1.An anonymous dataset containing each user's salary (in dollars) and tenure as a data scientist (in years) is given.

salaries and tenures = [(83000, 8.7), (88000, 8.1), (48000, 0.7), (76000, 6), (69000, 6.5), (76000, 7.5), (60000, 2.5), (83000, 10), (48000, 1.9), (63000, 4.2)]

Find out the average salary for each tenure and print a massage according to its value, i.e. "less than two", "between two and five" and "more than five" tenure and group together the salaries corresponding to each bucket. Compute the average salary for each group

Code :-

```
#Author : Dhrub khatry

#Regd.No. : 2141011043

from collections import defaultdict

salaries_and_tenures = [(83000, 8.7),(88000, 8.1),(48000, 0.7),(76000, 6),(69000, 6.5),(76000, 7.5),(60000, 2.5),(83000, 10),(48000, 1.9),(63000, 4.2)]

salary_per_tenure = defaultdict(list)

for salary, tenure in salaries_and_tenures:

    salary_per_tenure[tenure].append(salary)

average_salary_by_tenure = {

tenure:sum(salaries) / len(salaries)

for tenure, salaries in salary_per_tenure.items()

}

print(average_salary_by_tenure)

def tenure_bucket(tenure):

 if tenure < 2:

  return "less than two"

 elif tenure < 5:

  return "between two and five"

 else:

  return "more than five"

salary_by_tenure_bucket = defaultdict(list)

for salary, tenure in salaries_and_tenures:

    bucket = tenure_bucket(tenure)

    salary_by_tenure_bucket[bucket].append(salary)
```

average_salary_by_bucket = {

tenure_bucket: sum(salaries)/ len(salaries)

for tenure_bucket,salaries in salary_by_tenure_bucket.items()

}

output

```
IPython 7.13.0 -- An enhanced Interactive Python.

In [1]: runfile('/home/student/2141016146/a1.py', wdir='/home/student/2141016146')
{8.7: 83000.0, 8.1: 88000.0, 0.7: 48000.0, 6: 76000.0, 6.5: 69000.0, 7.5: 76000.0, 2.5: 60000.0, 10:
83000.0, 1.9: 48000.0, 4.2: 63000.0}
{'more than five': 79166.66666666667, 'less than two': 48000.0, 'between two and five': 61500.0}

In [2]:
```

2. For the above data there seems to be a correspondence between years of experience and paid accounts Users with very few and very many years of experience tend to pay; users with average amounts of experience don't. Find out the condition for this correspondence and print it.

Code

#Author : Dhrub khatry

#Regd.No. : 2141011043

from collections import defaultdict

salaries_and_tenures = [(83000, 8.7),(88000, 8.1),(48000, 0.7),(76000, 6),(69000, 6.5),(76000, 7.5),(60000, 2.5),(83000, 10),(48000, 1.9),(63000, 4.2)]

def predict_paid_or_unpaid(years_experience):

 if years_experience < 3.0:

  return "paid"

 elif years_experience < 8.5:

  return "unpaid"

 else:

  return "paid"

salary_per_tenure = defaultdict(list)

for salary, tenure in salaries_and_tenures:

   paid_or_unpaid = predict_paid_or_unpaid(tenure)

   salary_per_tenure[tenure].append(paid_or_unpaid)

res={}

for key, value in salary_per_tenure.items()

```
 res[key]=value[0]

print(res)
```

output

```
In [2]: runfile('/home/student/2141016146/a2.py', wdir='/home/student/2141016146')
{8.7: 'paid', 8.1: 'unpaid', 0.7: 'paid', 6: 'unpaid', 6.5: 'unpaid', 7.5: 'unpaid', 2.5: 'paid', 10:
'paid', 1.9: 'paid', 4.2: 'unpaid'}
```

3.State the difference between the following:

i. all() and any() ii.__ str__ and __repr__ function iii. sort and sorted iv. random.choice and random.sample.

ans: i. Both all() and any() are built-in functions in Python used to check conditions within iterables (like lists, tuples, or strings) but they differ in what they check for:

all(): This function returns True only if all elements in the iterable evaluate to True. If even a single element is False or any other falsy value (like 0, empty string, etc.), all() returns False. It also returns True for an empty iterable.

any(): This function returns True if any element in the iterable evaluates to True. If all elements are False or other falsy values, any() returns False. It also returns False for an empty iterable.

ii. _str_(): This method aims to provide a human-readable string representation of the object, intended for users of the class.

