\nDelete the new row and display the original  rows:")
df = df.drop('k')
print(df)
```

```
Original rows:
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
d      James    NaN         3      no
e      Emily    9.0         2      no
f    Michael   20.0         3     yes
g    Matthew   14.5         1     yes
h      Laura    NaN         1      no
i      Kevin    8.0         2      no
j      Jonas   19.0         1     yes

Append a new row:
Print all records after insert a new record:
        name   score attempts qualify
a  Anastasia    12.5        1     yes
b       Dima     9.0        3      no
c  Katherine    16.5        2     yes
d      James     NaN        3      no
e      Emily     9.0        2      no
f    Michael    20.0        3     yes
g    Matthew    14.5        1     yes
h      Laura     NaN        1      no
i      Kevin     8.0        2      no
j      Jonas    19.0        1     yes
k          1  Suresh      yes    15.5

Delete the new row and display the original  rows:
        name score attempts qualify
a  Anastasia  12.5        1     yes
b       Dima   9.0        3      no
c  Katherine  16.5        2     yes
d      James   NaN        3      no
e      Emily   9.0        2      no
f    Michael  20.0        3     yes
g    Matthew  14.5        1     yes
h      Laura   NaN        1      no
i      Kevin   8.0        2      no
j      Jonas  19.0        1     yes
```

8) Write a Pandas program to insert a new column in existing DataFrame.

```
import pandas as pd
import numpy as np
exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthe
```

```
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data , index=labels)
print("Original rows:")
print(df)
color = ['Red','Blue','Orange','Red','White','White','Blue','Green','Green','Red']
df['color'] = color
print("\nNew DataFrame after inserting the 'color' column")
print(df)
```

```
Original rows:
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
d      James    NaN         3      no
e      Emily    9.0         2      no
f    Michael   20.0         3     yes
g    Matthew   14.5         1     yes
h      Laura    NaN         1      no
i      Kevin    8.0         2      no
j      Jonas   19.0         1     yes

New DataFrame after inserting the 'color' column
        name  score  attempts qualify   color
a  Anastasia   12.5         1     yes     Red
b       Dima    9.0         3      no    Blue
c  Katherine   16.5         2     yes  Orange
d      James    NaN         3      no     Red
e      Emily    9.0         2      no   White
f    Michael   20.0         3     yes   White
g    Matthew   14.5         1     yes    Blue
h      Laura    NaN         1      no   Green
i      Kevin    8.0         2      no   Green
j      Jonas   19.0         1     yes     Red
```

9) Write a Pandas program to select rows from a given DataFrame based on values in some columns

```
import pandas as pd
import numpy as np
d = {'col1': [1, 4, 3, 4, 5], 'col2': [4, 5, 6, 7, 8], 'col3': [7, 8, 9, 0, 1]}
df = pd.DataFrame(data=d)
print("Original DataFrame")
print(df)
print('Rows for colum1 value == 4')
print(df.loc[df['col1'] == 4])
```

```
Original DataFrame
   col1  col2  col3
0     1     4     7
1     4     5     8
2     3     6     9
3     4     7     0
4     5     8     1
Rows for colum1 value == 4
   col1  col2  col3
1     4     5     8
3     4     7     0
```

10) Write a Pandas program to set a given value for particular cell in DataFrame using index value

```
import pandas as pd
import numpy as np
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
df = pd.DataFrame(exam_data)
print("Original DataFrame")
print(df)
print("\nSet a given value for particular cell in the DataFrame")
df.set_value(8, 'score', 10.2)
print(df)
```

11) Write a Pandas program to get the specified row value of a given DataFrame

```
import pandas as pd
d = {'col1': [1, 2, 3, 4, 7], 'col2': [4, 5, 6, 9, 5], 'col3': [7, 8, 12, 1, 11]}
df = pd.DataFrame(data=d)
print("Original DataFrame")
print(df)
print("Value of Row1")
print(df.iloc[0])
print("Value of Row4")
print(df.iloc[3])
```

```
Original DataFrame
   col1  col2  col3
0     1     4     7
1     2     5     8
2     3     6    12
3     4     9     1
4     7     5    11
Value of Row1
```

```
col1    1
col2    4
col3    7
Name: 0, dtype: int64
Value of Row4
col1    4
col2    9
col3    1
Name: 3, dtype: int64
```

12) Write a Pandas program to add a prefix or suffix to all columns of a given DataFrame

```
import pandas as pd
df = pd.DataFrame({'W':[68,75,86,80,66],'X':[78,85,96,80,86], 'Y':[84,94,89,83,86],'Z':[86,97
print("Original DataFrame")
print(df)
print("\nAdd prefix:")
print(df.add_prefix("OO_"))
print("\nAdd suffix:")
print(df.add_suffix("_OYO"))
```

```
Original DataFrame
    W   X   Y   Z
0  68  78  84  86
1  75  85  94  97
2  86  96  89  96
3  80  80  83  72
4  66  86  86  83

Add prefix:
   OO_W  OO_X  OO_Y  OO_Z
0    68    78    84    86
1    75    85    94    97
2    86    96    89    96
3    80    80    83    72
4    66    86    86    83

Add suffix:
   W_OYO  X_OYO  Y_OYO  Z_OYO
0     68     78     84     86
1     75     85     94     97
2     86     96     89     96
3     80     80     83     72
4     66     86     86     83
```

13) Write a Pandas program to combine many given series to create a DataFrame.

