In [ ]:
```
import random   #playing a game for computer to randomly select a string from a l
name = ["Ridwan", "Aisha", "Ameenah", "Ameer", "mariam"]
print(random.choice(name))
```

In [ ]:
```
counter = 1    # calculating the product of first few numbers
result = 1

while counter <= 10:
    result = result * counter
    counter += 1
print (result)
```

In [ ]:
```
counter = 1    # calculating the sum of first few numbers
result = 0

while counter <= 5:
    result = result + counter
    counter += 1
print (result)
```

In [ ]:
```
line = "*"
max_length = 5
while len(line) <= max_length:
    print(line)
    line += "*"

while len(line) > 0:
    print(line)
    line = line[:-1]
```

In [ ]:
```
counter = 1 #calculating the product of all even numbers between 1 and 20
result = 1

while counter < 20:
    counter += 1
    if counter % 2 == 0:
        continue
    result = result * counter

print(result)
```

In [ ]:
```
family = {
    "father": "Ridwan",
    "mother": "mariam",
    "daughter1": "Ameenah",
    "daughter2": "Aisha",
    "son": "Ameer"
}

for key in family:
    print("key:", key, ', ' "value:", family[key])
```

when using a dictionary

- for key in family will give you they keys alone
- for key in family.values will give you the values alone
- for key in family.items will give you the key and the values

In [ ]:
```
def filter_odd(numbers):  #filtering of odd number function
    result = []
    for num in numbers:
        if num % 2 == 1:
            result.append(num)
    return result


odd = filter_odd([1, 3, 5, 7, 9, 11, 13, 15])
print(odd)
```

In [ ]:
```
import math

def equal_monthly(principal, rate, number, down_payment=0):
    principals = principal - payment
    emi = principals * rate * (1 + rate) ** number / ((1 + rate) ** number - 1)
    emi = math.ceil(emi)
    return emi


emi1 = equal_monthly(800000, 0.07 / 12, 6 * 12, payment=200000)
emi2 = equal_monthly(60000, 0.12 / 12, 1 * 12)
print(f" total loan repayment is {emi1 + emi2}")
```

In [ ]:
```
cities = ["paris", "london", "dubai", "mumbai"]
values = [[200, 20, 200], [250, 30, 120], [370, 15, 80], [450, 10, 70]]
x = zip(cities, values)
yes = list(x)
print(yes)
```

In [1]:
```
dico = {
    "one": 1,
    "two": 2,
    "three": 3
}
now = dico.items()
print(now)
```

```
dict_items([('one', 1), ('two', 2), ('three', 3)])
```

In [ ]:
```
import numpy as np

a = np.array([2, 5, 9])
b = np.array([9, 34, 9])
c = a + b
print(np.sin(c))
```

In [ ]:
```
import jovian
```

In [ ]:
```
cities = ['paris', 'London', 'Dubai', 'Mumbai']  #algorithm to determine which ci
amounts = [[200, 20, 200], [250, 30, 120], [370, 15, 80], [450, 10, 70]]


def vacation_plan(cities, amounts):
    dico = {}
    for city, amount in zip(cities, amounts):
        dico[city] = amount
    return dico


vacate = vacation_plan(cities, amounts)


def least_expensive_city(vacate, duration):
    result = {}
    for key, value in vacate.items():
        result[key] = sum(value) * duration
    min_city = min(result, key=result.get)
    return min_city


the_city = least_expensive_city(vacate, 7)

print(f"the least expensive city is {the_city}")
```

In [ ]:
```
import jovian dict_items([('one', 1), ('two', 2), ('three', 3)])
```

In [ ]:
```
jovian.commit()
```

In [ ]:
```
pip install jovian --upgrade
```

In [ ]:
```
import jovian
```

In [ ]:
```
jovian.commit()
```

In [ ]:
```
import numpy as np

a = np.array([2, 5, 9])
b = np.array([9, 34, 9])
c = a + b
print(np.sin(c))
```

In [ ]:
```
import numpy as np
import urllib.request

urllib.request.urlretrieve("C:\Users\ENR RILWAN\Desktop\climatic.txt",
                           "clime.txt")

climate_data = np.genfromtxt("clime.txt", delimeter=",", skip_header=1)
print(climate_data)
```

