Exception Handling

```
#we are only trapping error, not what type of error
a=int(input("Enter a number: "))
b=int(input("Enter another number: "))
c=a/b
print("C is", c)
print("Code is running")
print("Coded by amarpanchal.education")
```

```
Enter a number: 9
Enter another number: 0
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-2-e3ce7bb6bea7> in <cell line: 4>()
      2 a=int(input("Enter a number: "))
      3 b=int(input("Enter another number: "))
----> 4 c=a/b
      5 print("C is", c)
      6 print("Code is running")

ZeroDivisionError: division by zero
```

SEARCH STACK OVERFLOW

```
#we are only trapping error, not what type of error
a=int(input("Enter a number: "))
b=int(input("Enter another number: "))
try:
    c=a/b
except Exception:
    print("Exception")
else:
    print("C is", c)
finally:
    print("by Amar sir")
    # part of code after handling exception
print("still code is running")
```

```
Enter a number: 9
Enter another number: 0
Exception
by Amar sir
still code is running
```

```
#we are trapping error as well as the type of error
try:
   a=int(input("Enter a number: "))
   b=int(input("Enter another number: "))
   c=a/b
except Exception as arg:
```

```
    print("Exception:", arg)
else:
    print("C is", c)
finally:
    print("by Amar sir")
    # part of code after handling exception
print("still code is running")
```

```
Enter a number: a
Exception: invalid literal for int() with base 10: 'a'
by Amar sir
still code is running
```

```
#executing with only 'try' & 'finally' blocks which are mandatory
# creates a risk that it cannot catch error
try:
    a=int(input("Enter a number: "))
    b=int(input("Enter another number: "))
    c=a/b
    print("C is", c)
finally:
    print("finally executed")
    # part of code after handling exception
print("still trying to live life")
print("alive and kicking")
```

```
Enter a number: 7
Enter another number: 0
finally executed
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-16-11571f439cdb> in <cell line: 3>()
      4     a=int(input("Enter a number: "))
      5     b=int(input("Enter another number: "))
----> 6     c=a/b
      7     print("C is", c)
      8 finally:

ZeroDivisionError: division by zero
```

SEARCH STACK OVERFLOW

```
# create custom exception
class AgeException(Exception): # inherit super class Exception to create custom exception
    def __init__(self, data):   # accept the exception which is raised
        self.data=data
    def __str__(self):  # throws the exception
        return "Exception : "+str(self.data)+" Under 18"

try:
    n=int(input("Enter age: "))
    if n>=18:
        print("You can vote, your age is", n)
    else:
        raise AgeException(n)   # raises exception
except Exception as msg:    # excepting exception to throw it
```

```
    print(msg)
finally:
    print("code is executing at last step, now stopping")
```

```
Enter age: 17
Exception : 17 Under 18
code is executing at last step, now stopping
```

pandas

```
import pandas as pd
```

```
s=pd.Series([1, 2, 3, 4, 5, 6, 7, 8])
```

```
type(s)
print(s)
```

```
0    1
1    2
2    3
3    4
4    5
5    6
6    7
7    8
dtype: int64
```

```
for i in s:
    print(i) # elements of series can be accessed using loop
```

```
1
2
3
4
5
6
7
8
```

```
for i in range(0, len(s)):
    print("at", i, "we have", s[i]) # index based access is allowed in series
```

```
at 0 we have 1
at 1 we have 2
at 2 we have 3
at 3 we have 4
at 4 we have 5
at 5 we have 6
at 6 we have 7
at 7 we have 8
```

```
s
```

```
0    1
1    2
2    3
3    4
4    5
5    6
6    7
7    8
dtype: int64
```

```
s*2 # scalar operation allowed
```

```
0     2
1     4
2     6
3     8
4    10
5    12
6    14
7    16
dtype: int64
```

```
s/2 # scalar operation allowed
```

```
0    0.5
1    1.0
2    1.5
3    2.0
4    2.5
5    3.0
6    3.5
7    4.0
dtype: float64
```

```
s+2 # scalar operation allowed
```

```
0     3
1     4
2     5
3     6
4     7
5     8
6     9
7    10
dtype: int64
```

```
s-2 # scalar operation allowed
```

```
0   -1
1    0
2    1
3    2
4    3
5    4
6    5
7    6
dtype: int64
```

```
1 s
```

```
0    1
1    2
2    3
3    4
4    5
5    6
6    7
7    8
dtype: int64
```

```
1 s*10
```

```
0    10
1    20
2    30
3    40
4    50
5    60
6    70
7    80
dtype: int64
```

```
1 s
```

```
0    1
1    2
2    3
3    4
4    5
5    6
6    7
7    8
dtype: int64
```

```
1 s[3] # allows index based access
```

```
4
```

```
1 s[-1] # does not support negative index
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/usr/local/lib/python3.9/dist-packages/pandas/core/indexes/range.py in get_loc(self, key, method, tolerance)
    384                 try:
--> 385                     return self._range.index(new_key)
    386                 except ValueError as err:

ValueError: -1 is not in range

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

