Q.1

```
name='''Hi How are you?
Starterd learning python.
It's really interesting.'''
```

```
name[:]
```

```
'Hi How are you?\nStarterd learning python.\nIt's really interesting.'
```

```
name[-10:-5]
```

```
'teres'
```

```
name[3:12]
```

```
'How are y'
```

```
name[12:3]
```

```
''
```

```
name[5:6]
```

```
'w'
```

```
name[-4:-12]
```

```
''
```

```
name[::2]
```

```
'H o r o?Satr erigpto.I' elyitrsig'
```

```
name[::-2]
```

```
'.nteen larst\nnhy nnaldert\nuyeawHi'
```

Q.2

```
L1=['a','b',20,30,'t',100,300,400,'Happy','major']
```

```
L1[:]
```

```
['a', 'b', 20, 30, 't', 100, 300, 400, 'Happy', 'major']
```

```
L1[::3]
```

```
['a', 30, 300, 'major']
```

```
L1[::-2]
```

d. How to extract value “Happy” based on index and negative index

```
L1[8]
```

```
'Happy'
```

```
L1[-2]
```

e. How to check type of data in list at 4th position

```
type(L1[4])
```

```
str
```

f. Extract values for 100, 300, 400

```
L1[5:8]
```

```
[100, 300, 400]
```

Q.3

```
L2 =[1,2,3,5,['a', 'b', 'work hard'],100 , 200, "Success"]
```

```
L2[4]
```

```
['a', 'b', 'work hard']
```

```
L2[1:5]
```

```
[2, 3, 5, ['a', 'b', 'work hard']]
```

```
L2[7]
```

```
'Success'
```

```
L2[7][2]
```

```
'c'
```

```
L2[7][2:]
```

```
'ccess'
```

```
L2[:3]
```

```
[1, 2, 3]
```

```
L2[3:]
```

```
[5, ['a', 'b', 'work hard'], 100, 200, 'Success']
```

4. From the above l2 value ‘b’ must be changed to ‘BEE’.

```
L2[4][1]='BEE'
L2
```

```
[1, 2, 3, 5, ['a', 'BEE', 'work hard'], 100, 200, 'Success']
```

5. From l2 “BEE” has to discard.

```
L2[4].remove('BEE')
L2
```

```
[1, 2, 3, 5, ['a', 'work hard'], 100, 200, 'Success']
```

6. In l2 add a dictionary at the end {‘insect’: [‘bee’, ‘moth’] , ‘bird’ : [‘parrot’, ‘sparrow’]}

```
L2.append({'insect':['bee','moth'],'bird':['parrot','sparrow']})
L2
```

```
[1,
 2,
 3,
 5,
 ['a', 'work hard'],
 100,
 200,
 'Success',
 {'insect': ['bee', 'moth'], 'bird': ['parrot', 'sparrow']}]
```

7. From l2 extract insect information.

```
L2[8]['insect']
# or
L2[-1]['insect']
```

```
['bee', 'moth']
```

8. Create a dictionary d1 = {'a':10, 'b':20, 'c' : 30} and add the d1 at 2nd position of l2

```
d1={'a':10,'b':20,'c':30}
L2.insert(2,d1)
L2
```

```
[1,
 2,
 {'a': 10, 'b': 20, 'c': 30},
 3,
 5,
 ['a', 'work hard'],
 100,
 200,
 'Success',
 {'insect': ['bee', 'moth'], 'bird': ['parrot', 'sparrow']}]
```

9. Based on new l2 created here extract the value 10 from l2 dictionary.

```
L2[2]['a']
```

```
10
```

10. If l2 =[1,2,3,5, (90,40,50,10), 'Python', 400 ,['a', 'b', 'work hard'],100 , 200, "Success", (200,300, "Hundreds")] then what is the output of following

```
L2 =[1,2,3,5, (90,40,50,10), 'Python', 400 ,['a', 'b', 'work hard'],100 , 200, "Success", (200,300, "Hundreds")]

L2
```

```
[1,
 2,
 3,
 5,
 (90, 40, 50, 10),
 'Python',
 400,
 ['a', 'b', 'work hard'],
 100,
 200,
 'Success',
 (200, 300, 'Hundreds')]
```

```
L2[4][2]
```

```
50
```

```
L2[5][:]
```

```
'Python'
```

```
L2[2]
```

```
3
```

```
L2[1:5]
```

```
[2, 3, 5, (90, 40, 50, 10)]
```

```
L2[5]
```

```
'Python'
```

```
L2[5][3:-1]
```

```
'ho'
```

```
L2[-1]
```

```
(200, 300, 'Hundreds')
```

```
L2[-4:-10]
```

```
[]
```

```
L2[7][2]
```

```
'work hard'
```

```
L2[-7][2:]
```

```
'thon'
```

```
L2[:-3]
```

```
[1, 2, 3, 5, (90, 40, 50, 10), 'Python', 400, ['a', 'b', 'work hard'], 100]
```

```
L2[-3:]
```

```
[200, 'Success', (200, 300, 'Hundreds')]
```