```
import pandas as pd
sr1 = pd.Series(['php', 'python', 'java', 'c#', 'c++'])
sr2 = pd.Series([1, 2, 3, 4, 5])
```

```
print("Original Series:")
print(sr1)
print(sr2)
print("Combine above series to a dataframe:")
ser_df = pd.DataFrame(sr1, sr2).reset_index()
print(ser_df.head())
print("\nUsing pandas concat:")
ser_df = pd.concat([sr1, sr2], axis = 1)
print(ser_df.head())
print("\nUsing pandas DataFrame with a dictionary, gives a specific name to the columns:")
ser_df = pd.DataFrame({"col1":sr1, "col2":sr2})
print(ser_df.head(5))
```

```
Original Series:
0       php
1    python
2      java
3        c#
4       c++
dtype: object
0    1
1    2
2    3
3    4
4    5
dtype: int64
Combine above series to a dataframe:
   index       0
0      1  python
1      2    java
2      3      c#
3      4     c++
4      5     NaN

Using pandas concat:
        0  1
0     php  1
1  python  2
2    java  3
3      c#  4
4     c++  5

Using pandas DataFrame with a dictionary, gives a specific name to the columns:
     col1  col2
0     php     1
1  python     2
2    java     3
3      c#     4
4     c++     5
```

14) Write a Pandas program to fill missing values in time series data

```
import pandas as pd
import numpy as np
sdata = {"c1":[120, 130 ,140, 150, np.nan, 170], "c2":[7, np.nan, 10, np.nan, 5.5, 16.5]}
df = pd.DataFrame(sdata)
df.index = pd.util.testing.makeDateIndex()[0:6]
print("Original DataFrame:")
print(df)
print("\nDataFrame after interpolate:")
print(df.interpolate())
```

```
Original DataFrame:
               c1    c2
2000-01-03  120.0   7.0
2000-01-04  130.0   NaN
2000-01-05  140.0  10.0
2000-01-06  150.0   NaN
2000-01-07    NaN   5.5
2000-01-10  170.0  16.5

DataFrame after interpolate:
               c1     c2
2000-01-03  120.0   7.00
2000-01-04  130.0   8.50
2000-01-05  140.0  10.00
2000-01-06  150.0   7.75
2000-01-07  160.0   5.50
2000-01-10  170.0  16.50
/usr/local/lib/python3.9/dist-packages/pandas/util/__init__.py:16: FutureWarning: pandas
  import pandas.util.testing
```

15) Write a function division() that accepts two arguments. The function should be able to catch an exception such as ZeroDivisionError, ValueError, or any unknown error you might come across when you are doing a division operation.

```
def division(x,y):
    try:
        div = x/y
        print(f"{x}/{y} = {div}")
    except ZeroDivisionError as e:
        print("Exception occurred!:", e)
    except Exception as e:
        print("Exception occurred!:", e)
a = int(input("Enter number a: "))
b = int(input("Enter number b: "))
division(a,b)
```

```
Enter number a: 10
Enter number b: 0
Exception occurred!: division by zero
```

16) Write a function division() that accepts two arguments. The function should be able to catch an exception such as ZeroDivisionError, ValueError, or any unknown error you might come across when you are doing a division operation. Modify the earlier “perfect division function” task in such a way that the function division() has a clean-up action.

```
def division(x,y):
    try:
        div = x/y
        print(f"{x}/{y} = {div}")
    except ZeroDivisionError as e:
        print("Exception occurred!:", e)
    except Exception as e:
        print("Exception occurred!:", e)
    finally:
        print("Working on division function.")
a = int(input("Enter number a: "))
b = int(input("Enter number b: "))
division(a,b)
```

```
 Enter number a: 10
 Enter number b: 2
 10/2 = 5.0
 Working on division function.
```

17) Write a Python program to generate a random color hex, a random alphabetical string, random value between two integers (inclusive) and a random multiple of 7 between 0 and 70. Use random.randint()

```
import random
import string
print("Generate a random color hex:")
print("#{:06x}".format(random.randint(0, 0xFFFFFF)))
print("\nGenerate a random alphabetical string:")
max_length = 255
s = ""
for i in range(random.randint(1, max_length)):
    s += random.choice(string.ascii_letters)
print(s)
print("Generate a random value between two integers, inclusive:")
print(random.randint(0, 10))
print(random.randint(-7, 7))
print(random.randint(1, 1))
print("Generate a random multiple of 7 between 0 and 70:")
print(random.randint(0, 10) * 7)
```

```
Generate a random color hex:
#8476d8

Generate a random alphabetical string:
egecMJdxgWnTuFqiiqdHhlEqixIEwfcsNLOiwwctNMbvksPtSoDaolDJtmwsjKPxYjtJZufJyEsnrelMKIVcXyFr
Generate a random value between two integers, inclusive:
4
1
1
Generate a random multiple of 7 between 0 and 70:
49
```

18) Write a Python program that generates random alphabetical characters, alphabetical strings, and alphabetical strings of a fixed length. Use random.choice()

```
import random
import string
print("Generate a random alphabetical character:")
print(random.choice(string.ascii_letters))
print("\nGenerate a random alphabetical string:")
max_length = 255
str1 = ""
for i in range(random.randint(1, max_length)):
    str1 += random.choice(string.ascii_letters)
print(str1)
print("\nGenerate a random alphabetical string of a fixed length:")
str1 = ""
for i in range(10):
    str1 += random.choice(string.ascii_letters)
print(str1)
```

```
Generate a random alphabetical character:
m

Generate a random alphabetical string:
pgbFStqTZzEuxidLSUFvHKrUaZPRRGviEcHKWDsZmgAKqeRmOzGTRsHbBAOL

Generate a random alphabetical string of a fixed length:
owXGtCzxgn
```

19) Write a Python program to construct a seeded random number generator, also generate a float between 0 and 1, excluding 1. Use random.random()