It should be clear and concise, and ideally, reflect the object's intended use.

This method is implicitly called by functions like print(), str(), and format().

repr(): This method aims to provide an unambiguous and informative string representation of the object, intended for developers.

It should allow developers to recreate the object from the string representation, usually in the form of a valid Python expression.

This method is implicitly called by the repr() function.

iii. Use sort() when you want to directly modify the list and don't need a separate sorted copy.

Use sorted() when you need a new sorted list while preserving the original iterable or when working with non-list iterables.

iv. Use random.choice when you need one random element and allowing duplicates is okay.

Use random.sample when you need multiple random elements without duplicates.

4. Write a Python Script to generate random passwords (alphanumeric). Ask users to enter the length of password and number of passwords they want to generate and then print all the generated passwords.

Code

```
#Author : Dhrub khatry

#Regd.No. : 2141011043

import random

import string

x=string.ascii_uppercase + string.ascii_lowercase +string.ascii_uppercase +string.digits

number = input('Number of passwords - ')

number = int(number)

length = input('password length? - ')

length = int(length)

for i in range(number):

 password = ''

 for j in range(length):

  password += random.choice(x)

 print(password)

 password += "\n"
```

output

```
In [3]: runfile('/home/student/2141016146/a4.py', wdir='/home/student/2141016146')

Number of passwords - 5

password length? - 8
QHO316UP
psEQ0f7R
olSToqUa
MZFaxZqF
KaXGYNVQ
```

5. What is Type Annotations? What are the uses of this in any python program? How to write this, give one example.

Ans:- Python type hints (annotations) add comments specifying data types (like str, int) for variables and functions. While not enforced, they enhance code readability, aid static type checking, and empower IDE features like autocompletion and error highlighting.

6. Read a lists named StringList1 containing strings from the key board. Generate a list MStringList1 that contains all items of StringList1 that are repeated twice or more number of times and print this list. By observing the outcome of MStringList1 perform the following tasks:

a. Check wather an item of MStringList1 occurs even number of times or odd number of times in StringList1.

b. Remove the i th (i ≥ 2) occurrence of a given word in a StringList1.

Code

```
#Author : Dhrub khatry

#Regd.No. : 2141011043

StringList1 = input("Enter strings separated by space: ").split()

MStringList1 = [item for item in set(StringList1) if StringList1.count(item) >= 2]

print("MStringList1:", MStringList1)

for item in MStringList1:

    occurrence_count = StringList1.count(item)

    print(f"{item} occurs {occurrence_count} times in StringList1. It occurs {'even' if occurrence_count % 2 == 0 else 'odd'} number of times.")

word_to_remove = input("Enter the word to remove: ")

if word_to_remove in StringList1:

    occurrences = [index for index, value in enumerate(StringList1) if value == word_to_remove]

    if len(occurrences) >= 2:

        index_to_remove = occurrences[1]  # Remove the 2nd occurrence

        StringList1.pop(index_to_remove)

print("Updated StringList1:", StringList1)
```

output

```
Enter strings separated by space: Dhrub is going to be an engineer soon as he is good at languages.
MStringList1: ['is']
is occurs 2 times in StringList1. It occurs even number of times.

Enter the word to remove: is
Updated StringList1: ['Dhrub', 'is', 'going', 'to', 'be', 'an', 'engineer', 'soon', 'as', 'he', 'good',
'at', 'languages.']
```

7. Count frequencies of various alphabets (Convert upper case into lower case and input given by user), plot the results for this as a bar chart with x-axis being the letter and y-axis as the corresponding frequency.