11. Ask user to enter the marks and define if the user got distinction, first class, second class, pass, Fails

If marks are more than 80 output must be “You got distinction”

For marks more than 60 output must be “You got First class”

Marks more than 40 will display “You got second class”

Marks more than or equal to 35 “PASS” Otherwise Fail

```
Marks=input("Enter the marks: ")
if (not Marks.isalpha() and not Marks.isalpha()) or Marks.isdigit():
  Marks=int(Marks)
  if Marks>100 or Marks<0:
    print("Enter correct marks")
  elif Marks >= 80:
    print("distiction")
  elif Marks >= 70 :
    print("First class")
  elif Marks >= 50 :
    print("Second Class")
  elif Marks == 35:
    print("pass")
  else:
    print("fail")
else:
  print("enter a correct Marks")
```

```
Enter the marks: 88
distiction
```

```
def check_marks():

  Marks=input("Enter the marks: ")
  if (not Marks.isalpha() and not Marks.isalpha()) or Marks.isdigit():
    Marks=int(Marks)
    if Marks>100 or Marks<0:
      return "Enter correct marks"
    elif Marks >= 80:
      return "Distiction"
    elif Marks >= 70 :
      return "First class"
    elif Marks >= 50 :
      return "Second Class"
    elif Marks >= 35:
      return "Pass"
    else:
      return "Fail"
  else:
    return "Enter a correct Marks"
check_marks()
```

12. Ask user to enter information about salary of the employee per year and rating received as A, B, C, D

If salary upto 5 lpa then increament based on ratings are A = 16% , B=12%, C= 10%, D=6%

If salary upto 10 lpa then increament based on ratings are A = 14% , B=10%, C= 8%, D=6%

If salary upto 15 lpa then increament based on ratings are A = 8% , B=6%, C= 4%, D = no increment

If salary upto 23 lpa then increament based on ratings are A = 7% , B=5%, C= 4%, D=no

```
salary=float(input("Enter the salary: "))
rating=input("Enter the rating: ").capitalize()

if salary > 0 and  rating in ['A','B','C','D']:

  if salary <=500000:
    if rating == 'A':
      print("Increment will be 16%:",salary*0.16)
    elif rating == 'B':
      print("Increment will be 12%:",salary*0.12)
    elif rating == 'C':
      print("Increment will be 10%:",salary*0.10)
    elif rating == 'D':
      print("Increment will be 6%:",salary*0.06)

  elif salary <=1000000:
    if rating == 'A':
      print("Increment will be 14%:",salary*0.14)
    elif rating == 'B':
      print("Increment will be 10%:",salary*0.10)
    elif rating == 'C':
     print("Increment will be 8%:",salary*0.08)
    elif rating == 'D':
      print("Increment will be 6%:",salary*0.06)

  elif salary <=1500000:
    if rating == 'A':
      print("Increment will be 8%:",salary*0.08)
    elif rating == 'B':
      print("Increment will be 6%:",salary*0.06)
    elif rating == 'C':
      print("Increment will be 4%:",salary*0.04)
    elif rating == 'D':
      print("No increment")


  elif salary <=2300000:
    if rating == 'A':
      print("Increment will be 7%:",salary*0.07)
    elif rating == 'B':
      print("Increment will be 5%:",salary*0.05)
    elif rating == 'C':
      print("Increment will be 4%:",salary*0.04)
    elif rating == 'D':
```

```
        print("No Increment")

else:
  print("please give correct value")
```

```
Enter the salary: 80000
Enter the rating: A
Increment will be 16%: 12800.0
```

```
def increment():
    salary = input("Enter the salary: ")
    rating = input("Enter the ratinga: ").upper()


    if not salary.isdigit() or int(salary) < 0 or rating not in ['A', 'B', 'C', 'D']:
        return "Please give correct value"


    salary = int(salary)

    if salary <= 500000:
        if rating == 'A':
            print("Increment will be 16%:", salary * 0.16)
        elif rating == 'B':
            print("Increment will be 12%:", salary * 0.12)
        elif rating == 'C':
            print("Increment will be 10%:", salary * 0.10)
        elif rating == 'D':
            print("Increment will be 6%:", salary * 0.06)

    elif salary <= 1000000:
        if rating == 'A':
            print("Increment will be 14%:", salary * 0.14)
        elif rating == 'B':
            print("Increment will be 10%:", salary * 0.10)
        elif rating == 'C':
            print("Increment will be 8%:", salary * 0.08)
        elif rating == 'D':
            print("Increment will be 6%:", salary * 0.06)

    elif salary <= 1500000:
        if rating == 'A':
            print("Increment will be 8%:", salary * 0.08)
        elif rating == 'B':
            print("Increment will be 6%:", salary * 0.06)
        elif rating == 'C':
            print("Increment will be 4%:", salary * 0.04)
        elif rating == 'D':
            print("No increment")

    elif salary <= 2300000:
        if rating == 'A':
            print("Increment will be 7%:", salary * 0.07)
        elif rating == 'B':
            print("Increment will be 5%:", salary * 0.05)
        elif rating == 'C':
            print("Increment will be 4%:", salary * 0.04)
        elif rating == 'D':
            print("No increment")
    else:
        print("Salary out of range")

increment()
```

```
Enter the salary: 80000
Enter the ratinga: A
Increment will be 16%: 12800.0
```