```
import random
print("Construct a seeded random number generator:")
print(random.Random().random())
print(random.Random(0).random())
```

```
print("\nGenerate a float between 0 and 1, excluding 1:")
print(random.random())
```

```
Construct a seeded random number generator:
0.21531388930732276
0.8444218515250481

Generate a float between 0 and 1, excluding 1:
0.5916651103198761
```

20) Write a Python program to create a list of random integers and randomly select multiple items from the said list. Use random.sample()

```
import random
print("Create a list of random integers:")
population = range(0, 100)
nums_list = random.sample(population, 10)
print(nums_list)
no_elements = 4
print("\nRandomly select",no_elements,"multiple items from the said list:")
result_elements = random.sample(nums_list, no_elements)
print(result_elements)
no_elements = 8
print("\nRandomly select",no_elements,"multiple items from the said list:")
result_elements = random.sample(nums_list, no_elements)
print(result_elements)
```

```
Create a list of random integers:
[26, 34, 22, 87, 92, 94, 12, 75, 10, 82]

Randomly select 4 multiple items from the said list:
[10, 22, 12, 75]

Randomly select 8 multiple items from the said list:
[12, 22, 87, 26, 10, 75, 92, 94]
```

21) Write a Python program to check if a given value is a method of a user-defined class. Use types.MethodType()

```
import types
class C:
    def x():
        return 1
    def y():
        return 1
```

```
def b():
    return 2

print(isinstance(C().x, types.MethodType))
print(isinstance(C().y, types.MethodType))
print(isinstance(b, types.MethodType))
print(isinstance(max, types.MethodType))
print(isinstance(abs, types.MethodType))
```

```
True
True
False
False
False
```

22) Write a Python program to check if a given value is compiled code or not. Also check if a given value is a module or not. Use types.CodeType, types.ModuleType()

```
import types
print("Check if a given value is compiled code:")
code = compile("print('Hello')", "sample", "exec")
print(isinstance(code, types.CodeType))
print(isinstance("print(abs(-111))", types.CodeType))
print("\nCheck if a given value is a module:")
print(isinstance(types, types.ModuleType))
```

```
Check if a given value is compiled code:
True
False

Check if a given value is a module:
True
```

23) Write a Python program to construct a Decimal from a float and a Decimal from a string. Also represent the decimal value as a tuple. Use decimal.Decimal

```
import decimal
print("Construct a Decimal from a float:")
pi_val = decimal.Decimal(3.14159)
print(pi_val)
print(pi_val.as_tuple())
print("\nConstruct a Decimal from a string:")
num_str = decimal.Decimal("123.25")
print(num_str)
print(num_str.as_tuple())
```

```
Construct a Decimal from a float:
3.14158999999999988261834005243144929409027099609375
DecimalTuple(sign=0, digits=(3, 1, 4, 1, 5, 8, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 8, 8, 2, 6,

Construct a Decimal from a string:
123.25
DecimalTuple(sign=0, digits=(1, 2, 3, 2, 5), exponent=-2)
```

24) Write a Python program to round a decimal value to the nearest multiple of 0.10, unless already an exact multiple of 0.05. Use decimal.Decimal

```
from decimal import Decimal


def round_to_10_cents(x):
    remainder = x.remainder_near(Decimal('0.10'))
    if abs(remainder) == Decimal('0.05'):
        return x
    else:
        return x - remainder

# Test code.
for x in range(80, 120):
    y = Decimal(x) / Decimal('1E2')
    print("{0} rounds to {1}".format(y, round_to_10_cents(y)))
```

```
0.80 rounds to 0.80
0.81 rounds to 0.80
0.82 rounds to 0.80
0.83 rounds to 0.80
0.84 rounds to 0.80
0.85 rounds to 0.85
0.86 rounds to 0.90
0.87 rounds to 0.90
0.88 rounds to 0.90
0.89 rounds to 0.90
0.90 rounds to 0.90
0.91 rounds to 0.90
0.92 rounds to 0.90
0.93 rounds to 0.90
0.94 rounds to 0.90
0.95 rounds to 0.95
0.96 rounds to 1.00
0.97 rounds to 1.00
0.98 rounds to 1.00
0.99 rounds to 1.00
1.00 rounds to 1.00
1.01 rounds to 1.00
1.02 rounds to 1.00
1.03 rounds to 1.00
```

```
1.04 rounds to 1.00
1.05 rounds to 1.05
1.06 rounds to 1.10
1.07 rounds to 1.10
1.08 rounds to 1.10
1.09 rounds to 1.10
1.10 rounds to 1.10
1.11 rounds to 1.10
1.12 rounds to 1.10
1.13 rounds to 1.10
1.14 rounds to 1.10
1.15 rounds to 1.15
1.16 rounds to 1.20
1.17 rounds to 1.20
1.18 rounds to 1.20
1.19 rounds to 1.20
```

25) Write a Python program to configure the rounding to round to the floor, ceiling. Use decimal.ROUND_FLOOR, decimal.ROUND_CEILING

```
import decimal
print("Configure the rounding to round to the floor:")
decimal.getcontext().prec = 4
decimal.getcontext().rounding = decimal.ROUND_FLOOR
print(decimal.Decimal(20) / decimal.Decimal(6))
print("\nConfigure the rounding to round to the ceiling:")
decimal.getcontext().prec = 4
decimal.getcontext().rounding = decimal.ROUND_CEILING
print(decimal.Decimal(20) / decimal.Decimal(6))
```

