Assignment

Course Code: CSE233

Course Title: Object Oriented Programming II

Topic: Python Modules

Submitted To:

Teacher's Name: Mst. Umme Ayman(UA)

Designation: Lecturer

Department: CSE

Daffodil International University

Submitted By:

Student Name: Bishwajit Chakraborty

ID: 0242220005101414

Section: 63_I

Department: CSE

Daffodil International University

Date of submission: 2024-12-15

kmpvtn1ne

December 15, 2024

0.0.1 Pandas Problem Set-01

[16]:
```
import pandas as pd
```

[17]:
```
df = pd.read_csv("train.csv")
```

[18]:
```
print(df.head())
```

```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

[19]:
```
print(df.shape)
```

```
(891, 12)
```

[20]:
```
print(df.describe())
```

```
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   14.526497    1.102743
```

```
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   20.125000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   38.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000

            Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
min      0.000000    0.000000
25%      0.000000    7.910400
50%      0.000000   14.454200
75%      0.000000   31.000000
max      6.000000  512.329200
```

[21]:
```
print(df['Pclass'].value_counts())
```

```
Pclass
3    491
1    216
2    184
Name: count, dtype: int64
```

[22]:
```
print(df['Sex'].value_counts())
```

```
Sex
male      577
female    314
Name: count, dtype: int64
```

[23]:
```
print(df['Survived'].value_counts())
```

```
Survived
0    549
1    342
Name: count, dtype: int64
```

[24]:
```
print(df['Embarked'].value_counts())
```

```
Embarked
S    644
C    168
Q     77
Name: count, dtype: int64
```

[25]:
```
average_age = df['Age'].mean()
```

[26]:
```
num_survivors = df['Survived'].sum()
```

[27]:
```
print(df['SibSp'].value_counts())
```

```
SibSp
0    608
1    209
2     28
4     18
3     16
8      7
5      5
Name: count, dtype: int64
```

[28]:
```
print(df.isnull().sum())
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

[29]:
```
df['Age'] = df['Age'].fillna(df['Age'].median(numeric_only=True))
df.isnull().sum()
```

[29]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

[30]:
```
df.drop('Cabin', axis=1, inplace=True)
df.describe()
```

[30]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.361582	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	13.019697	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	22.000000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	35.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

[31]:
```
print(df.describe(include='all'))
```

```
        PassengerId    Survived      Pclass                 Name   Sex  \
count    891.000000  891.000000  891.000000                  891   891
unique          NaN         NaN         NaN                  891     2
top             NaN         NaN         NaN  Dooley, Mr. Patrick  male
freq            NaN         NaN         NaN                    1   577
mean     446.000000    0.383838    2.308642                  NaN   NaN
std      257.353842    0.486592    0.836071                  NaN   NaN
min        1.000000    0.000000    1.000000                  NaN   NaN
25%      223.500000    0.000000    2.000000                  NaN   NaN
50%      446.000000    0.000000    3.000000                  NaN   NaN
75%      668.500000    1.000000    3.000000                  NaN   NaN
max      891.000000    1.000000    3.000000                  NaN   NaN

                Age       SibSp       Parch Ticket        Fare Embarked
count    891.000000  891.000000  891.000000    891  891.000000      889
unique          NaN         NaN         NaN    681         NaN        3
top             NaN         NaN         NaN   1601         NaN        S
freq            NaN         NaN         NaN      7         NaN      644
mean      29.361582    0.523008    0.381594    NaN   32.204208      NaN
std       13.019697    1.102743    0.806057    NaN   49.693429      NaN
min        0.420000    0.000000    0.000000    NaN    0.000000      NaN
25%       22.000000    0.000000    0.000000    NaN    7.910400      NaN
```

```
50%      28.000000    0.000000    0.000000    NaN   14.454200      NaN
75%      35.000000    1.000000    0.000000    NaN   31.000000      NaN
max      80.000000    8.000000    6.000000    NaN  512.329200      NaN
```