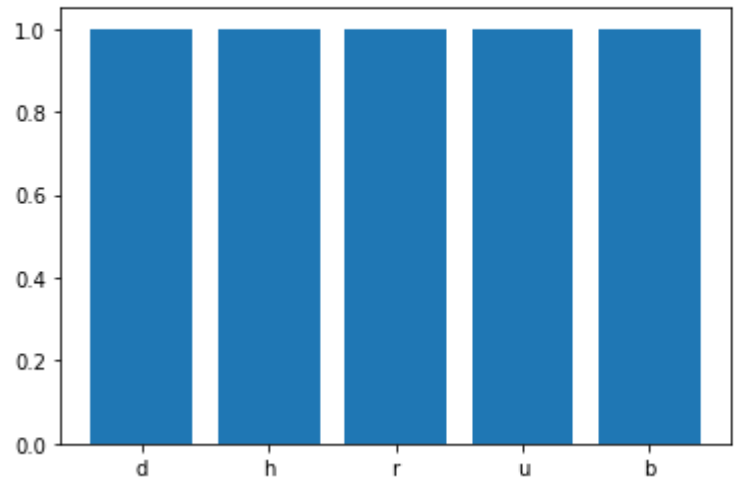
Code

```
#Author : Dhrub khatry

#Regd.No. : 2141011043
```

```
import matplotlib.pyplot as plt
f = input("Enter string :-  ")
freq = {}
for i in f:
 if i in freq:
  freq[i] += 1
 else:
  freq[i] = 1
freq = dict((k.lower(), v) for k, v in freq.items())
xaxis = list(freq.keys())
yaxis = list(freq.values())
plt.bar(range(len(freq)), yaxis, tick_label=xaxis)
plt.show()
```

output

```
Enter string :-  Dhrub
```



8. Use the following data to plot the number of applicant per year as a scatter plot.

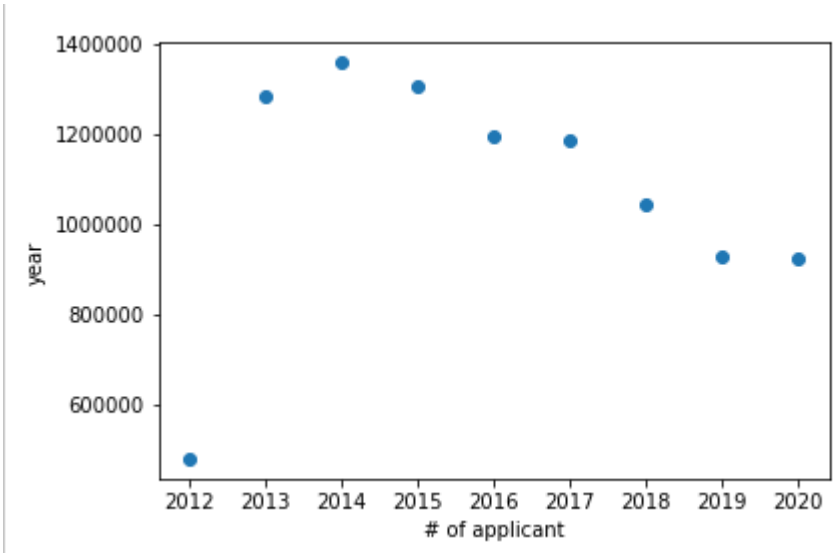
year = [2020, 2019, 2018, 2017, 2016, 2015, 2014, 2013, 2012] no application per year = [921261, 929198, 1043739, 1186454, 1194938, 1304495, 1356805, 1282000, 479651]

Code

```
#Author : Dhrub khatry
#Regd.No. : 2141011043
```

```
from matplotlib import pyplot as plt
year = [2020, 2019, 2018, 2017, 2016, 2015, 2014, 2013, 2012]
no_application_per_year = [921261, 929198, 1043739, 1186454,
1194938, 1304495, 1356805, 1282000, 479651]
plt.scatter(year,no_application_per_year)
plt.xlabel('# of applicant')
plt.ylabel('year')
plt.show()
```