```
Configure the rounding to round to the floor:
3.333

Configure the rounding to round to the ceiling:
3.334
```

26) Write a Python program to configure rounding to round to the nearest integer, with ties going to the nearest even integer. Use decimal.ROUND_HALF_EVEN

```
import decimal
print("Configure the rounding to round to the nearest, with ties going to the nearest even in
decimal.getcontext().prec = 1
decimal.getcontext().rounding = decimal.ROUND_HALF_EVEN
print(decimal.Decimal(10) / decimal.Decimal(4))
```

```
Configure the rounding to round to the nearest, with ties going to the nearest even inte
2
```

27) Write a Python program to create a shallow copy of a given list. Use copy.copy

```
import copy
nums_x = [1, [2, 3, 4]]
print("Original list: ", nums_x)
nums_y = copy.copy(nums_x)
print("\nCopy of the said list:")
print(nums_y)
print("\nChange the value of an element of the original list:")
nums_x[1][1] = 10
print(nums_x)
print("\nSecond list:")
print(nums_y)
nums =  [[1], [2]]
nums_copy = copy.copy(nums)
print("\nOriginal list:")
print(nums)
print("\nCopy of the said list:")
print(nums_copy)
print("\nChange the value of an element of the original list:")
nums[0][0] = 0
print("\nFirst list:")
print(nums)
print("\nSecond list:")
print(nums_copy)
```

```
Original list:  [1, [2, 3, 4]]

Copy of the said list:
[1, [2, 3, 4]]

Change the value of an element of the original list:
[1, [2, 10, 4]]

Second list:
[1, [2, 10, 4]]

Original list:
[[1], [2]]

Copy of the said list:
[[1], [2]]

Change the value of an element of the original list:

First list:
[[0], [2]]
```

```
Second list:
[[0], [2]]
```

27) Write a Python program to create a deep copy of a given list. Use copy.copy

```
import copy
nums_x = [1, [2, 3, 4]]
print("Original list: ", nums_x)
nums_y = copy.deepcopy(nums_x)
print("\nDeep copy of the said list:")
print(nums_y)
print("\nChange the value of an element of the original list:")
nums_x[1][1] = 10
print(nums_x)
print("\nCopy of the second list (Deep copy):")
print(nums_y)
nums = [[1, 2, 3], [4, 5, 6]]
deep_copy = copy.deepcopy(nums)
print("\nOriginal list:")
print(nums)
print("\nDeep copy of the said list:")
print(deep_copy)
print("\nChange the value of some elements of the original list:")
nums[0][2] = 55
nums[1][1] = 77
print("\nOriginal list:")
print(nums)
print("\nSecond list (Deep copy):")
print(deep_copy)
```

```
Original list:  [1, [2, 3, 4]]

Deep copy of the said list:
[1, [2, 3, 4]]

Change the value of an element of the original list:
[1, [2, 10, 4]]

Copy of the second list (Deep copy):
[1, [2, 3, 4]]

Original list:
[[1, 2, 3], [4, 5, 6]]

Deep copy of the said list:
[[1, 2, 3], [4, 5, 6]]

Change the value of some elements of the original list:

Original list:
[[1, 2, 55], [4, 77, 6]]
```

```
Second list (Deep copy):
[[1, 2, 3], [4, 5, 6]]
```

28) Write a Python program to create a shallow copy of a given dictionary. Use copy.copy

```
import copy
nums_x = {"a":1, "b":2, 'cc':{"c":3}}
print("Original dictionary: ", nums_x)
nums_y = copy.copy(nums_x)
print("\nCopy of the said list:")
print(nums_y)
print("\nChange the value of an element of the original dictionary:")
nums_x["cc"]["c"] = 10
print(nums_x)
print("\nSecond dictionary:")
print(nums_y)

nums = {"x":1, "y":2, 'zz':{"z":3}}
nums_copy = copy.copy(nums)
print("\nOriginal dictionary :")
print(nums)
print("\nCopy of the said list:")
print(nums_copy)
print("\nChange the value of an element of the original dictionary:")
nums["zz"]["z"] = 10
print("\nFirst dictionary:")
print(nums)
print("\nSecond dictionary (copy):")
print(nums_copy)
```

```
Original dictionary:  {'a': 1, 'b': 2, 'cc': {'c': 3}}

Copy of the said list:
{'a': 1, 'b': 2, 'cc': {'c': 3}}

Change the value of an element of the original dictionary:
{'a': 1, 'b': 2, 'cc': {'c': 10}}

Second dictionary:
{'a': 1, 'b': 2, 'cc': {'c': 10}}

Original dictionary :
{'x': 1, 'y': 2, 'zz': {'z': 3}}

Copy of the said list:
{'x': 1, 'y': 2, 'zz': {'z': 3}}

Change the value of an element of the original dictionary:

First dictionary:
```

```
{'x': 1, 'y': 2, 'zz': {'z': 10}}

Second dictionary (copy):
{'x': 1, 'y': 2, 'zz': {'z': 10}}
```

29) Write a Python program to create a deep copy of a given dictionary. Use copy.copy

```
import copy
nums_x = {"a":1, "b":2, 'cc':{"c":3}}
print("Original dictionary: ", nums_x)
nums_y = copy.deepcopy(nums_x)
print("\nDeep copy of the said list:")
print(nums_y)
print("\nChange the value of an element of the original dictionary:")
nums_x["cc"]["c"] = 10
print(nums_x)
print("\nSecond dictionary (Deep copy):")
print(nums_y)

nums = {"x":1, "y":2, 'zz':{"z":3}}
nums_copy = copy.deepcopy(nums)
print("\nOriginal dictionary :")
print(nums)
print("\nDeep copy of the said list:")
print(nums_copy)
print("\nChange the value of an element of the original dictionary:")
nums["zz"]["z"] = 10
print("\nFirst dictionary:")
print(nums)
print("\nSecond dictionary (Deep copy):")
print(nums_copy)
```