[32]:
```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          891 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(4)
memory usage: 76.7+ KB
None
```

tmbxyq126

December 15, 2024

0.0.1 Numpy Problem Set

[1]:
```
import numpy as np
```

[2]:
```
arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]])
```

[3]:
```
arr.shape
```

[3]:
```
(4, 4)
```

[4]:
```
arr.size
```

[4]:
```
16
```

[5]:
```
arr.ndim
```

[5]:
```
2
```

[6]:
```
arr.dtype
```

[6]:
```
dtype('int64')
```

[7]:
```
reshaped_array = arr.reshape(2, 8)
arr.shape
```

[7]:
```
(4, 4)
```

[8]:
```
reshaped_array
```

[8]:
```
array([[ 1,  2,  3,  4,  5,  6,  7,  8],
       [ 9, 10, 11, 12, 13, 14, 15, 16]])
```

[9]:
```
arr.sum()
```

[9]:
```
np.int64(136)
```

[10]:
```
np.mean(arr)
```

[10]:
```
np.float64(8.5)
```

[11]:
```
np.median(arr)
```

[11]:
```
np.float64(8.5)
```

[12]:
```
np.std(arr)
```

[12]:
```
np.float64(4.6097722286464435)
```

[13]:
```
arr.max()
np.unravel_index(arr.argmax(), arr.shape)
```

[13]:
```
(np.int64(3), np.int64(3))
```

[14]:
```
arr.min()
np.unravel_index(arr.argmin(), arr.shape)
```

[14]:
```
(np.int64(0), np.int64(0))
```

[15]:
```
arr.sum(axis=1)
```

[15]:
```
array([10, 26, 42, 58])
```

[16]:
```
arr.sum(axis=0)
```

[16]:
```
array([28, 32, 36, 40])
```

[17]:
```
print("First row:", arr[0])
print("Last column:", arr[:, -1])
```

```
First row: [1 2 3 4]
Last column: [ 4  8 12 16]
```

[18]:
```
print("6, 7, 10, 11:\n", arr[1:3, 1:3])
```

```
6, 7, 10, 11:
 [[ 6  7]
 [10 11]]
```

[19]:
```
arr[1] = [20, 21, 22, 23]
print(arr)
```

```
[[ 1  2  3  4]
 [20 21 22 23]
 [ 9 10 11 12]
 [13 14 15 16]]
```

[20]:
```
np.cumsum(arr)
```

[20]:
```
array([  1,   3,   6,  10,  30,  51,  73,  96, 105, 115, 126, 138, 151,
       165, 180, 196])
```

[21]:
```
np.sort(arr.flatten())
```

[21]:
```
array([ 1,  2,  3,  4,  9, 10, 11, 12, 13, 14, 15, 16, 20, 21, 22, 23])
```

[22]:
```
arr.T
```

[22]:
```
array([[ 1, 20,  9, 13],
       [ 2, 21, 10, 14],
       [ 3, 22, 11, 15],
       [ 4, 23, 12, 16]])
```

[23]:
```
arr * arr
```

[23]:
```
array([[  1,   4,   9,  16],
       [400, 441, 484, 529],
       [ 81, 100, 121, 144],
       [169, 196, 225, 256]])
```

[24]:
```
arr[arr > 10]
```

[24]:
```
array([20, 21, 22, 23, 11, 12, 13, 14, 15, 16])
```

[25]:
```
arr[arr < 10] = 0
arr
```

[25]:
```
array([[ 0,  0,  0,  0],
       [20, 21, 22, 23],
       [ 0, 10, 11, 12],
       [13, 14, 15, 16]])
```

[26]:
```
np.any(arr > 15)
```

[26]:
```
np.True_
```

[27]:
```
arr
```

[27]:
```
array([[ 0,  0,  0,  0],
       [20, 21, 22, 23],
       [ 0, 10, 11, 12],
       [13, 14, 15, 16]])
```

[28]:
```
np.all(arr > 0)
```

[28]: np.False_

8d6hmt7eb

December 15, 2024

0.0.1 Pandas Problem Set-02

[1]:
```
import pandas as pd
```

[2]:
```
df = pd.read_csv('Medicaldataset.csv')
```

[3]:
```
print(df.head(5))
```

```
   Age  Gender  Heart rate  Systolic blood pressure  Diastolic blood pressure  \
0   64  Female        66.0                    160.0                      83.0
1   21  Female        94.0                     98.0                      46.0
2   55  Female        64.0                    160.0                       NaN
3   64  Female        70.0                    120.0                      55.0
4   55  Female        64.0                    112.0                      65.0

   Blood sugar  CK-MB  Troponin    Result
0        160.0   1.80     0.012  negative
1        296.0   6.75     1.060  positive
2        270.0   1.99     0.003  negative
3        270.0  13.87     0.122  positive
4        300.0   1.08     0.003  negative
```