13. Ask user to opt for courses for master degree based on the following

L1 = [“HR”, “Finance”, “Marketing”, “DS”]

Based on above subject there are two different streams. For example- HR is having HR core and HR analytics and Marketing is having core and Marketing analytics. Analytics is the optional subject and having added extra fees. DS is not having analytics.

If fees for L1 is 2 lakhs for each course core subject having the same fees but analytics subject having 10% extra on 2 lakhs.

If student opts for hostel then 2 lakhs per year is added. For food monthly 2000 .

Transportation charges 13000 per semester. Calculate the total annual cost based on selected service.

User will enter values as subject, analytics(Y/N), Hostel (Y/N), food(How many months?), Transportation(semester/annual)

```
L1 = ["HR", "Finance", "Marketing", "DS"]

# Base fees
BASE_FEE = 200000
HOSTEL_FEE = 200000
FOOD_PER_MONTH = 2000
TRANSPORT_PER_SEM = 13000

print("Available Courses:", L1)

# User Inputs
subject = input("Enter subject: ").strip()
analytics = input("Analytics (Y/N): ").strip().upper()
hostel = input("Hostel (Y/N): ").strip().upper()
food_months = int(input("Food (How many months?): "))
transport = input("Transportation (semester/annual): ").strip().lower()

total_fee = 0

# Course Fee
if subject not in L1:
    print("Invalid Subject Selected")
    exit()

course_fee = BASE_FEE

# Analytics Fee Logic
if analytics == "Y":
    if subject in ["HR", "Marketing"]:
        course_fee += BASE_FEE * 0.10
    else:
        print("Analytics is not available for this subject")

total_fee += course_fee

# Hostel Fee
if hostel == "Y":
    total_fee += HOSTEL_FEE

# Food Fee
total_fee += food_months * FOOD_PER_MONTH

# Transport Fee
if transport == "semester":
    total_fee += TRANSPORT_PER_SEM
elif transport == "annual":
    total_fee += TRANSPORT_PER_SEM * 2

# Final Output
print("\n------ Fee Summary ------")
print("Subject:", subject)
print("Course Fee: ₹", int(course_fee))
print("Total Annual Cost: ₹", int(total_fee))
```

```
Available Courses: ['HR', 'Finance', 'Marketing', 'DS']
Enter subject: HR
Analytics (Y/N): Y
Hostel (Y/N): Y
Food (How many months?): 12
Transportation (semester/annual): SEMESTER

------ Fee Summary ------
Subject: HR
Course Fee: ₹ 220000
Total Annual Cost: ₹ 457000
```

```
# above code using OOPS

class CollegeFee:
```

```
    # Class Variables
    L1 = ["HR", "Finance", "Marketing", "DS"]
    BASE_FEE = 200000
    HOSTEL_FEE = 200000
    FOOD_PER_MONTH = 2000
    TRANSPORT_PER_SEM = 13000

    def __init__(self, subject, analytics, hostel, food_months, transport):
        self.subject = subject
        self.analytics = analytics
        self.hostel = hostel
        self.food_months = food_months
        self.transport = transport
        self.total_fee = 0
        self.course_fee = 0

    def calculate_course_fee(self):
        if self.subject not in CollegeFee.L1:
            print("Invalid Subject Selected")
            exit()

        self.course_fee = CollegeFee.BASE_FEE

        if self.analytics == "Y":
            if self.subject in ["HR", "Marketing"]:
                self.course_fee += CollegeFee.BASE_FEE * 0.10
            else:
                print("Analytics is not available for this subject")

        self.total_fee += self.course_fee

    def add_hostel_fee(self):
        if self.hostel == "Y":
            self.total_fee += CollegeFee.HOSTEL_FEE

    def add_food_fee(self):
        self.total_fee += self.food_months * CollegeFee.FOOD_PER_MONTH

    def add_transport_fee(self):
        if self.transport == "semester":
            self.total_fee += CollegeFee.TRANSPORT_PER_SEM
        elif self.transport == "annual":
            self.total_fee += CollegeFee.TRANSPORT_PER_SEM * 2

    def display_summary(self):
        print("\n------ Fee Summary ------")
        print("Subject:", self.subject)
        print("Course Fee: ₹", int(self.course_fee))
        print("Total Annual Cost: ₹", int(self.total_fee))


# ----------------- Execution -----------------

print("Available Courses:", CollegeFee.L1)

subject = input("Enter subject: ").strip()
analytics = input("Analytics (Y/N): ").strip().upper()
hostel = input("Hostel (Y/N): ").strip().upper()
food_months = int(input("Food (How many months?): "))
transport = input("Transportation (semester/annual): ").strip().lower()

student = CollegeFee(subject, analytics, hostel, food_months, transport)

student.calculate_course_fee()
student.add_hostel_fee()
student.add_food_fee()
student.add_transport_fee()
student.display_summary()
```