```
Original dictionary:  {'a': 1, 'b': 2, 'cc': {'c': 3}}

Deep copy of the said list:
{'a': 1, 'b': 2, 'cc': {'c': 3}}

Change the value of an element of the original dictionary:
{'a': 1, 'b': 2, 'cc': {'c': 10}}

Second dictionary (Deep copy):
{'a': 1, 'b': 2, 'cc': {'c': 3}}

Original dictionary :
{'x': 1, 'y': 2, 'zz': {'z': 3}}

Deep copy of the said list:
{'x': 1, 'y': 2, 'zz': {'z': 3}}

Change the value of an element of the original dictionary:
```

```
First dictionary:
{'x': 1, 'y': 2, 'zz': {'z': 10}}

Second dictionary (Deep copy):
{'x': 1, 'y': 2, 'zz': {'z': 3}}
```

30) Write a Python GUI program to create a label and change the label font style (font name, bold, size) using tkinter module

```
import tkinter as tk
parent = tk.Tk()
parent.title("-Welcome to Python tkinter Basic exercises-")
my_label = tk.Label(parent, text="Hello", font=("Arial Bold", 70))
my_label.grid(column=0, row=0)
parent.mainloop()
```

31) Write a Python GUI program to create a window and disable to resize the window using tkinter module.

```
import tkinter as tk
parent = tk.Tk()
parent.title("-Welcome to Python tkinter Basic exercises-")
# Disable resizing the GUI
parent.resizable(0,0)
parent.mainloop()
```

32) Write a Python GUI program to create two buttons exit and hello using tkinter module.

```
import tkinter as tk

def write_text():
    print("Tkinter is easy to create GUI!")

parent = tk.Tk()
frame = tk.Frame(parent)
frame.pack()

text_disp= tk.Button(frame,
                   text="Hello",
                   command=write_text
                   )

text_disp.pack(side=tk.LEFT)
```

```
exit_button = tk.Button(frame,
                   text="Exit",
                   fg="green",
                   command=quit)
exit_button.pack(side=tk.RIGHT)

parent.mainloop()
```

33) Write a Python GUI program to create a Text widget using tkinter module. Insert a string at the beginning then insert a string into the current text. Delete the first and last character of the text.

```
import tkinter as tk

parent = tk.Tk()
# create the widget.
mytext = tk.Text(parent)

# insert a string at the beginning
mytext.insert('1.0', "- Python exercises, solution -")

# insert a string into the current text
mytext.insert('1.19', ' Practice,')

# delete the first and last character (including a newline character)
mytext.delete('1.0')
mytext.delete('end - 2 chars')
mytext.pack()
parent.mainloop()
```

34) Write a Python GUI program to create a Progress bar widgets using tkinter module.

```
import tkinter as tk
from tkinter.ttk import Progressbar
from tkinter import ttk
parent = tk.Tk()
parent.title("Progressbar")
parent.geometry('350x200')
style = ttk.Style()
style.theme_use('default')
style.configure("black.Horizontal.TProgressbar", background='green')
bar = Progressbar(parent, length=220, style='black.Horizontal.TProgressbar')
bar['value'] = 80
bar.grid(column=0, row=0)
parent.mainloop()
```

35)Write a Python GUI program to create a Progress bar widgets using tkinter module.

```
import tkinter as tk
from tkinter.ttk import Progressbar
from tkinter import ttk
parent = tk.Tk()
parent.title("Progressbar")
parent.geometry('350x200')
style = ttk.Style()
style.theme_use('default')
style.configure("black.Horizontal.TProgressbar", background='green')
bar = Progressbar(parent, length=220, style='black.Horizontal.TProgressbar')
bar['value'] = 80
bar.grid(column=0, row=0)
parent.mainloop()
```

36) Write a Python GUI program to create three radio buttons widgets using tkinter module