[4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 999 entries, 0 to 998
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       999 non-null    int64
 1   Gender                    999 non-null    object
 2   Heart rate                997 non-null    float64
 3   Systolic blood pressure   993 non-null    float64
 4   Diastolic blood pressure  993 non-null    float64
 5   Blood sugar               998 non-null    float64
 6   CK-MB                     999 non-null    float64
 7   Troponin                  999 non-null    float64
 8   Result                    999 non-null    object
```

```
dtypes: float64(6), int64(1), object(2)
memory usage: 70.4+ KB
```

[5]:
```
df.isnull().sum()
```

[5]:
```
Age                         0
Gender                      0
Heart rate                  2
Systolic blood pressure     6
Diastolic blood pressure    6
Blood sugar                 1
CK-MB                       0
Troponin                    0
Result                      0
dtype: int64
```

[ ]:
```
numeric_cols = df.select_dtypes(include=['number'])
non_numeric_cols = df.select_dtypes(exclude=['number'])

numeric_cols.fillna(numeric_cols.mean(), inplace=True)

non_numeric_cols.fillna(non_numeric_cols.mode().iloc[0], inplace=True)

df = pd.concat([numeric_cols, non_numeric_cols], axis=1)
df.isnull().sum()
```

[ ]:
```
Age                         0
Heart rate                  0
Systolic blood pressure     0
Diastolic blood pressure    0
Blood sugar                 0
CK-MB                       0
Troponin                    0
Gender                      0
Result                      0
dtype: int64
```

[7]:
```
rows = df.iloc[100:111]
rows
```

[7]:

	Age	Heart rate	Systolic blood pressure	Diastolic blood pressure	\
100	71	71.0	119.0	76.0	
101	53	73.0	135.0	81.0	
102	43	68.0	116.0	74.0	
103	66	70.0	113.0	62.0	
104	67	87.0	148.0	89.0	
105	51	85.0	140.0	82.0	

```
106   50        83.0                    140.0                    81.0
107   67        82.0                    164.0                    90.0
108   59        81.0                    150.0                    51.0
109   20        60.0                    156.0                    60.0
110   55        67.0                    192.0                    56.0
```

```
     Blood sugar    CK-MB  Troponin  Gender    Result
100        159.0    0.468     0.029  Female  positive
101        115.0  165.100     0.014  Female  positive
102         81.0    1.640     0.015    Male  positive
103        266.0  300.000     0.012    Male  positive
104        142.0    1.870     0.010  Female  negative
105        101.0    1.690     0.008    Male  negative
106        244.0    3.270     2.230  Female  positive
107        130.0    3.750     0.009  Female  negative
108        117.0    1.510     1.550  Female  positive
109        103.0    5.220     1.840  Female  positive
110        120.0    2.160     0.011  Female  negative
```

[ ]:
```
df['Gender'] = pd.to_numeric(df['Gender'].replace({'Male': 0, 'Female': 1}),␣
  ↪downcast='integer')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 999 entries, 0 to 998
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       999 non-null    int64
 1   Heart rate                999 non-null    float64
 2   Systolic blood pressure   999 non-null    float64
 3   Diastolic blood pressure  999 non-null    float64
 4   Blood sugar               999 non-null    float64
 5   CK-MB                     999 non-null    float64
 6   Troponin                  999 non-null    float64
 7   Gender                    999 non-null    int8
 8   Result                    999 non-null    object
dtypes: float64(6), int64(1), int8(1), object(1)
memory usage: 63.5+ KB
```