```
len(s)
```

```
8
```

```
    385                     return self._range.index(new_key)
```

```
max(s)
```

```
8
```

```
min(s)
```

```
1
```

```
sum(s)
```

```
36
```

```
s=pd.Series([1, 2, 3, 4])
s2=pd.Series([10, 20, 30, 40, 50])
s3=s+s2
```

```
print(s3) # NaN:NotaNumber, would add only existing values
```

```
0    11.0
1    22.0
2    33.0
3    44.0
4     NaN
dtype: float64
```

```
s[0:4]
```

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

```
from google.colab import files
uploaded=files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving mycsv.csv to mycsv.csv

```
import pandas as pd
import io
df=pd.read_csv(io.BytesIO(uploaded['mycsv.csv'])) # creates data frame; data frame is immutable, it needs to be saved
print(df)
```

```
    id   name  gender   salary
0    1   aaaa    male    10000
1    2   bbbb  female     8000
2    3  ccccc    male   120000
3    4   dddd    male    45000
4    5   eeee  female   334567
5    6   ffff  female    21234
6    7   gggg  female     2345
7    8   hhhh  female    23456
8    9   iiii    male     7654
9   10    jjj   male   3456578
10  11   kkkk    male    45678
11  12   llll    male      345
12  13   mmmm  female     9876
13  14  nnnnn    male    34567
14  15   jjjj    male    87654
15  16   iiii  female    34567
16  17   mmmm  female     1234
17  18  nnnnn  female    23456
18  19   oooo  female     5555
19  20   pppp    male     3455
```

```
print(df.name) # to print field 'name' as serialized data
```

```
0      aaaa
1      bbbb
2     ccccc
3      dddd
4      eeee
5      ffff
6      gggg
7      hhhh
8      iiii
9       jjj
10     kkkk
11     llll
12     mmmm
13    nnnnn
14     jjjj
15     iiii
16     mmmm
17    nnnnn
18     oooo
19     pppp
Name: name, dtype: object
```

```
print(df.name, df.salary) # to print fields 'name' & 'salary' as serialized data
```

```
0      aaaa
1      bbbb
2     ccccc
3      dddd
4      eeee
```

```
5      ffff
6      gggg
7      hhhh
8      iiii
9       jjj
10     kkkk
11     llll
12     mmmm
13    nnnnn
14     jjjj
15     iiii
16     mmmm
17    nnnnn
18     oooo
19     pppp
Name: name, dtype: object 0       10000
1        8000
2      120000
3       45000
4      334567
5       21234
6        2345
7       23456
8        7654
9     3456578
10      45678
11        345
12       9876
13      34567
14      87654
15      34567
16       1234
17      23456
18       5555
19       3455
Name: salary, dtype: int64
```

```
df[['name', 'salary']] # to print fields 'name' & 'salary' as data frame
```

	name	salary
0	aaaa	10000
1	bbbb	8000
2	ccccc	120000
3	dddd	45000
4	eeee	334567
5	ffff	21234
6	gggg	2345
7	hhhh	23456

```
df[['salary', 'name']] # changed the view/order or fields
```

	salary	name
0	10000	aaaa
1	8000	bbbb
2	120000	ccccc
3	45000	dddd
4	334567	eeee
5	21234	ffff
6	2345	gggg
7	23456	hhhh
8	7654	iiii
9	3456578	jjj
10	45678	kkkk
11	345	llll
12	9876	mmmm
13	34567	nnnnn
14	87654	jjjj
15	34567	iiii
16	1234	mmmm
17	23456	nnnnn
18	5555	oooo
19	3455	pppp

```
df
```

	id	name	gender	salary
0	1	aaaa	male	10000
1	2	bbbb	female	8000
2	3	ccccc	male	120000
3	4	dddd	male	45000
4	5	eeee	female	334567
5	6	ffff	female	21234
6	7	gggg	female	2345
7	8	hhhh	female	23456
8	9	iiii	male	7654
9	10	jjj	male	3456578
10	11	kkkk	male	45678
11	12	llll	male	345
12	13	mmmm	female	9876
13	14	nnnnn	male	34567
14	15	jjjj	male	87654
15	16	iiii	female	34567
16	17	mmmm	female	1234
17	18	nnnnn	female	23456
18	19	oooo	female	5555
19	20	pppp	male	3455

```
1 df.salary*0.10
```

```
0       1000.0
1        800.0
2      12000.0
3       4500.0
4      33456.7
5       2123.4
6        234.5
7       2345.6
8        765.4
9     345657.8
10      4567.8
11        34.5
12       987.6
13      3456.7
14      8765.4
15      3456.7
16       123.4
17      2345.6
18       555.5
```

```
19       345.5
Name: salary, dtype: float64
```

```
1
```

```
1 df['tax']=df.salary*0.10 # adds another field in data frame
```

```
1 df
```

	id	name	gender	salary	tax
0	1	aaaa	male	10000	1000.0
1	2	bbbb	female	8000	800.0
2	3	ccccc	male	120000	12000.0
3	4	dddd	male	45000	4500.0
4	5	eeee	female	334567	33456.7
5	6	ffff	female	21234	2123.4
6	7	gggg	female	2345	234.5
7	8	hhhh	female	23456	2345.6
8	9	iiii	male	7654	765.4
9	10	jjj	male	3456578	345657.8
10	11	kkkk	male	45678	4567.8
11	12	llll	male	345	34.5
12	13	mmmm	female	9876	987.6
13	14	nnnnn	male	34567	3456.7
14	15	jjjj	male	87654	8765.4
15	16	iiii	female	34567	3456.7
16	17	mmmm	female	1234	123.4
17	18	nnnnn	female	23456	2345.6
18	19	oooo	female	5555	555.5
19	20	pppp	male	3455	345.5

```
1 # add one more col in df, named net salary where net salary=salary-tax
2 df['net salary']=df.salary-df.tax
```

```
1 df
```