output



9. Plot xsinx, x 2 sinx , x 3 sinx and x 4 sinx in a single plot in the range x ∈ [−10, 10].

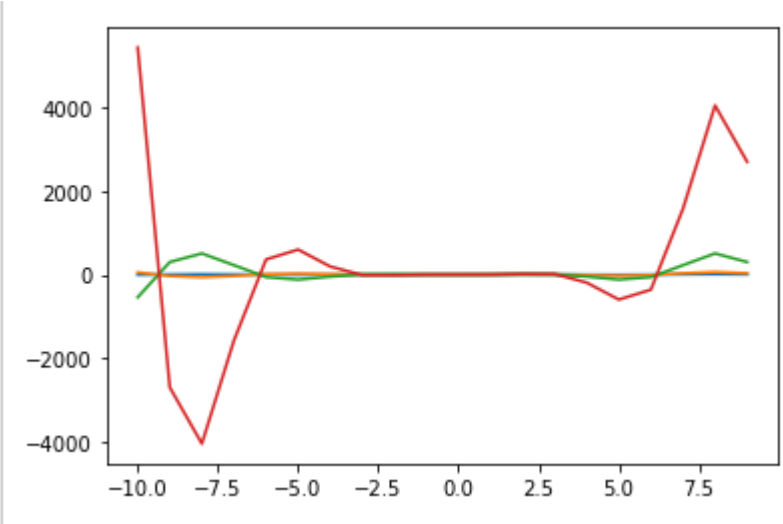
Code

```
#Author : Dhrub khatry
#Regd.No. : 2141011043
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(-10,10)
y = x*(np.sin(x))
y1 = x*x*(np.sin(x))
y2 = x*x*x*(np.sin(x))
y3 = x*x*x*x*(np.sin(x))
```

plt.plot(x,y,x,y1,x,y2,x,y3)

plt.show()

output



10. Plot histogram for age of male and female in different plots for the following data of male and female age. male age = [53,51,71,31,33,39,52,27,54,30,64,26,21,54,52,20,59,32] female age = [53,65,68,21,75,46,24,63,61,24,49,41,39,40,25,54,42, 32,48,23,23]

Code

#Author : Dhrub khatry

#Regd.No. : 2141011043

#Male

import matplotlib.pyplot as plt

male_age = [53,51,71,31,33,39,52,27,54,30,64,26,21,54,52,20,59,32]

male_age_set = set(male_age)

plt.hist(male_age, edgecolor='black')
plt.xlabel('age')

plt.ylabel('Male count')

plt.show()