```
/tmp/ipykernel_810795/1022457581.py:1: FutureWarning: Downcasting behavior in
`replace` is deprecated and will be removed in a future version. To retain the
old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to
the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
  df['Gender'] = pd.to_numeric(df['Gender'].replace({'Male': 0, 'Female': 1}),
downcast='integer')
```

[9]:
```
newDF = df.copy()
```

[ ]:
```
newDF.describe()
```

[ ]:
```
                Age   Heart rate  Systolic blood pressure  \
count  999.000000   999.000000               999.000000
mean    56.333333    77.763290               127.313192
std     13.229817    48.625292                25.800134
min     14.000000    20.000000                42.000000
25%     48.000000    64.000000               110.000000
50%     58.000000    74.000000               125.000000
75%     65.000000    84.500000               143.000000
max    100.000000  1111.000000               223.000000

       Diastolic blood pressure  Blood sugar       CK-MB    Troponin  \
count                999.000000   999.000000  999.000000  999.000000
mean                  72.145015   147.123948   15.989179    0.345756
std                   13.865411    73.416461   47.760004    1.026621
min                   38.000000    35.000000    0.321000    0.001000
25%                   62.000000    98.000000    1.655000    0.006000
50%                   72.000000   117.000000    2.910000    0.014000
75%                   81.000000   174.000000    5.815000    0.099500
max                  128.000000   541.000000  300.000000   10.300000

           Gender
count  999.000000
mean     0.667668
std      0.471285
min      0.000000
25%      0.000000
50%      1.000000
75%      1.000000
max      1.000000
```

[11]:
```
filter_patients = newDF[(newDF['Age'] > 50) & (newDF['Result'] == 'Negative')]
filter_patients
```

[11]:
```
Empty DataFrame
Columns: [Age, Heart rate, Systolic blood pressure, Diastolic blood pressure,
Blood sugar, CK-MB, Troponin, Gender, Result]
Index: []
```

[12]:
```
numeric_df = newDF.select_dtypes(include=['number'])
correlation_matrix = numeric_df.corr()
correlation_matrix
```

[12]:
```
                             Age  Heart rate  Systolic blood pressure  \
Age                     1.000000    0.010721                 0.025219
Heart rate              0.010721    1.000000                 0.008797
```

```
Systolic blood pressure   0.025219    0.008797                1.000000
Diastolic blood pressure  0.015241    0.071444                0.605781
Blood sugar               0.001028   -0.008874                0.028258
CK-MB                     0.002831   -0.015889               -0.010500
Troponin                  0.055168    0.028176                0.041078
Gender                   -0.055979   -0.017690                0.001955

                          Diastolic blood pressure  Blood sugar     CK-MB  \
Age                                       0.015241    0.001028  0.002831
Heart rate                                0.071444   -0.008874 -0.015889
Systolic blood pressure                   0.605781    0.028258 -0.010500
Diastolic blood pressure                  1.000000   -0.011773 -0.025068
Blood sugar                              -0.011773    1.000000  0.038670
CK-MB                                    -0.025068    0.038670  1.000000
Troponin                                  0.036813    0.027697 -0.006962
Gender                                   -0.031566    0.011713  0.019584

                          Troponin    Gender
Age                       0.055168 -0.055979
Heart rate                0.028176 -0.017690
Systolic blood pressure   0.041078  0.001955
Diastolic blood pressure  0.036813 -0.031566
Blood sugar               0.027697  0.011713
CK-MB                    -0.006962  0.019584
Troponin                  1.000000  0.103128
Gender                    0.103128  1.000000
```

[13]:
```
newDF['Heart rate'].value_counts()
```

[13]:
```
Heart rate
60.0     71
61.0     42
70.0     38
64.0     37
80.0     37
         ..
104.0     1
49.0      1
46.0      1
135.0     1
20.0      1
Name: count, Length: 78, dtype: int64
```

[14]:
```
newDF['Age'].value_counts()
```

[14]:
```
Age
60     86
```

```
70     56
50     55
63     49
55     48
       ..
91      1
88      1
100     1
23      1
14      1
Name: count, Length: 72, dtype: int64
```

[15]:
```
newDF.to_excel('new_medical_dataset.xlsx', index=False)
```

Creating and Using a Custom Module