	id	name	gender	salary	tax	net salary
0	1	aaaa	male	10000	1000.0	9000.0
1	2	bbbb	female	8000	800.0	7200.0
2	3	ccccc	male	120000	12000.0	108000.0
3	4	dddd	male	45000	4500.0	40500.0
4	5	eeee	female	334567	33456.7	301110.3
5	6	ffff	female	21234	2123.4	19110.6
6	7	gggg	female	2345	234.5	2110.5
7	8	hhhh	female	23456	2345.6	21110.4
8	9	iiii	male	7654	765.4	6888.6
9	10	jjj	male	3456578	345657.8	3110920.2
10	11	kkkk	male	45678	4567.8	41110.2
11	12	llll	male	345	34.5	310.5
12	13	mmmm	female	9876	987.6	8888.4
13	14	nnnnn	male	34567	3456.7	31110.3
14	15	jjjj	male	87654	8765.4	78888.6
15	16	iiii	female	34567	3456.7	31110.3
16	17	mmmm	female	1234	123.4	1110.6
17	18	nnnnn	female	23456	2345.6	21110.4

```
df.loc[4] # lists one record from data frame
```

```
id                 5
name            eeee
gender          female
salary          334567
tax            33456.7
net salary    301110.3
Name: 4, dtype: object
```

```
# list entire dataframe records
for i in range(0, len(df)):
    print(df.loc[i])
```

```
id                1
name           aaaa
gender         male
salary        10000
tax          1000.0
net salary   9000.0
Name: 0, dtype: object
id                2
name           bbbb
gender       female
```

```
salary          8000
tax            800.0
net salary    7200.0
Name: 1, dtype: object
id                  3
name            ccccc
gender           male
salary         120000
tax           12000.0
net salary   108000.0
Name: 2, dtype: object
id                  4
name             dddd
gender           male
salary          45000
tax            4500.0
net salary    40500.0
Name: 3, dtype: object
id                  5
name             eeee
gender         female
salary         334567
tax           33456.7
net salary   301110.3
Name: 4, dtype: object
id                  6
name             ffff
gender         female
salary          21234
tax            2123.4
net salary    19110.6
Name: 5, dtype: object
id                  7
name            gggg
gender        female
salary          2345
tax            234.5
net salary    2110.5
Name: 6, dtype: object
id                  8
name             hhhh
gender         female
salary          23456
tax            2345.6
net salary    21110.4
Name: 7, dtype: object
id                  9
```

```
df.to_csv('final.csv') # to create a new csv file in virtual local storage, so you can download
```

```
df.to_sql("data.sql")
```

```
df
```

	id	name	gender	salary	tax	net salary
0	1	aaaa	male	10000	1000.0	9000.0
1	2	bbbb	female	8000	800.0	7200.0
2	3	ccccc	male	120000	12000.0	108000.0
3	4	dddd	male	45000	4500.0	40500.0
4	5	eeee	female	334567	33456.7	301110.3
5	6	ffff	female	21234	2123.4	19110.6
6	7	gggg	female	2345	234.5	2110.5
7	8	hhhh	female	23456	2345.6	21110.4
8	9	iiii	male	7654	765.4	6888.6
9	10	jjj	male	3456578	345657.8	3110920.2
10	11	kkkk	male	45678	4567.8	41110.2
11	12	llll	male	345	34.5	310.5
12	13	mmmm	female	9876	987.6	8888.4
13	14	nnnnn	male	34567	3456.7	31110.3
14	15	jjjj	male	87654	8765.4	78888.6
15	16	iiii	female	34567	3456.7	31110.3
16	17	mmmm	female	1234	123.4	1110.6
17	18	nnnnn	female	23456	2345.6	21110.4
18	19	oooo	female	5555	555.5	4999.5
19	20	pppp	male	3455	345.5	3109.5

```
# read first 5 records from dataframe
df.head(5)
```

	id	name	gender	salary	tax	net salary
0	1	aaaa	male	10000	1000.0	9000.0
1	2	bbbb	female	8000	800.0	7200.0
2	3	ccccc	male	120000	12000.0	108000.0
3	4	dddd	male	45000	4500.0	40500.0
4	5	eeee	female	334567	33456.7	301110.3

```
# read last 5 records from dataframe
df.tail(5)
```

	id	name	gender	salary	tax	net salary
15	16	iiii	female	34567	3456.7	31110.3
16	17	mmmm	female	1234	123.4	1110.6
17	18	nnnnn	female	23456	2345.6	21110.4
18	19	oooo	female	5555	555.5	4999.5
19	20	pppp	male	3455	345.5	3109.5

```
# print 5th record to 15th record
df.loc[4:14] # .loc [0, n]
```

	id	name	gender	salary	tax	net salary
4	5	eeee	female	334567	33456.7	301110.3
5	6	ffff	female	21234	2123.4	19110.6
6	7	gggg	female	2345	234.5	2110.5
7	8	hhhh	female	23456	2345.6	21110.4
8	9	iiii	male	7654	765.4	6888.6
9	10	jjj	male	3456578	345657.8	3110920.2
10	11	kkkk	male	45678	4567.8	41110.2
11	12	llll	male	345	34.5	310.5
12	13	mmmm	female	9876	987.6	8888.4
13	14	nnnnn	male	34567	3456.7	31110.3
14	15	jjjj	male	87654	8765.4	78888.6

```
# print 5th record to 15th record
df[4:15] # normal slicint [0:n-1]
```

	id	name	gender	salary	tax	net salary
4	5	eeee	female	334567	33456.7	301110.3
5	6	ffff	female	21234	2123.4	19110.6
6	7	gggg	female	2345	234.5	2110.5

```
# to filer only males
df['gender']=='male' # only returns boolean
```

```
0      True
1     False
2      True
3      True
4     False
5     False
6     False
7     False
8      True
9     False
10     True
11     True
12    False
13     True
14     True
15    False
16    False
17    False
18    False
19     True
Name: gender, dtype: bool
```

```
df[df['gender']=='male']
```

	id	name	gender	salary	tax	net salary
0	1	aaaa	male	10000	1000.0	9000.0
2	3	ccccc	male	120000	12000.0	108000.0
3	4	dddd	male	45000	4500.0	40500.0
8	9	iiii	male	7654	765.4	6888.6
10	11	kkkk	male	45678	4567.8	41110.2
11	12	llll	male	345	34.5	310.5
13	14	nnnnn	male	34567	3456.7	31110.3
14	15	jjjj	male	87654	8765.4	78888.6
19	20	pppp	male	3455	345.5	3109.5

```
# from current file create two CSVs, where male.csv contains only males data, while female.csv contans only female data
dfmale=df[df['gender']=='male']
dfmale.to_csv("male.csv")
dffemale=df[df['gender']=='female']
dffemale.to_csv("female.csv")
```

```
# all the females earning more than 30,000
dffemalegt30k=df[(df['gender']=='female')&(df['salary']>=30000)] # logical op & is used here
dffemalegt30k.to_csv('femalegt30k.csv')
```

```
df
```