#female

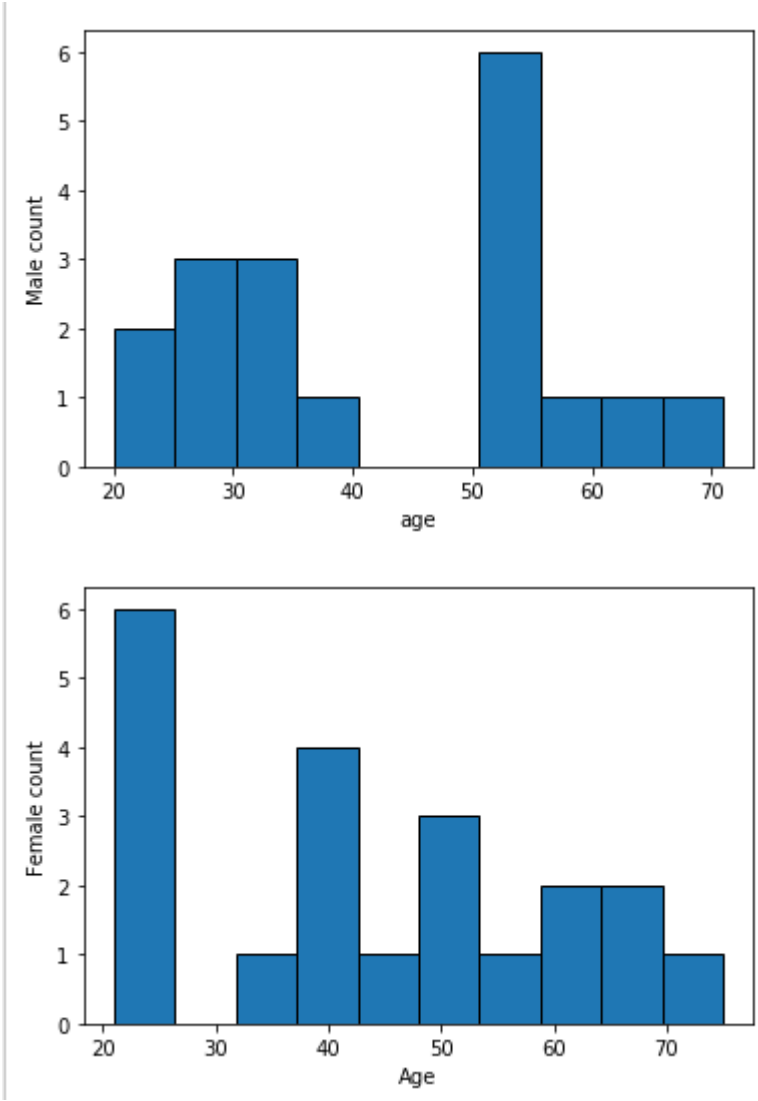
import matplotlib.pyplot as plt

female_age = [53,65,68,21,75,46,24,63,61,24,49,41,39,40,25,54,42,

32,48,23,23]

```
plt.hist(female_age, edgecolor='black')

plt.xlabel('Age')

plt.ylabel('Female count')

plt.show()
```

output



11. Plot the temperature extremes in certain region of India for each month, starting in January, which are given by (in degrees Celsius).

max: 17, 19, 21, 28, 33, 38, 37, 37, 31, 23, 19, 18 min: -62, -59, -56, -46, -32, -18, -9, -13, -25, -46, -52, -58

Code

```
#Author : Dhrub khatry
```

#Regd.No. : 2141011043

from matplotlib import pyplot as plt

x = [1,2,3,4,5,6,7,8,9,10,11,12]

max = [17, 19, 21, 28, 33, 38, 37, 37, 31, 23, 19, 18]

min = [-62, -59, -56, -46, -32, -18, -9, -13, -25, -46, -52, -58]

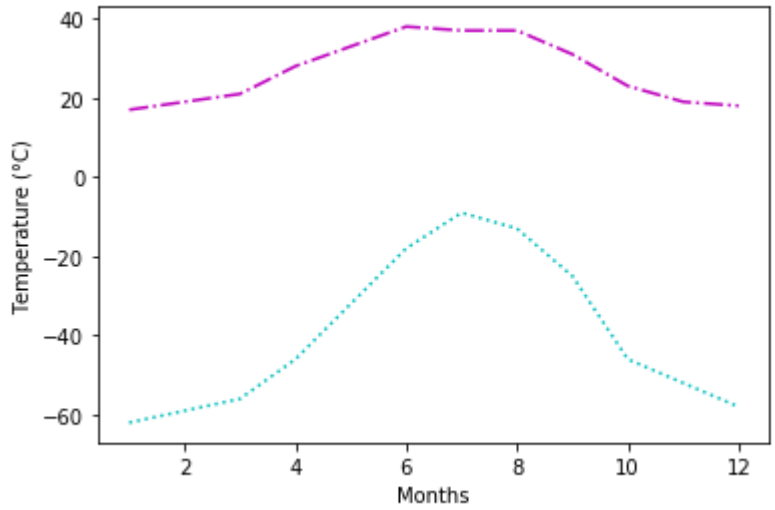
plt.xlabel('Months')

plt.ylabel('Temperature (°C)')

plt.plot(x,max, 'm-.', x, min, 'c:')

plt.show()

output



12. Python Program to find all Numbers in a Range (given by user) which are Perfect Squares and Sum of all Digits in the Number is Less than 10.

Code

#Author : Dhrub khatry

#Regd.No. : 2141011043
l=int (input ("Enter lower bound "))

u=int (input ("Enter upper bound "))

a= []

a= [x for x in range(l,u+1) if (int(x**0.5))**2==x and

sum(list(map(int,str(x))))<10]

print(a)

output

```
In [9]: runfile('/home/student/2141016146/a12.py', wdir='/home/student/2141016146')

Enter lower bound 30

Enter upper bound 300
[36, 81, 100, 121, 144, 225]
```

---

13. Plot a bar chart with axis labels for given data:

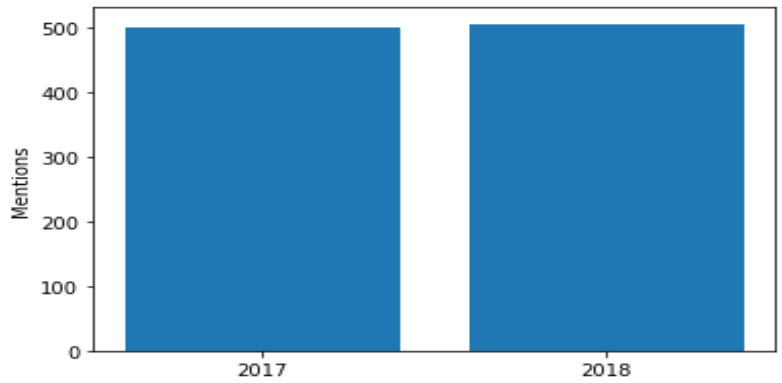
mentions = [500, 505] years = [2017, 2018]

Do not give any extra condition for x-axis as well as y-axis. Now again plot the bar chart for this data and start y-axis from 0. State the difference in both the bar chart.