```
import tkinter as tk
parent = tk.Tk()
parent.title("Radiobutton ")
parent.geometry('350x200')
radio1 = tk.Radiobutton(parent, text='First', value=1)
radio2 = tk.Radiobutton(parent, text='Second', value=2)
radio3 = tk.Radiobutton(parent, text='Third', value=3)
radio1.grid(column=0, row=0)
radio2.grid(column=1, row=0)
```

36) Write a Python GUI program to create a Spinbox widget using tkinter module

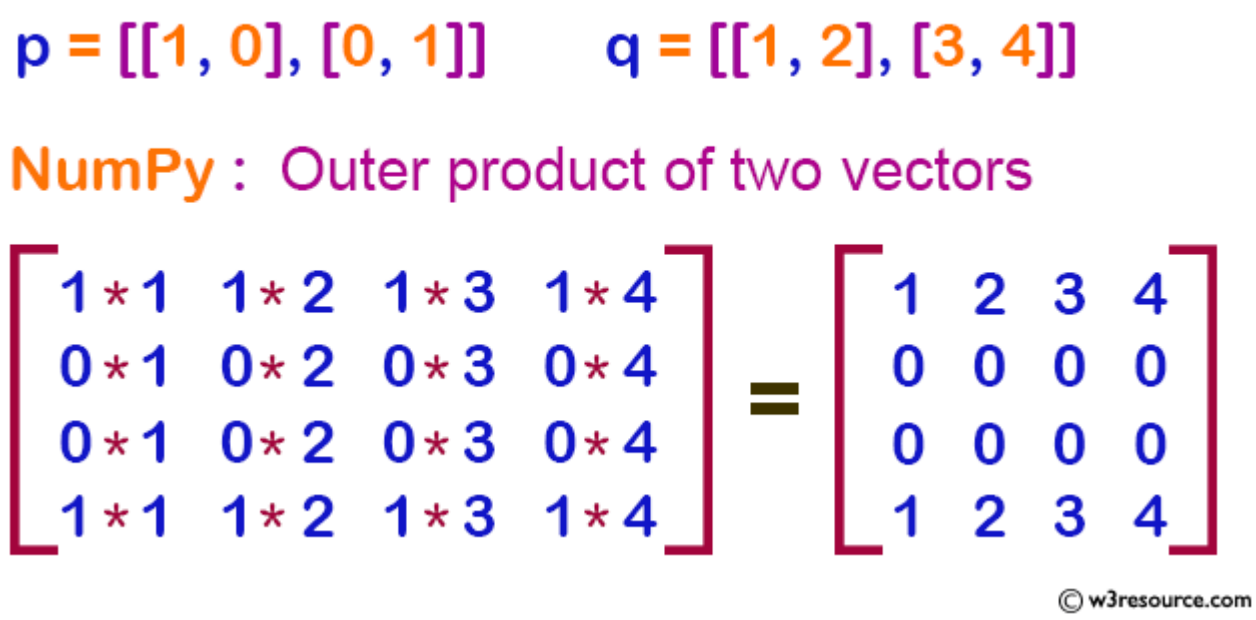
```
import tkinter as tk
root = tk.Tk()
text_var = tk.DoubleVar()

spin_box = tk.Spinbox(
    root,
    from_=0.6,
    to=50.0,
    increment=.01,
    textvariable=text_var
)
spin_box.pack()
root.mainloop()
```

37) Write a NumPy program to compute the multiplication of two given matrixes

```
import numpy as np
p = [[1, 0], [0, 1]]
q = [[1, 2], [3, 4]]
print("original matrix:")
print(p)
print(q)
result1 = np.dot(p, q)
print("Result of the said matrix multiplication:")
print(result1)
```

```
original matrix:
[[1, 0], [0, 1]]
[[1, 2], [3, 4]]
Result of the said matrix multiplication:
[[1 2]
 [3 4]]
```

38) Write a NumPy program to compute the outer product of two given vectors.




```
import numpy as np
p = [[1, 0], [0, 1]]
q = [[1, 2], [3, 4]]
print("original matrix:")
print(p)
print(q)
result = np.outer(p, q)
print("Outer product of the said two vectors:")
```

```
print(result)
```

```
original matrix:
[[1, 0], [0, 1]]
[[1, 2], [3, 4]]
Outer product of the said two vectors:
[[1 2 3 4]
 [0 0 0 0]
 [0 0 0 0]
 [1 2 3 4]]
```

39) Write a NumPy program to evaluate Einstein's summation convention of two given multidimensional arrays.

Note: In mathematics, especially in applications of linear algebra to physics, the Einstein notation or Einstein summation convention is a notational convention that implies summation over a set of indexed terms in a formula, thus achieving notational brevity.

```
import numpy as np
a = np.array([1,2,3])
b = np.array([0,1,0])
print("Original 1-d arrays:")
print(a)
print(b)
result =  np.einsum("n,n", a, b)
print("Einstein's summation convention of the said arrays:")
print(result)
x = np.arange(9).reshape(3, 3)
y = np.arange(3, 12).reshape(3, 3)
print("Original Higher dimension:")
print(x)
print(y)
result = np.einsum("mk,kn", x, y)
print("Einstein's summation convention of the said arrays:")
print(result)
```

```
Original 1-d arrays:
[1 2 3]
[0 1 0]
Einstein's summation convention of the said arrays:
2
Original Higher dimension:
[[0 1 2]
 [3 4 5]
 [6 7 8]]
[[ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
Einstein's summation convention of the said arrays:
```

```
[[ 24  27  30]
 [ 78  90 102]
 [132 153 174]]
```

40) Write a NumPy program to compute the determinant of an array.

```
import numpy as np
a = np.array([[1,2],[3,4]])
print("Original array:")
print(a)
result =  np.linalg.det(a)
print("Determinant of the said array:")
print(result)
```

```
Original array:
[[1 2]
 [3 4]]
Determinant of the said array:
-2.0000000000000004
```

41) Write a NumPy program to compute the sum of the diagonal element of a given array

```
import numpy as np
m = np.arange(6).reshape(2,3)
print("Original matrix:")
print(m)
result =  np.trace(m)
print("Condition number of the said matrix:")
print(result)
```

```
Original matrix:
[[0 1 2]
 [3 4 5]]
Condition number of the said matrix:
4
```

42) Write a NumPy program to compute the factor of a given array by Singular Value Decomposition.

```
import numpy as np
a = np.array([[1, 0, 0, 0, 2], [0, 0, 3, 0, 0], [0, 0, 0, 0, 0], [0, 2, 0, 0, 0]], dtype=np.f
print("Original array:")
print(a)
U, s, V = np.linalg.svd(a, full_matrices=False)
q, r = np.linalg.qr(a)
```

```
print("Factor of a given array  by Singular Value Decomposition:")
print("U=\n", U, "\ns=\n", s, "\nV=\n", V)
```

```
Original array:
[[1. 0. 0. 0. 2.]
 [0. 0. 3. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 2. 0. 0. 0.]]
Factor of a given array  by Singular Value Decomposition:
U=
 [[ 0.  1.  0.  0.]
 [ 1.  0.  0.  0.]
 [ 0.  0.  0. -1.]
 [ 0.  0.  1.  0.]]
s=
 [3.       2.236068 2.       0.      ]
V=
 [[-0.         0.         1.        -0.         0.       ]
 [ 0.4472136  0.         0.         0.         0.8944272]
 [-0.         1.         0.        -0.         0.       ]
 [ 0.         0.         0.         1.         0.       ]]
```

43) Write a NumPy program to calculate the Frobenius norm and the condition number of a given array.