	id	name	gender	salary	tax	net salary
0	1	aaaa	male	10000	1000.0	9000.0
1	2	bbbb	female	8000	800.0	7200.0
2	3	ccccc	male	120000	12000.0	108000.0
3	4	dddd	male	45000	4500.0	40500.0
4	5	eeee	female	334567	33456.7	301110.3
5	6	ffff	female	21234	2123.4	19110.6
6	7	gggg	female	2345	234.5	2110.5
7	8	hhhh	female	23456	2345.6	21110.4
8	9	iiii	male	7654	765.4	6888.6
9	10	jjj	male	3456578	345657.8	3110920.2
10	11	kkkk	male	45678	4567.8	41110.2
11	12	llll	male	345	34.5	310.5
12	13	mmmm	female	9876	987.6	8888.4
13	14	nnnnn	male	34567	3456.7	31110.3
14	15	jjjj	male	87654	8765.4	78888.6
15	16	iiii	female	34567	3456.7	31110.3
16	17	mmmm	female	1234	123.4	1110.6
17	18	nnnnn	female	23456	2345.6	21110.4
18	19	oooo	female	5555	555.5	4999.5
19	20	pppp	male	3455	345.5	3109.5

```
# find max of ssalary
max(df.salary)
```

```
3456578
```

```
leng=len(df)
summ=sum(df.salary)
avgsal=summ/leng
print("average salary:", avgsal)
print(df[df['salary']>avgsal])
```

```
average salary: 213761.05
   id  name  gender   salary       tax  net salary
4   5  eeee  female   334567   33456.7    301110.3
9  10   jjj    male  3456578  345657.8   3110920.2
```

```
# sort by a particular field
df.sort_values(by='salary')
```

	id	name	gender	salary	tax	net salary
11	12	llll	male	345	34.5	310.5
16	17	mmmm	female	1234	123.4	1110.6
6	7	gggg	female	2345	234.5	2110.5
19	20	pppp	male	3455	345.5	3109.5
18	19	oooo	female	5555	555.5	4999.5
8	9	iiii	male	7654	765.4	6888.6
1	2	bbbb	female	8000	800.0	7200.0
12	13	mmmm	female	9876	987.6	8888.4
0	1	aaaa	male	10000	1000.0	9000.0
5	6	ffff	female	21234	2123.4	19110.6
17	18	nnnnn	female	23456	2345.6	21110.4
7	8	hhhh	female	23456	2345.6	21110.4
13	14	nnnnn	male	34567	3456.7	31110.3
15	16	iiii	female	34567	3456.7	31110.3
3	4	dddd	male	45000	4500.0	40500.0
10	11	kkkk	male	45678	4567.8	41110.2
14	15	jjjj	male	87654	8765.4	78888.6
2	3	ccccc	male	120000	12000.0	108000.0
4	5	eeee	female	334567	33456.7	301110.3
9	10	jjj	male	3456578	345657.8	3110920.2

```
df.sort_values(by='gender')
```

	id	name	gender	salary	tax	net salary
1	2	bbbb	female	8000	800.0	7200.0
17	18	nnnnn	female	23456	2345.6	21110.4
16	17	mmmm	female	1234	123.4	1110.6
4	5	eeee	female	334567	33456.7	301110.3
5	6	ffff	female	21234	2123.4	19110.6
6	7	gggg	female	2345	234.5	2110.5
7	8	hhhh	female	23456	2345.6	21110.4
15	16	iiii	female	34567	3456.7	31110.3
18	19	oooo	female	5555	555.5	4999.5
12	13	mmmm	female	9876	987.6	8888.4
0	1	aaaa	male	10000	1000.0	9000.0
14	15	jjjj	male	87654	8765.4	78888.6
13	14	nnnnn	male	34567	3456.7	31110.3
19	20	pppp	male	3455	345.5	3109.5
10	11	kkkk	male	45678	4567.8	41110.2

```
df.sort_values(by='name')
```

	id	name	gender	salary	tax	net salary
0	1	aaaa	male	10000	1000.0	9000.0
1	2	bbbb	female	8000	800.0	7200.0
2	3	ccccc	male	120000	12000.0	108000.0
3	4	dddd	male	45000	4500.0	40500.0

```
# complete details of person earning second highest salary
dfs=df.sort_values(by='salary')[-2:-1]
dfs
```

	id	name	gender	salary	tax	net salary
4	5	eeee	female	334567	33456.7	301110.3

```
df
```

	id	name	gender	salary	tax	net salary
0	1	aaaa	male	10000	1000.0	9000.0
1	2	bbbb	female	8000	800.0	7200.0
2	3	ccccc	male	120000	12000.0	108000.0
3	4	dddd	male	45000	4500.0	40500.0
4	5	eeee	female	334567	33456.7	301110.3
5	6	ffff	female	21234	2123.4	19110.6
6	7	gggg	female	2345	234.5	2110.5
7	8	hhhh	female	23456	2345.6	21110.4
8	9	iiii	male	7654	765.4	6888.6
9	10	jjj	male	3456578	345657.8	3110920.2
10	11	kkkk	male	45678	4567.8	41110.2
11	12	llll	male	345	34.5	310.5
12	13	mmmm	female	9876	987.6	8888.4
13	14	nnnnn	male	34567	3456.7	31110.3
14	15	jjjj	male	87654	8765.4	78888.6
15	16	iiii	female	34567	3456.7	31110.3
16	17	mmmm	female	1234	123.4	1110.6
17	18	nnnnn	female	23456	2345.6	21110.4
18	19	oooo	female	5555	555.5	4999.5
19	20	pppp	male	3455	345.5	3109.5

```
df.drop(labels=['tax', 'salary'], axis=1) #  axis:0--> row; axis:1--> column, default axis:0
```