In [ ]:
```
file = open("climate.txt", "r")

f = file.readlines()
print(f)
```

In [ ]:
```
import numpy as np   #how to read a file from a path


climate_data = np.genfromtxt(r"C:\Users\ENR RILWAN\PycharmProjects\pythonProject1
                             delimiter=",", skip_header=1)

print(climate_data.shape)
```

In [ ]:
```
import numpy as np   #calculating the yields of apples using matrix formular


climate_data = np.genfromtxt(r"C:\Users\ENR RILWAN\Desktop\climate.txt", #to have
                             delimiter=",", skip_header=1)

weights = np.array([0.3, 0.2, 0.1])

yields = climate_data @ weights

climate_results = np.concatenate((climate_data, yields.reshape(21, 1)), axis=1) #

print(climate_results)
np.savetxt("climate_results.txt", climate_results, fmt="%.2f",
           header="temperature,hummidity,rainfall,yield", comments="")
```

In [ ]:
```
file = open("family.txt", "r")
f = file.readlines()

result = []
for i in f:
    if i[:-1] == "\n":
        result.append(i[:-1])
    else:
        result.append(i)

print(result)
```

In [2]:
```
import numpy as np

a = np.array(42)
b = np.array([1, 2, 3, 4, 5])
c = np.array([[1, 2, 3], [4, 5, 6]])
d = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])

print(a.ndim)
print(b.ndim)
print(c.ndim)
print(d.ndim)
```

```
0
1
2
3
```

In [ ]:
```
import numpy as np

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

print(arr)
print('number of dimensions :', arr.ndim)
```

In [7]:
```
import os   #making a new directory

p = os.listdir("./glo")
print(p)
```

```
['covid.txt', 'loan.txt']
```

In [ ]:
```
import numpy as np    #CONVERTING NP FLOAT TO INTEGER. you can either use optionc

arr = np.array([1.1, 2.1, 3.1])

newarr = arr.astype('i')

print(newarr)
print(newarr.dtype)
```

In [ ]:
```
import numpy as np    #copy is not affected by the changes made in array, view is

arr = np.array([1, 2, 3, 4, 5, 6])
arr1 = arr.copy()
arr2 = arr.view()
arr[3] = 8

print(arr)
print(arr1)
print(arr2)
```

In [ ]:
```
import numpy as np   # dot base will return the copy of the array, all these will

arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])

arr3 = arr.view()

print(arr.reshape(2, 4).base)
arr2 = arr.copy()
print(arr2)
print(arr3)
```

In [ ]:
```
import numpy as np  #these 2 functions will return the same outpu

arr = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])

for x in np.nditer(arr):
    print(x)

for x in arr:
    for y in x:
        for z in y:
            print(z)
```

In [ ]:
```
import numpy as np  #iterating arrays with different data types

arr = np.array([1, 2, 3])

for x in np.nditer(arr, flags=['buffered'], op_dtypes=['S']):
  print(x)
```

In [ ]:
```
import numpy as np   #this will show the sequence while carrying out the iteratic

arr = np.array([1, 2, 3])

for idx, x in np.ndenumerate(arr):
  print(idx, x)
```

In [ ]:
```
import numpy as np   #joining arrays together

arr1 = np.array([3, 5, 8])
arr2 = np.array([4, 7, 9])

arr3 = np.concatenate((arr1, arr2))
print(arr3)
```

In [6]:
```
import os   #making a new directory

p = os.listdir(".")
print(p)
```

```
['.anaconda', '.cache', '.conda', '.condarc', '.idlerc', '.ipynb_checkpoints',
'.ipython', '.jovianrc', '.jupyter', '.vscode', '3D Objects', 'app.ipynb', 'App
Data', 'Application Data', 'Contacts', 'Cookies', 'Desktop', 'Documents', 'Down
loads', 'Favorites', 'glo', 'IntelGraphicsProfiles', 'JN', 'Links', 'Local Sett
ings', 'Music', 'My Documents', 'NetHood', 'New folder', 'ntuser.dat', 'ntuser.
dat.LOG1', 'ntuser.dat.LOG2', 'NTUSER.DAT{016888bd-6c6f-11de-8d1d-001e0bcde3e
c}.TM.blf', 'NTUSER.DAT{016888bd-6c6f-11de-8d1d-001e0bcde3ec}.TMContainer000000
00000000000001.regtrans-ms', 'NTUSER.DAT{016888bd-6c6f-11de-8d1d-001e0bcde3ec}.
TMContainer00000000000000000002.regtrans-ms', 'ntuser.dat{464f8f48-c5d4-11ec-be
6a-c4d9878281fa}.TM.blf', 'ntuser.dat{464f8f48-c5d4-11ec-be6a-c4d9878281fa}.TMC
ontainer00000000000000000001.regtrans-ms', 'ntuser.dat{464f8f48-c5d4-11ec-be6a-
c4d9878281fa}.TMContainer00000000000000000002.regtrans-ms', 'ntuser.ini', 'OneD
rive', 'Pictures', 'PrintHood', 'PycharmProjects', 'Recent', 'Ridwan', 'Saved G
ames', 'Searches', 'SendTo', 'Start Menu', 'Templates', 'Untitled Folder 2', 'U
ntitled Folder 3', 'Untitled Folder 4', 'Untitled Folder 5', 'Untitled.ipynb',
'Untitled1.ipynb', 'Untitled2.ipynb', 'Untitled3.ipynb', 'Untitled4.ipynb', 'Vi
deos']
```