```python
import math
def add(a, b):


    return a + b


def subtract(a, b):

    return a - b


def multiply(a, b):

    return a * b


def divide(a, b):
    if b == 0:
        return "Error: Division by zero is not allowed."
    return a / b


def factorial(n):
    return math.factorial(n)
```

```python
def power(a, b):
    return math.pow(a, b)
```

```python
import math_operations



print("Addition of 10 and 5:", math_operations.add(14, 14))
print("Subtraction of 10 and 5:", math_operations.subtract(14, 14))
print("Multiplication of 10 and 5:", math_operations.multiply(14, 14))
print("Division of 10 by 5:", math_operations.divide(14, 14))
print("Division of 10 by 0:", math_operations.divide(14, 0))
```

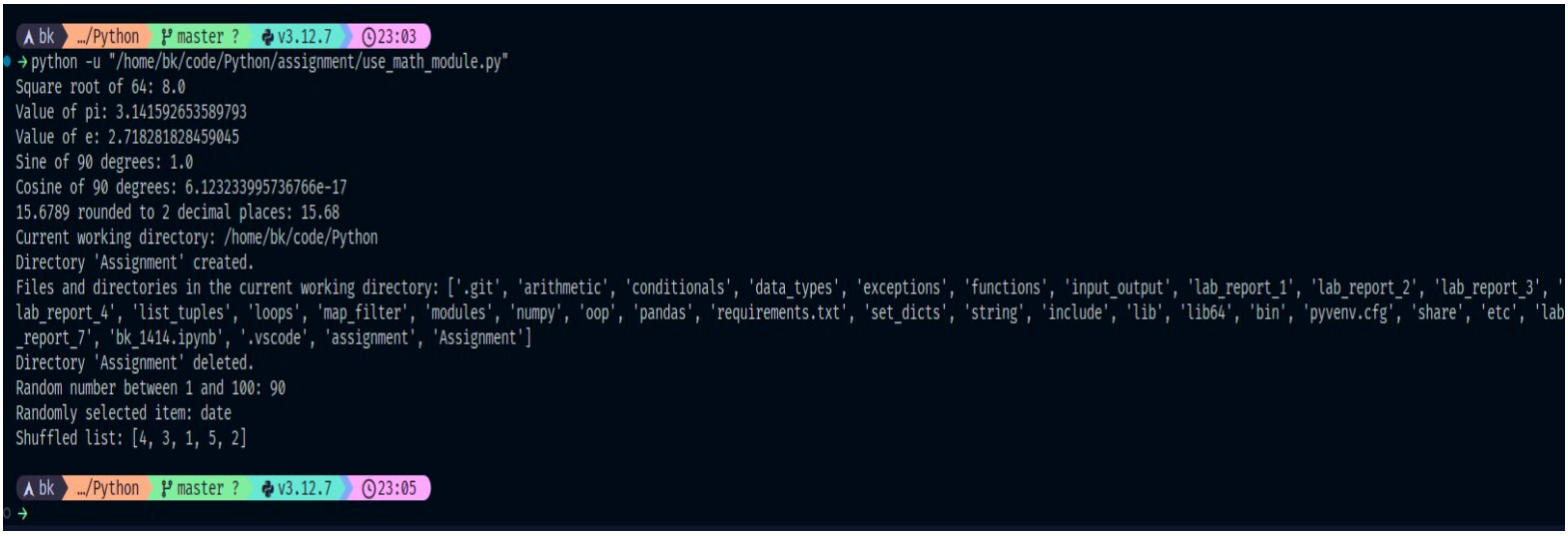
```
bk .../Python master ? v3.12.7 23:03
→ python -u "/home/bk/code/Python/assignment/use_math_operations.py"
Addition of 10 and 5: 28
Subtraction of 10 and 5: 0
Multiplication of 10 and 5: 196
Division of 10 by 5: 1.0
Division of 10 by 0: Error: Division by zero is not allowed.

bk .../Python master ? v3.12.7 23:03
→
```

Exploring Built-In Modules

```
import math


sqrt_64 = math.sqrt(64)
print("Square root of 64:", sqrt_64)


pi_value = math.pi
e_value = math.e
print("Value of pi:", pi_value)
print("Value of e:", e_value)


sine_90 = math.sin(math.radians(90))
cosine_90 = math.cos(math.radians(90))
print("Sine of 90 degrees:", sine_90)
print("Cosine of 90 degrees:", cosine_90)


rounded_value = round(15.6789, 2)
print("15.6789 rounded to 2 decimal places:", rounded_value)