	id	name	gender	net salary
0	1	aaaa	male	9000.0
1	2	bbbb	female	7200.0
2	3	ccccc	male	108000.0
3	4	dddd	male	40500.0
4	5	eeee	female	301110.3
5	6	ffff	female	19110.6
6	7	gggg	female	2110.5
7	8	hhhh	female	21110.4
8	9	iiii	male	6888.6
9	10	jjj	male	3110920.2
10	11	kkkk	male	41110.2
11	12	llll	male	310.5
12	13	mmmm	female	8888.4
13	14	nnnnn	male	31110.3
14	15	jjjj	male	78888.6
15	16	iiii	female	31110.3
16	17	mmmm	female	1110.6
17	18	nnnnn	female	21110.4
18	19	oooo	female	4999.5
19	20	pppp	male	3109.5

```
df.drop(0) # deletes 0th row
```

	id	name	gender	salary	tax	net salary
1	2	bbbb	female	8000	800.0	7200.0
2	3	ccccc	male	120000	12000.0	108000.0
3	4	dddd	male	45000	4500.0	40500.0
4	5	eeee	female	334567	33456.7	301110.3
5	6	ffff	female	21234	2123.4	19110.6
6	7	gggg	female	2345	234.5	2110.5
7	8	hhhh	female	23456	2345.6	21110.4
8	9	iiii	male	7654	765.4	6888.6
9	10	jjj	male	3456578	345657.8	3110920.2
10	11	kkkk	male	45678	4567.8	41110.2

```
df.drop([1, 5, 10])
```

	id	name	gender	salary	tax	net salary
0	1	aaaa	male	10000	1000.0	9000.0
2	3	ccccc	male	120000	12000.0	108000.0
3	4	dddd	male	45000	4500.0	40500.0
4	5	eeee	female	334567	33456.7	301110.3
6	7	gggg	female	2345	234.5	2110.5
7	8	hhhh	female	23456	2345.6	21110.4
8	9	iiii	male	7654	765.4	6888.6
9	10	jjj	male	3456578	345657.8	3110920.2
11	12	llll	male	345	34.5	310.5
12	13	mmmm	female	9876	987.6	8888.4
13	14	nnnnn	male	34567	3456.7	31110.3
14	15	jjjj	male	87654	8765.4	78888.6
15	16	iiii	female	34567	3456.7	31110.3
16	17	mmmm	female	1234	123.4	1110.6
17	18	nnnnn	female	23456	2345.6	21110.4
18	19	oooo	female	5555	555.5	4999.5
19	20	pppp	male	3455	345.5	3109.5

```
df.drop('female') # by default axis:0, which means it'll search rows, but 'female' is column/field i.e. axis:1
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-114-c17764ef9438> in <cell line: 1>()
----> 1 df.drop('female')

                                 4 frames
/usr/local/lib/python3.9/dist-packages/pandas/core/indexes/base.py in drop(self, labels, errors)
   6659         if mask.any():
   6660             if errors != "ignore":
-> 6661                 raise KeyError(f"{list(labels[mask])} not found in axis")
   6662             indexer = indexer[~mask]
   6663         return self.delete(indexer)

KeyError: "['female'] not found in axis"
```

```
df.drop('gender', axis=1)
```

	id	name	salary	tax	net salary
0	1	aaaa	10000	1000.0	9000.0
1	2	bbbb	8000	800.0	7200.0
2	3	ccccc	120000	12000.0	108000.0
3	4	dddd	45000	4500.0	40500.0
4	5	eeee	334567	33456.7	301110.3
5	6	ffff	21234	2123.4	19110.6
6	7	gggg	2345	234.5	2110.5
7	8	hhhh	23456	2345.6	21110.4
8	9	iiii	7654	765.4	6888.6
9	10	jjj	3456578	345657.8	3110920.2
10	11	kkkk	45678	4567.8	41110.2
11	12	llll	345	34.5	310.5
12	13	mmmm	9876	987.6	8888.4
13	14	nnnnn	34567	3456.7	31110.3
14	15	jjjj	87654	8765.4	78888.6
15	16	iiii	34567	3456.7	31110.3
16	17	mmmm	1234	123.4	1110.6
17	18	nnnnn	23456	2345.6	21110.4
18	19	oooo	5555	555.5	4999.5
19	20	pppp	3455	345.5	3109.5

```
import io
df=pd.read_csv(io.BytesIO(uploaded['mycsv.csv'])) # creates data frame; data frame is immutable, it needs to be saved
print(df)
```

	id	name	gender	salary
0	1	aaaa	male	10000
1	2	bbbb	female	8000
2	3	ccccc	male	120000
3	4	dddd	male	45000
4	5	eeee	female	334567
5	6	ffff	female	21234
6	7	gggg	female	2345
7	8	hhhh	female	23456
8	9	iiii	male	7654
9	10	jjj	male	3456578
10	11	kkkk	male	45678
11	12	llll	male	345
12	13	mmmm	female	9876
13	14	nnnnn	male	34567
14	15	jjjj	male	87654
15	16	iiii	female	34567
16	17	mmmm	female	1234
17	18	nnnnn	female	23456
18	19	oooo	female	5555
19	20	pppp	male	3455

```
1 df
```

	id	name	gender	salary

```
df.drop(df[df['gender']=='female'].index)
```

	id	name	gender	salary
0	1	aaaa	male	10000
2	3	ccccc	male	120000
3	4	dddd	male	45000
8	9	iiii	male	7654
9	10	jjj	male	3456578
10	11	kkkk	male	45678
11	12	llll	male	345
13	14	nnnnn	male	34567
14	15	jjjj	male	87654
19	20	pppp	male	3455

```
df.describe()
```

	id	salary
count	20.00000	2.000000e+01
mean	10.50000	2.137610e+05
std	5.91608	7.669659e+05
min	1.00000	3.450000e+02
25%	5.75000	7.129250e+03
50%	10.50000	2.234500e+04
75%	15.25000	4.516950e+04
max	20.00000	3.456578e+06