Code

```
#Author : Dhrub khatry

#Regd.No.: 2141011043

from matplotlib import pyplot as plt

mentions = [500, 505]

years = [2017, 2018]

plt.bar(range(len(years)), mentions)

plt.ylim(bottom=0)

plt.ylabel("Mentions")

plt.xticks(range(len(years)),years)

plt.show()
```

output



14. Plot the scatter plot for following data with unequal axis and then equal axis. Also state the difference in two.

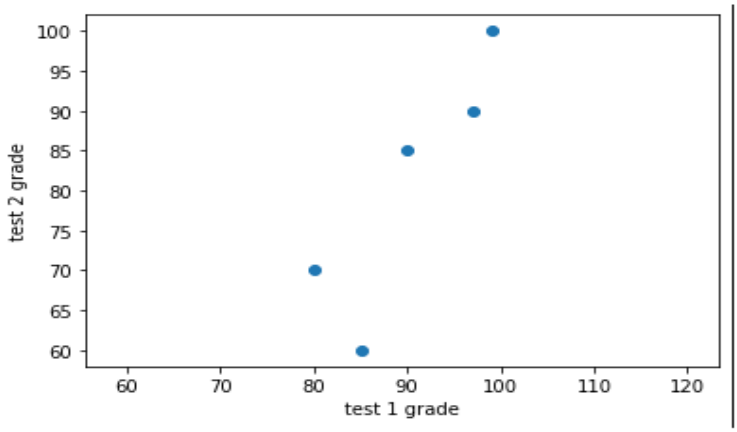
test 1 grades = [99, 90, 85, 97, 80]

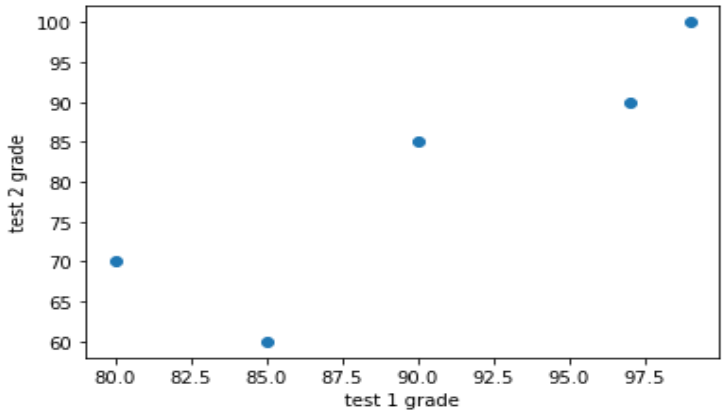
test 2 grades = [100, 85, 60, 90, 70]

Code

```
#Author : Dhrub khatry

#Regd.No. : 2141011043

#equal

from matplotlib import pyplot as plt

test_1_grades = [ 99, 90, 85, 97, 80]

test_2_grades = [100, 85, 60, 90, 70]
plt.scatter(test_1_grades, test_2_grades)

plt.xlabel("test 1 grade")

plt.ylabel("test 2 grade")

plt.axis("equal")

plt.show()

#unequal

from matplotlib import pyplot as plt

test_1_grades = [ 99, 90, 85, 97, 80]

test_2_grades = [100, 85, 60, 90, 70]

plt.scatter(test_1_grades, test_2_grades)

plt.xlabel("test 1 grade")

plt.ylabel("test 2 grade")
```

output


100
95
90
85
80
75
70
65
60
test 2 grade
60
70
80
90
100
110
120
test 1 grade


100
95
90
85
80
75
70
65
60
test 2 grade
80.0
82.5
85.0
87.5
90.0
92.5
95.0
97.5
test 1 grade