import os
```

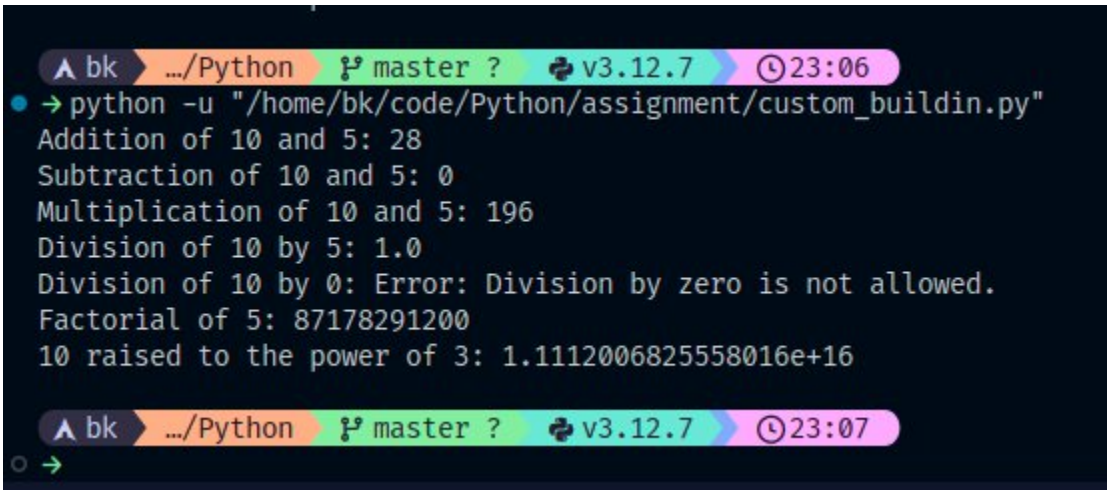
```python
cwd = os.getcwd()

print("Current working directory:", cwd)



try:

    os.mkdir("Assignment")

    print("Directory 'Assignment' created.")

except FileExistsError:

    print("Directory 'Assignment' already exists.")



entries = os.listdir(cwd)

print("Files and directories in the current working directory:",
entries)



try:

    os.rmdir("Assignment")

    print("Directory 'Assignment' deleted.")

except FileNotFoundError:

    print("Directory 'Assignment' does not exist.")

except OSError:

    print("Directory 'Assignment' is not empty or cannot be
deleted.")



import random
```

```python
random_number = random.randint(1, 100)
print("Random number between 1 and 100:", random_number)



items = ['apple', 'banana', 'cherry', 'date']
random_item = random.choice(items)
print("Randomly selected item:", random_item)



numbers = [1, 2, 3, 4, 5]
random.shuffle(numbers)
print("Shuffled list:", numbers)
```



Combining Custom and Built-In Modules

```python
import math_operations



print("Addition of 10 and 5:", math_operations.add(14, 14))

print("Subtraction of 10 and 5:", math_operations.subtract(14, 14))

print("Multiplication of 10 and 5:", math_operations.multiply(14,
14))

print("Division of 10 by 5:", math_operations.divide(14, 14))

print("Division of 10 by 0:", math_operations.divide(14, 0))

print("Factorial of 5:", math_operations.factorial(14))

print("10 raised to the power of 3:", math_operations.power(14,
14))
```



Date Module in Python

```python
import math


def add(a, b):
    return a + b


def subtract(a, b):
    return a - b


def multiply(a, b):
    return a * b


def divide(a, b):
    if b == 0:
        return "Error: Division by zero"
    return a / b


def factorial(n):
    return math.factorial(n)