In [ ]:
```
import os
os.makedirs("./glo", exist_ok=True)
```

In [ ]:
```
words = ["dog", "cat", "rat", "pig", "cow"]  #how to create a dictionary
meanings = ["aja", "musu", "eku", "elede", "malu"]

def animal(key, values):
    result = {}
    for word, meaning in zip(words, meanings):
        result[word] = meaning
    return result

this = animal(words, meanings)
print(this)
```

In [ ]:
```
a = [1, 'x', 'y']
b = [1, 2]
a.extend(b)

print(a)
```

In [2]:
```
import os

import math
with open("./glo/loan.txt", "r") as file1:
    file2 = file1.readlines()
```

In [ ]:
```
read_csv("./glo/loan.txt")
```

In [9]:
```
def parse_headers(header_line):
    return header_line.strip().split(",")
headers = parse_headers(file2[0]
print(headers)
```

```
  File "C:\Users\ENR RILWAN\AppData\Local\Temp\ipykernel_11796\523103517.py", l
ine 4
    print(headers)
        ^
SyntaxError: invalid syntax
```

In [ ]:
```
import os

import math
with open("./glo/loan.txt", "r") as file1:
    file2 = file1.readlines()

def parse_headers(header_line):
    return header_line.strip().split(",")
headers = parse_headers(file2[0])

def parse_values(data_line):
    values = []
    for item in data_line.strip().split(","):
        if item == "":
            values.append(0.0)
        else:
            values.append(float(item))
    return values
values = parse_values(file2[2])
#print(values)

def create_dictionary(values, headers):
    result = {}
    for value, header in zip(values, headers):
        result[header] = value
    return result

def read_csv(path):
    result = []
    with open(path, "r") as f:
        file2 = f.readlines()
        headers = parse_headers(file2[0])
        for data_line in file2[1:]:
            values = parse_values(data_line)
            dictionari = create_dictionary(values, headers)
            result.append(dictionari)

    return result

with open("./glo/loan.txt") as file5:
    file4 = file5.read()
new_loan = (read_csv("./glo/loan.txt"))

def equal_monthly(principal, rate, number, payment=0):
    principals = principal - payment
    emi = principals * rate * (1 + rate) ** number / ((1 + rate) ** number - 1)
    emi = math.ceil(emi)
    return emi

emi1 = equal_monthly(800000, 0.07 / 12, 6 * 12, payment=200000)
emi2 = equal_monthly(60000, 0.12 / 12, 1 * 12)
print(f" total loan repayment is {emi1 + emi2}")
def comp_emi(loan):
    for loans in new_loan:
        loans["emi"] = equal_monthly(loans["amount"],
                                     loans["duration"],
```

```
                                    loans["rate"]/12,
                                    loans["down_payment"])
print(new_loan)
```

In [ ]:
```
new_loan
```

In [ ]:
```
import os

import math
with open("./glo/loan.txt", "r") as file1:
    file2 = file1.readlines()
print(file2)
```

In [ ]:
```
file2
```

In [ ]:
```
file2[0].strip().split(",")
```

In [11]:
```
import os

import pandas as pd

covid_df = pd.read_csv("./glo/covid.txt")

print(type(covid_df))
```

```
<class 'pandas.core.frame.DataFrame'>
```

In [12]:
```
print(type(covid_df))
```

```
<class 'pandas.core.frame.DataFrame'>
```