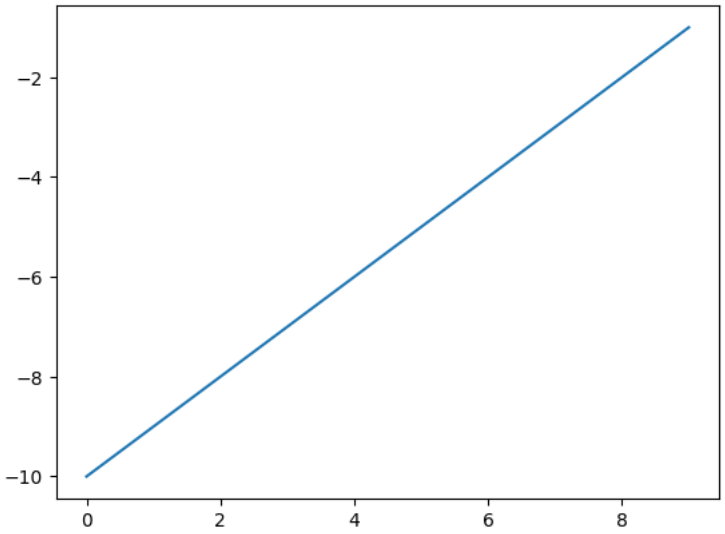
```
df.isnull()
```

	id	name	gender	salary
0	False	False	False	False
1	False	False	False	False
2	False	False	False	False
3	False	False	False	False
4	False	False	False	False
5	False	False	False	False
6	False	False	False	False
7	False	False	False	False
8	False	False	False	False
9	False	False	False	False
10	False	False	False	False
11	False	False	False	False
12	False	False	False	False
13	False	False	False	False

```
1 import matplotlib.pyplot as plt
```

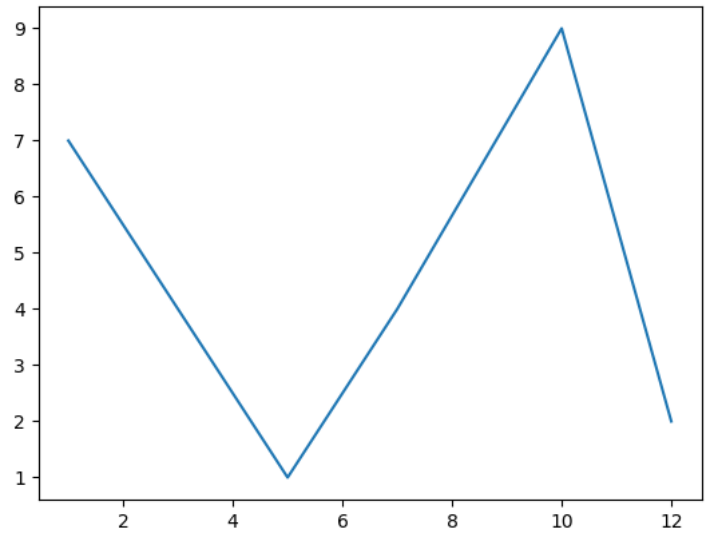
```
1 x=list(range(0, 10))
2 y=list(range(-10, 0))
3 plt.plot(x, y)
```

```
[<matplotlib.lines.Line2D at 0x7f15c4fe6a00>]
```



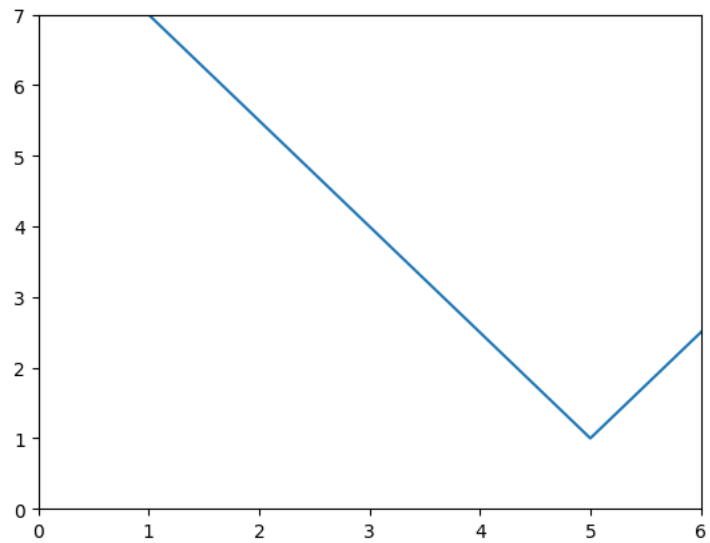
```
x=list([1, 5, 7, 10, 12])
y=list([7, 1, 4, 9, 2])
plt.plot(x, y)
```

```
[<matplotlib.lines.Line2D at 0x7f15c40b4bb0>]
```



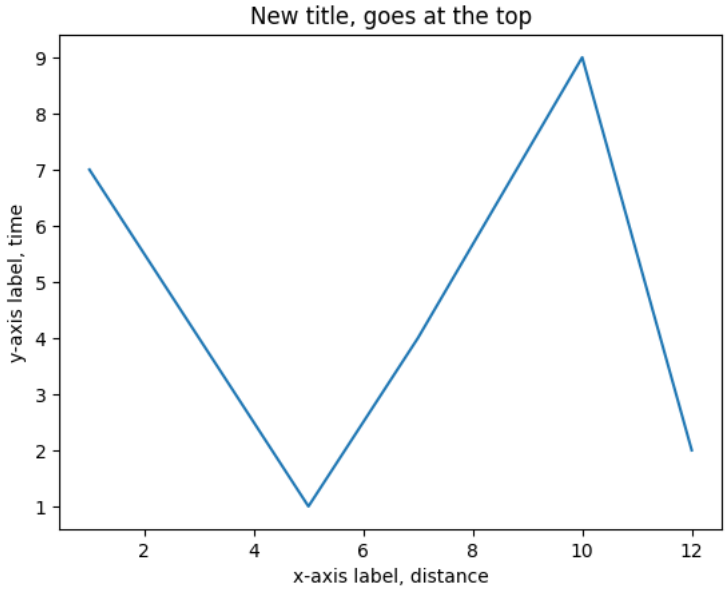

```
plt.axis([0, 6, 0, 7]) # to start display plotting between x[0 to 6] y[0 to 7]
plt.plot(x, y)
```

```
[<matplotlib.lines.Line2D at 0x7f15c40e7eb0>]
```



```
plt.title("New title, goes at the top") # adds title to graph at top
plt.xlabel("x-axis label, distance") # adds label to x-axis
plt.ylabel("y-axis label, time") # adds label to y-axis)
plt.plot(x, y)
```

```
[<matplotlib.lines.Line2D at 0x7f15c3e9f250>]
```