```
import numpy as np
a = np.arange(1, 10).reshape((3, 3))
print("Original array:")
print(a)
print("Frobenius norm and the condition number:")
print(np.linalg.norm(a, 'fro'))
print(np.linalg.cond(a, 'fro'))
```

```
Original array:
[[1 2 3]
 [4 5 6]
 [7 8 9]]
Frobenius norm and the condition number:
16.881943016134134
inf
```

44) Write a NumPy program to reverse an array (the first element becomes the last)

```
import numpy as np
import numpy as np
x = np.arange(12, 38)
print("Original array:")
print(x)
```

```
print("Reverse array:")
x = x[::-1]
print(x)
```

```
Original array:
[12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
 36 37]
Reverse array:
[37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14
 13 12]
```

45) Write a NumPy program to create an 8x8 matrix and fill it with a checkerboard pattern

```
import numpy as np
x = np.ones((3,3))
print("Checkerboard pattern:")
x = np.zeros((8,8),dtype=int)
x[1::2,::2] = 1
x[::2,1::2] = 1
print(x)
```

```
Checkerboard pattern:
[[0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]]
```

46) Write a NumPy program to convert a list and tuple into arrays.

```
import numpy as np
my_list = [1, 2, 3, 4, 5, 6, 7, 8]
print("List to array: ")
print(np.asarray(my_list))
my_tuple = ([8, 4, 6], [1, 2, 3])
print("Tuple to array: ")
print(np.asarray(my_tuple))
```

```
List to array:
[1 2 3 4 5 6 7 8]
Tuple to array:
[[8 4 6]
 [1 2 3]]
```

47) Write a NumPy program to convert Centigrade degrees into Fahrenheit degrees. Centigrade values are stored in a NumPy array

```
import numpy as np
fvalues = [0, 12, 45.21, 34, 99.91, 32]
F = np.array(fvalues)
print("Values in Fahrenheit degrees:")
print(F)
print("Values in  Centigrade degrees:")
print(np.round((5*F/9 - 5*32/9),2))
```

```
Values in Fahrenheit degrees:
[ 0.   12.   45.21 34.   99.91 32.  ]
Values in  Centigrade degrees:
[-17.78 -11.11   7.34   1.11  37.73   0.  ]
```

48) Write a NumPy program to find the set difference between two arrays. The set difference will return sorted, distinct values in array1 that are not in array2.

```
import numpy as np
array1 = np.array([0, 10, 20, 40, 60, 80])
print("Array1: ",array1)
array2 = [10, 30, 40, 50, 70]
print("Array2: ",array2)
print("Unique values in array1 that are not in array2:")
print(np.setdiff1d(array1, array2))
```

```
Array1:  [ 0 10 20 40 60 80]
Array2:  [10, 30, 40, 50, 70]
Unique values in array1 that are not in array2:
[ 0 20 60 80]
```

49) Write a NumPy program to create a 5x5 matrix with row values ranging from 0 to 4

```
import numpy as np
x = np.zeros((5,5))
print("Original array:")
print(x)
print("Row values ranging from 0 to 4.")
x += np.arange(5)
print(x)
```

```
Original array:
[[0. 0. 0. 0. 0.]
```

```
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]]
Row values ranging from 0 to 4.
[[0. 1. 2. 3. 4.]
 [0. 1. 2. 3. 4.]
 [0. 1. 2. 3. 4.]
 [0. 1. 2. 3. 4.]
 [0. 1. 2. 3. 4.]]
```

50) Write a Python program to read an entire text file.

```
def file_read(fname):
        txt = open(fname)
        print(txt.read())

file_read('test.txt')
```

51) Write a Python program to read a file line by line and store it into a list

```
def file_read(fname):
        with open(fname) as f:
                #Content_list is the list that contains the read lines.
                content_list = f.readlines()
                print(content_list)

file_read(\'test.txt\')
```

52) Write a Python program to count the number of lines in a text file

```
def file_lengthy(fname):
        with open(fname) as f:
                for i, l in enumerate(f):
                        pass
        return i + 1
print("Number of lines in the file: ",file_lengthy("test.txt"))
```

53) Write a Python program to copy the contents of a file to another file

```
from shutil import copyfile
copyfile('test.py', 'abc.py')
```

54) Write a Python program that takes a text file as input and returns the number of words of a given text file.

```
def count_words(filepath):
   with open(filepath) as f:
       data = f.read()
       data.replace(",", " ")
       return len(data.split(" "))
print(count_words("words.txt"))
```

55) Write a Python program to create a file where all letters of English alphabet are listed by specified number of letters on each line.