def power(a, b):
```

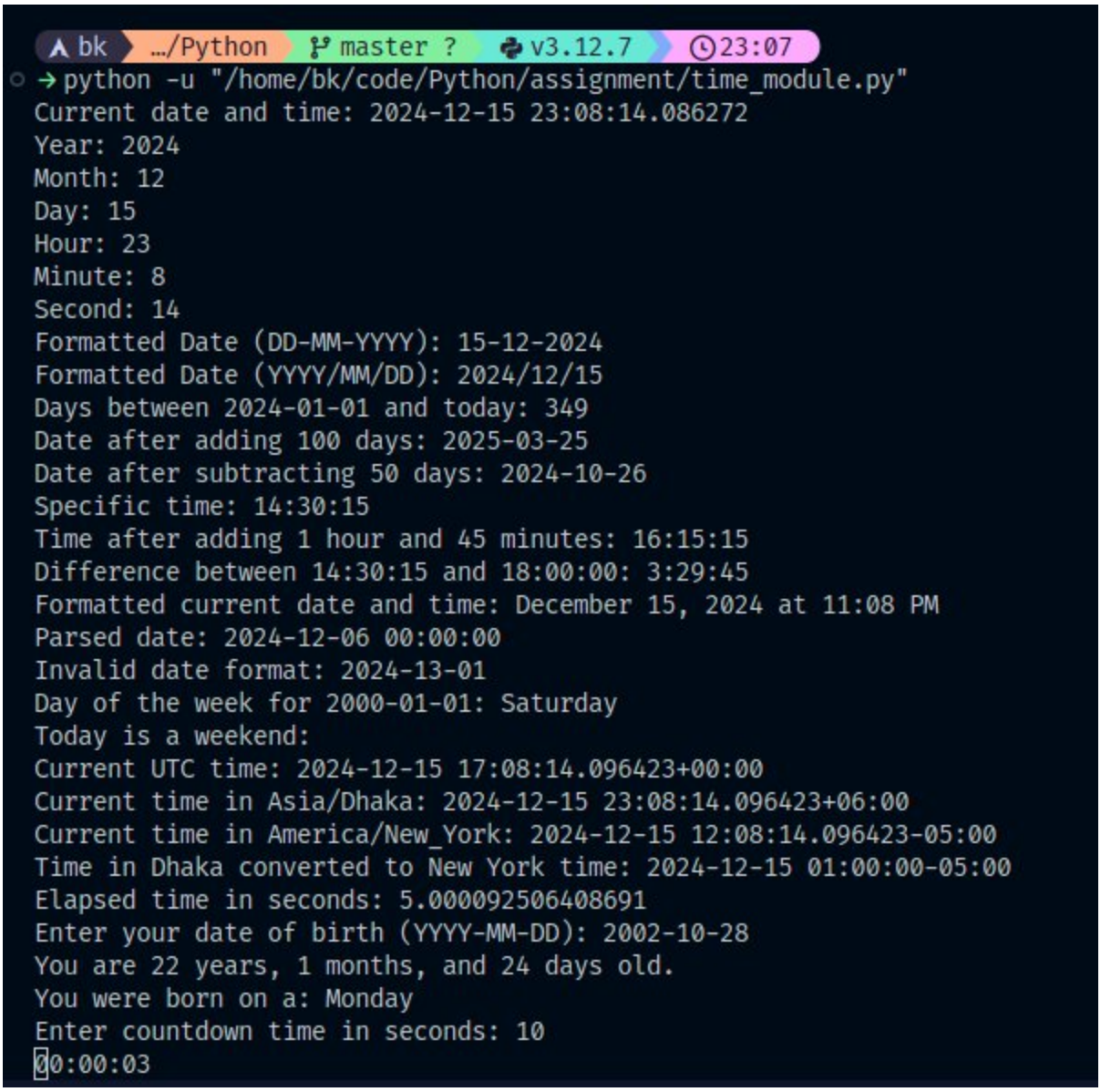
```
    return math.pow(a, b)


from datetime import datetime


now = datetime.now()
print("Current date and time:", now)
print("Year:", now.year)
print("Month:", now.month)
print("Day:", now.day)
print("Hour:", now.hour)
print("Minute:", now.minute)
print("Second:", now.second)
print("Formatted Date (DD-MM-YYYY):", now.strftime("%d-%m-%Y"))
print("Formatted Date (YYYY/MM/DD):", now.strftime("%Y/%m/%d"))


from datetime import timedelta, date


date1 = date(2024, 1, 1)
today = date.today()
days_between = (today - date1).days
print("Days between 2024-01-01 and today:", abs(days_between))


add_100_days = today + timedelta(days=100)
```

```
print("Date after adding 100 days:", add_100_days)



subtract_50_days = today - timedelta(days=50)

print("Date after subtracting 50 days:", subtract_50_days)



from datetime import time



specific_time = time(14, 30, 15)

print("Specific time:", specific_time)



new_time = (datetime.combine(date.today(), specific_time) +
timedelta(hours=1, minutes=45)).time()