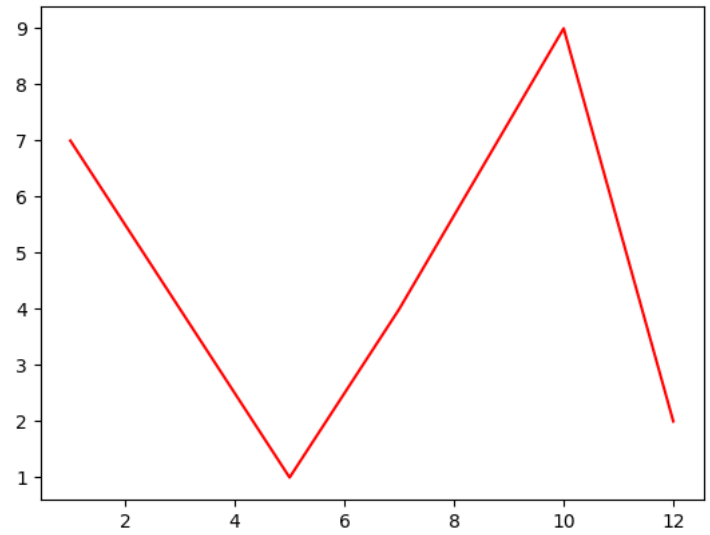



```
plt.xticks((2, 4, 6, 8, 10, 12),("two", "four", "six", "eight", "ten", "twelve")) # renames tics/points seen on x-axis
plt.yticks((1, 2, 3, 4, 5, 6, 7, 8, 9),("one", "two", "three", "four", "five", "six", "seven", "eight", "nine")) # renames tics/points seen on y-axis
plt.plot(x, y)
```

```
[<matplotlib.lines.Line2D at 0x7f15c3c19430>]
```

```
plt.plot(x, y, color="red") #adds color to graph
```

```
[<matplotlib.lines.Line2D at 0x7f15c3d15430>]
```




```
# continue visualization here
```

numpy

```
import numpy as np
```

```
m=np.array([[10, 20, 30, 40], [12, 23, 34, 45]])
print(m)
```

```
[[10 20 30 40]
 [12 23 34 45]]
```

```
print(m.shape) # prints rows, cols only if square or rectangular manner, else only number of rows
```

```
(2, 4)
```

```
m=np.array([[10, 20, 30], [12, 23, 34, 45]])
print(m)
```

```
[list([10, 20, 30]) list([12, 23, 34, 45])]
<ipython-input-86-0caeeedaa312>:1: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or s
  m=np.array([[10, 20, 30], [12, 23, 34, 45]])
```

```
print(m.shape) # only if square or rectangular manner, else only number of rows
print(m.size) # prints number of elements in numpy matrix/array only if square/rectangle manner otherwise only no of rows
```

```
(2,)
2
```

```
m=np.array([10, 20, 30, 12, 23, 34, 45])
print(m)
```

```
[10 20 30 12 23 34 45]
```

```
print(m.shape) # prints only number of cols in case of 1D matrix
```

```
(7,)
```

```
m=np.array([[10, 20, 30, 40], [12, 23, 34, 45]])
print(m)
```

```
[[10 20 30 40]
 [12 23 34 45]]
```

```
print(m.shape)
print(m.ndim) # prints no of
```

```
(2, 4)
2
```

```
m=np.array([[5, 10, 20, 30, 40], [12, 23, 34, 45]])
print(m)
```

```
[list([5, 10, 20, 30, 40]) list([12, 23, 34, 45])]
<ipython-input-90-73c3c866503a>:1: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or s
  m=np.array([[5, 10, 20, 30, 40], [12, 23, 34, 45]])
```

```
print(m.shape)
print(m.ndim)
```

```
(2,)
1
```

```
m=np.array([[10, 20, 30, 40], [12, 23, 34, 45]])
print(m)
```

```
[[10 20 30 40]
 [12 23 34 45]]
```

```
print(m.size) # prints number of elements in numpy matrix/array if matrix is square/rectangle otherwise prints no of rows
```

```
8
```

```
m=np.array([[20, 30, 40], [12, 23, 34, 45]])
print(m)
```

```
[list([20, 30, 40]) list([12, 23, 34, 45])]
<ipython-input-95-e069a42df23d>:1: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or s
  m=np.array([[20, 30, 40], [12, 23, 34, 45]])
```

```
m.size
```

```
2
```

```
m=np.array([[10, 20, 30, 40], [12, 23, 34, 45]])
print(m)
```

```
[[10 20 30 40]
 [12 23 34 45]]
```

```
m[0][2] # allows indexing
```

```
30
```

```
m=np.array([[11,22,33,44],[55,66,77,88],[99,111,222,333]])
print(m)
```

```
[[ 11  22  33  44]
 [ 55  66  77  88]
 [ 99 111 222 333]]
```

```
m.size
```

```
12
```

```
m.ndim
```

```
2
```

```
m.shape
```

```
(3, 4)
```

```
m[0][2]
```

```
33
```

```
m=np.array([[11,22,33,44],[55,66,77,88],[99,111,222,333]])
print(m)
```

```
[[ 11  22  33  44]
 [ 55  66  77  88]
 [ 99 111 222 333]]
```

```
#slice [start:end:step]
m[0:2, 0:2]
```

```
array([[11, 22],
       [55, 66]])
```

```
m[1:3, 2:4]
```

```
array([[ 77,  88],
       [222, 333]])
```

```
for i in m:
    print(i)
```

```
[11 22 33 44]
[55 66 77 88]
[ 99 111 222 333]
```

```
len(m) # returns number of rows
```

```
3
```

```
len(m[0]) # returns no of columns
```

```
4
```

```
for row in range(len(m)):
    for col in range(len(m[row])):
        print(m[row][col])
```

```
11
22
33
44
55
66
77
88
99
111
222
333
```

```
for sublist in m:
    for element in sublist:
        print(element)
```

```
11
22
33
44
55
66
77
88
```

```
99
111
222
333
```

```
for i in np.nditer(m[::]): # nditer allows you to slice a matrix
    print(i)
```

```
11
22
33
44
55
66
77
88
99
111
222
333
```

```
m=np.arange(1, 100, 10) # array-range, arange(start,end, step)
print(m)
```

```
[ 1 11 21 31 41 51 61 71 81 91]
```

```
m=np.arange(1, 13)
print(m)
```

```
[ 1  2  3  4  5  6  7  8  9 10 11 12]
```

```
m.reshape(3, 4)
```

```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