```
import string
def letters_file_line(n):
   with open("words1.txt", "w") as f:
       alphabet = string.ascii_uppercase
       letters = [alphabet[i:i + n] + "\n" for i in range(0, len(alphabet), n)]
       f.writelines(letters)
letters_file_line(3)
```

56) Write a Pandas program to merge two given datasets using multiple join keys

```
import pandas as pd
data1 = pd.DataFrame({'key1': ['K0', 'K0', 'K1', 'K2'],
                      'key2': ['K0', 'K1', 'K0', 'K1'],
                      'P': ['P0', 'P1', 'P2', 'P3'],
                      'Q': ['Q0', 'Q1', 'Q2', 'Q3']})
data2 = pd.DataFrame({'key1': ['K0', 'K1', 'K1', 'K2'],
                       'key2': ['K0', 'K0', 'K0', 'K0'],
                       'R': ['R0', 'R1', 'R2', 'R3'],
                       'S': ['S0', 'S1', 'S2', 'S3']})
print("Original DataFrames:")
print(data1)
print("--------------------")
print(data2)
print("\nMerged Data:")
merged_data = pd.merge(data1, data2, on=['key1', 'key2'])
print(merged_data)
```

```
Original DataFrames:
  key1 key2   P   Q
0   K0   K0  P0  Q0
1   K0   K1  P1  Q1
2   K1   K0  P2  Q2
3   K2   K1  P3  Q3
--------------------
  key1 key2   R   S
0   K0   K0  R0  S0
1   K1   K0  R1  S1
2   K1   K0  R2  S2
3   K2   K0  R3  S3

Merged Data:
  key1 key2   P   Q   R   S
0   K0   K0  P0  Q0  R0  S0
1   K1   K0  P2  Q2  R1  S1
2   K1   K0  P2  Q2  R2  S2
```

57) Write a Pandas program to Combine two DataFrame objects by filling null values in one DataFrame with non-null values from other DataFrame.

```
import pandas as pd
df1 = pd.DataFrame({'A': [None, 0, None], 'B': [3, 4, 5]})
df2 = pd.DataFrame({'A': [1, 1, 3], 'B': [3, None, 3]})
df1.combine_first(df2)
print("Original DataFrames:")
print(df1)
print("--------------------")
print(df2)
print("\nMerge two dataframes with different columns:")
result = df1.combine_first(df2)
print(result)
```

```
Original DataFrames:
     A  B
0  NaN  3
1  0.0  4
2  NaN  5
--------------------
   A    B
0  1  3.0
1  1  NaN
2  3  3.0

Merge two dataframes with different columns:
     A    B
0  1.0  3.0
1  0.0  4.0
2  3.0  5.0
```

58) Write a Pandas program to join the two dataframes using the common column of both dataframes.

```
import pandas as pd
student_data1 = pd.DataFrame({
        'student_id': ['S1', 'S2', 'S3', 'S4', 'S5'],
         'name': ['Danniella Fenton', 'Ryder Storey', 'Bryce Jensen', 'Ed Bernal', 'Kwame Mor
        'marks': [200, 210, 190, 222, 199]})

student_data2 = pd.DataFrame({
        'student_id': ['S4', 'S5', 'S6', 'S7', 'S8'],
        'name': ['Scarlette Fisher', 'Carla Williamson', 'Dante Morse', 'Kaiser William', 'Ma
        'marks': [201, 200, 198, 219, 201]})

print("Original DataFrames:")
print(student_data1)
print(student_data2)
merged_data = pd.merge(student_data1, student_data2, on='student_id', how='inner')
print("Merged data (inner join):")
print(merged_data)
```

```
Original DataFrames:
  student_id              name  marks
0         S1  Danniella Fenton    200
1         S2      Ryder Storey    210
2         S3      Bryce Jensen    190
3         S4         Ed Bernal    222
4         S5       Kwame Morin    199
  student_id              name  marks
0         S4  Scarlette Fisher    201
1         S5  Carla Williamson    200
2         S6       Dante Morse    198
```

```
3         S7    Kaiser William    219
4         S8   Madeeha Preston    201
Merged data (inner join):
  student_id       name_x  marks_x            name_y  marks_y
0         S4    Ed Bernal      222  Scarlette Fisher      201
1         S5  Kwame Morin      199  Carla Williamson      200
```

59) Write a Pandas program to join (left join) the two dataframes using keys from left dataframe only.

```
import pandas as pd
data1 = pd.DataFrame({'key1': ['K0', 'K0', 'K1', 'K2'],
                      'key2': ['K0', 'K1', 'K0', 'K1'],
                      'P': ['P0', 'P1', 'P2', 'P3'],
                      'Q': ['Q0', 'Q1', 'Q2', 'Q3']})
data2 = pd.DataFrame({'key1': ['K0', 'K1', 'K1', 'K2'],
                      'key2': ['K0', 'K0', 'K0', 'K0'],
                      'R': ['R0', 'R1', 'R2', 'R3'],
                      'S': ['S0', 'S1', 'S2', 'S3']})
print("Original DataFrames:")
print(data1)
print("--------------------")
print(data2)
print("\nMerged Data (keys from data1):")
merged_data = pd.merge(data1, data2, how='left', on=['key1', 'key2'])
print(merged_data)
print("\nMerged Data (keys from data2):")
merged_data = pd.merge(data2, data1, how='left', on=['key1', 'key2'])
print(merged_data)
```

```
Original DataFrames:
  key1 key2   P   Q
0   K0   K0  P0  Q0
1   K0   K1  P1  Q1
2   K1   K0  P2  Q2
3   K2   K1  P3  Q3
--------------------
  key1 key2   R   S
0   K0   K0  R0  S0
1   K1   K0  R1  S1
2   K1   K0  R2  S2
3   K2   K0  R3  S3

Merged Data (keys from data1):
  key1 key2   P   Q    R    S
0   K0   K0  P0  Q0   R0   S0
1   K0   K1  P1  Q1  NaN  NaN
2   K1   K0  P2  Q2   R1   S1
3   K1   K0  P2  Q2   R2   S2
4   K2   K1  P3  Q3  NaN  NaN
```

✓ 0s completed at 3:36 AM