In [8]:
```
import os

os.listdir("./glo")
```

Out[8]:
```
['covid.txt', 'loan.txt']
```

In [10]:
```
pip install pandas
```

```
Collecting pandasNote: you may need to restart the kernel to use updated packag
es.

  Downloading pandas-1.3.5-cp37-cp37m-win_amd64.whl (10.0 MB)
Collecting pytz>=2017.3
  Using cached pytz-2022.1-py2.py3-none-any.whl (503 kB)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\new\envs\snakes\lib
\site-packages (from pandas) (2.8.2)
Requirement already satisfied: numpy>=1.17.3 in c:\new\envs\snakes\lib\site-pac
kages (from pandas) (1.21.6)
Requirement already satisfied: six>=1.5 in c:\new\envs\snakes\lib\site-packages
(from python-dateutil>=2.7.3->pandas) (1.16.0)
Installing collected packages: pytz, pandas
Successfully installed pandas-1.3.5 pytz-2022.1
```

In [19]:
```
covid_df
```

Out[19]:

	date	new_cases	new_deaths	new_tests
0	2020-04-21	2256.0	454.0	28045.0
1	2020-03-11	2236.0	451.0	28095.0
2	2020-05-21	2250.0	354.0	28195.0
3	2020-04-21	2256.0	404.0	28090.0
4	2020-05-23	1256.0	804.0	18095.0
5	2020-04-20	2258.0	451.0	21095.0

In [14]:
```
print(covid_df)
```

```
         date  new_cases  new_deaths  new_tests
0  2020-04-21     2256.0       454.0    28045.0
1  2020-03-11     2236.0       451.0    28095.0
2  2020-05-21     2250.0       354.0    28195.0
3  2020-04-21     2256.0       404.0    28090.0
4  2020-05-23     1256.0       804.0    18095.0
5  2020-04-20     2258.0       451.0    21095.0
```

In [18]:
```
covid_df.describe()
```

Out[18]:

	new_cases	new_deaths	new_tests
count	6.000000	6.000000	6.000000
mean	2085.333333	486.333333	25269.166667
std	406.368634	160.465157	4496.677014
min	1256.000000	354.000000	18095.000000
25%	2239.500000	415.750000	22832.500000
50%	2253.000000	451.000000	28067.500000
75%	2256.000000	453.250000	28093.750000
max	2258.000000	804.000000	28195.000000

In [21]:
```
covid_df["new_tests"][4]
```

Out[21]:
```
18095.0
```

In [27]:
```
case = covid_df[["new_cases", "new_deaths"]]
case
```

Out[27]:

	new_cases	new_deaths
0	2256.0	454.0
1	2236.0	451.0
2	2250.0	354.0
3	2256.0	404.0
4	1256.0	804.0
5	2258.0	451.0

In [26]:
```
case
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_8396\396761482.py in <module>
----> 1 case

NameError: name 'case' is not defined
```

In [28]:
```
case
```

Out[28]:

	new_cases	new_deaths
0	2256.0	454.0
1	2236.0	451.0
2	2250.0	354.0
3	2256.0	404.0
4	1256.0	804.0
5	2258.0	451.0

In [29]:
```
covid_df.loc[4]
```

Out[29]:
```
date          2020-05-23
new_cases         1256.0
new_deaths         804.0
new_tests        18095.0
Name: 4, dtype: object
```

In [33]:
```
covid_df.at["new_deaths"]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_8396\3585644893.py in <module>
----> 1 covid_df.at["new_deaths"]

C:\New\envs\snakes\lib\site-packages\pandas\core\indexing.py in __getitem__(sel
f, key)
   2273             return self.obj.loc[key]
   2274
-> 2275         return super().__getitem__(key)
   2276
   2277     def __setitem__(self, key, value):

C:\New\envs\snakes\lib\site-packages\pandas\core\indexing.py in __getitem__(sel
f, key)
   2220
   2221         key = self._convert_key(key)
-> 2222         return self.obj._get_value(*key, takeable=self._takeable)
   2223
   2224     def __setitem__(self, key, value):

TypeError: _get_value() missing 1 required positional argument: 'col'
```

In [34]:
```
covid_df.at[4, "new_tests"]
```

Out[34]:
```
18095.0
```

In [36]:
```
covid_df["new_cases"]
```

Out[36]:
```
0    2256.0
1    2236.0
2    2250.0
3    2256.0
4    1256.0
5    2258.0
Name: new_cases, dtype: float64
```

In [ ]:
```

```