print("Time after adding 1 hour and 45 minutes:", new_time)



time1 = datetime.combine(date.today(), time(14, 30, 15))

time2 = datetime.combine(date.today(), time(18, 0, 0))

time_difference = time2 - time1

print("Difference between 14:30:15 and 18:00:00:", time_difference)



formatted_date = now.strftime("%B %d, %Y at %I:%M %p")

print("Formatted current date and time:", formatted_date)
```

```
string_date = "06-12-2024"
parsed_date = datetime.strptime(string_date, "%d-%m-%Y")
print("Parsed date:", parsed_date)


invalid_date = "2024-13-01"
try:
    validated_date = datetime.strptime(invalid_date, "%Y-%m-%d")
    print("Valid date:", validated_date)
except ValueError:
    print("Invalid date format:", invalid_date)


input_date = date(2000, 1, 1)
day_of_week = input_date.strftime("%A")
print("Day of the week for 2000-01-01:", day_of_week)


is_weekend = today.weekday() >= 5
print("Today is a weekend:" if is_weekend else "Today is a
weekday.")


from pytz import timezone, all_timezones
import pytz


utc_now = datetime.now(pytz.utc)
```

```python
print("Current UTC time:", utc_now)


for tz in ["Asia/Dhaka", "America/New_York"]:
    tz_time = utc_now.astimezone(timezone(tz))
    print(f"Current time in {tz}:", tz_time)


source_tz = timezone("Asia/Dhaka")
dest_tz = timezone("America/New_York")
source_time = source_tz.localize(datetime(2024, 12, 15, 12, 0, 0))
converted_time = source_time.astimezone(dest_tz)
print("Time in Dhaka converted to New York time:", converted_time)


import time
start_time = time.time()
time.sleep(5)
end_time = time.time()
elapsed_time = end_time - start_time
print("Elapsed time in seconds:", elapsed_time)


dob_input = input("Enter your date of birth (YYYY-MM-DD): ")
try:
    dob = datetime.strptime(dob_input, "%Y-%m-%d").date()
    age = today - dob
```

```
    years = age.days // 365
    months = (age.days % 365) // 30
    days = (age.days % 365) % 30
    print(f"You are {years} years, {months} months, and {days} days old.")
    print("You were born on a:", dob.strftime("%A"))
except ValueError:
    print("Invalid date format.")



countdown_seconds = int(input("Enter countdown time in seconds: "))
while countdown_seconds:
    mins, secs = divmod(countdown_seconds, 60)
    hours, mins = divmod(mins, 60)
    print(f"{hours:02}:{mins:02}:{secs:02}", end="\r")
    time.sleep(1)
    countdown_seconds -= 1
print("Time's up!")
```


bk
.../Python
master ?
v3.12.7
23:07
→ python -u "/home/bk/code/Python/assignment/time_module.py"
Current date and time: 2024-12-15 23:08:14.086272
Year: 2024
Month: 12
Day: 15
Hour: 23
Minute: 8
Second: 14
Formatted Date (DD-MM-YYYY): 15-12-2024
Formatted Date (YYYY/MM/DD): 2024/12/15
Days between 2024-01-01 and today: 349
Date after adding 100 days: 2025-03-25
Date after subtracting 50 days: 2024-10-26
Specific time: 14:30:15
Time after adding 1 hour and 45 minutes: 16:15:15
Difference between 14:30:15 and 18:00:00: 3:29:45
Formatted current date and time: December 15, 2024 at 11:08 PM
Parsed date: 2024-12-06 00:00:00
Invalid date format: 2024-13-01
Day of the week for 2000-01-01: Saturday
Today is a weekend:
Current UTC time: 2024-12-15 17:08:14.096423+00:00
Current time in Asia/Dhaka: 2024-12-15 23:08:14.096423+06:00
Current time in America/New_York: 2024-12-15 12:08:14.096423-05:00
Time in Dhaka converted to New York time: 2024-12-15 01:00:00-05:00
Elapsed time in seconds: 5.000092506408691
Enter your date of birth (YYYY-MM-DD): 2002-10-28
You are 22 years, 1 months, and 24 days old.
You were born on a: Monday
Enter countdown time in seconds: 10
00:00:03