```
m=np.arange(1, 24,2).reshape(3, 4)
print(m)
```

```
[[ 1  3  5  7]
 [ 9 11 13 15]
 [17 19 21 23]]
```

```
m=np.linspace(1, 10, 2) # to take samples from line-space
print(m)
```

```
[ 1. 10.]
```

```
m=np.linspace(1, 10, 4)
print(m)
```

```
[ 1.  4.  7. 10.]
```

```
m=np.linspace(1, 10, 10)
print(m)
```

```
[ 1.  2.  3.  4.  5.  6.  7.  8.  9. 10.]
```

```
m=np.linspace(1, 12, 4)
print(m)
```

```
[ 1.          4.66666667  8.33333333 12.        ]
```

Random

```
from numpy import random
```

```
print(random.randint(1000))
```

```
148
```

```
print(random.randint(6))
```

```
4
```

```
print(random.randint(6, size=6))
```

```
[2 4 0 5 2 0]
```

```
print(random.randint(100, size=6))
```

```
[ 0 40  3 36 40 89]
```

```
print(random.randint(100, size=(6,6)))
```

```
[[83 46 17 20 44 81]
 [56 15 33 27 91 39]
 [47 71  3  5  6 87]
 [63 84 30 75 40  8]
 [36 73 93  7 41 54]
 [82 38 19  2 59 31]]
```

```
print(random.randint(6, size=(6,6)))
```

```
[[4 3 2 0 5 0]
 [5 5 0 0 1 0]
 [2 2 5 3 5 0]
 [3 4 3 4 2 3]
 [5 1 1 4 4 1]
 [5 0 0 0 2 1]]
```

```
random.choice(["amar", "kiran", "shraddha", "mahesh"])
```

```
'kiran'
```

```
random.choice(["amar", "kiran", "shraddha", "mahesh"], size=(5,6))
```

```
array([['shraddha', 'amar', 'shraddha', 'kiran', 'shraddha', 'amar'],
       ['shraddha', 'shraddha', 'mahesh', 'mahesh', 'mahesh', 'kiran'],
       ['kiran', 'kiran', 'kiran', 'mahesh', 'kiran', 'mahesh'],
       ['amar', 'shraddha', 'mahesh', 'mahesh', 'amar', 'shraddha'],
       ['kiran', 'shraddha', 'amar', 'mahesh', 'amar', 'amar']],
      dtype='<U8')
```

```
random.rand(4, 5)
```

```
array([[0.90709204, 0.62176315, 0.85781287, 0.02289043, 0.05196228],
       [0.50285407, 0.33343447, 0.29778532, 0.30078703, 0.57918208],
       [0.94190511, 0.11215815, 0.50825315, 0.89404204, 0.28902269],
       [0.62693843, 0.53613931, 0.8830277 , 0.95597631, 0.30922973]])
```

```
m1=np.arange(1, 11).reshape(5, 2)
m2=np.arange(101, 111).reshape(5, 2)
print(m1, "\n")
print(m2, "\n")
m3=m1+m2
print(m3)
```

```
[[ 1  2]
 [ 3  4]
 [ 5  6]
 [ 7  8]
 [ 9 10]]

[[101 102]
 [103 104]
 [105 106]
 [107 108]
 [109 110]]

[[102 104]
 [106 108]
 [110 112]
 [114 116]
 [118 120]]
```

```
m1=random.randint(1, 50, size=(5, 5))
np.sort(m1)
print(m1)
```

```
[[34  5 15 16 43]
 [ 2 36 39 23 27]
 [45 48 17 24 41]
 [48  7 11 40  6]
 [21  9 45 18 15]]
```

```
m=random.randint(1, 50, size=(5, 5))
np.sort(m, axis=0) # sort by row
print(m)
```

```
[[40 31 49 30 21]
 [22 46  1  6 40]
 [21 15 36 18 28]
 [34 43 11 12  5]
 [26  1 34 46  1]]
```

```
v=np.where(m>=30)
print(v) # list of indexes that satisfies condition
```

```
(array([0, 0, 0, 0, 1, 1, 2, 3, 3, 4, 4]), array([0, 1, 2, 3, 1, 4, 2, 0, 1, 2, 3]))
```

```
print(v[0], v[1])
print(v[0][0], v[1][0])
print(v[0][4], v[1][4])
print(v[0][5], v[1][5])
```

```
[0 0 0 0 1 1 2 3 3 4 4] [0 1 2 3 1 4 2 0 1 2 3]
0 0
1 1
1 4
```

```
for idx in range(len(v[0])):
    #print(v[0][idx], v[1][idx])
    print(m[v[0][idx]][v[1][idx]])
```

```
40
31
49
30
46
40
36
34
43
34
46
```

```
print(m[v])
```

```
[42 46 35 37 34 34 41 39 31 36 35]
```

```

```