Q.14

14. Digitalize the book allotment process for school. Charges are mentioned here in the given table:

STANDARD	1st-4th	5th-8th	9-10th			
BOOKS				Notebook	100 pages	200 pages
Hindi	60	100	150	square	40	70
Marathi	60	100	150	4lines	30	50
English	80	100	150	2lines	30	50
Science	90	120	200	single lines	60	100
Maths	100	140	250	A4 notebook	100	180

```
# ---- STANDARD INPUT ----
s = int(input("Enter Standard (1-10): "))

if 1 <= s <= 4:
    standard = "1-4"
elif 5 <= s <= 8:
    standard = "5-8"
elif 9 <= s <= 10:
    standard = "9-10"
else:
    print("Invalid standard")
    exit()

# ---- PRICE DICTIONARIES ----
notebook_dict = {
    "1": 40,
    "2": 70,
    "3": 30,
    "4": 50,
    "5": 30,
    "6": 50,
    "7": 100,
    "8": 180
}

book_dict = {
    "1-4": {
        "hindi": 60, "marathi": 60, "english": 80,
        "science": 90, "maths": 100
    },
    "5-8": {
        "hindi": 100, "marathi": 100, "english": 100,
        "science": 120, "maths": 140
    },
    "9-10": {
        "hindi": 150, "marathi": 150, "english": 150,
        "science": 200, "maths": 250
    }
}

total_amount = 0

# ---- BOOKS OPTION ----
buy_books = input("Do you want to buy books? (yes/no): ").lower()

if buy_books == "yes":
    print("Available books: Hindi Marathi English Science Maths")
    subjects = input(
        "Enter the books you want (separated by space): "
    ).lower().split()

    for sub in subjects:
        if sub in book_dict[standard]:
            total_amount += book_dict[standard][sub]
        else:
            print(f"Invalid subject skipped: {sub}")

# ---- NOTEBOOK OPTION ----
buy_notebooks = input("Do you want to buy notebooks? (yes/no): ").lower()
```

```
if buy_notebooks == "yes":
    notebooks = input(
        """
Choose notebooks:
1 square 100 pages
2 square 200 pages
3 4lines 100 pages
4 4lines 200 pages
5 single_line 100 pages
6 single_line 200 pages
7 A4 notebook 100 pages
8 A4 notebook 200 pages

Enter notebook numbers (separated by space):
"""
    ).split()

    for nb in notebooks:
        if nb in notebook_dict:
            total_amount += notebook_dict[nb]
        else:
            print(f"Invalid notebook skipped: {nb}")

# ---- FINAL OUTPUT ----
print(f"\nTotal amount to pay: ₹{total_amount}")
```

```
Enter Standard (1-10): 2
Do you want to buy books? (yes/no): YES
Available books: Hindi Marathi English Science Maths
Enter the books you want (separated by space): Hindi
Do you want to buy notebooks? (yes/no): YES

Choose notebooks:
1 square 100 pages
2 square 200 pages
3 4lines 100 pages
4 4lines 200 pages
5 single_line 100 pages
6 single_line 200 pages
7 A4 notebook 100 pages
8 A4 notebook 200 pages

Enter notebook numbers (separated by space):
3

Total amount to pay: ₹90
```

```
# using def function


def get_standard():
    s = int(input("Enter Standard (1-10): "))
    if 1 <= s <= 4:
        return "1-4"
    elif 5 <= s <= 8:
        return "5-8"
    elif 9 <= s <= 10:
        return "9-10"
    else:
        print("Invalid standard")
        return None


def get_books_total(standard, book_dict):
    total = 0
    choice = input("Do you want to buy books? (yes/no): ").lower()

    if choice == "yes":
        print("Available books: Hindi Marathi English Science Maths")
        subjects = input(
            "Enter the books you want (separated by space): "
        ).lower().split()

        for sub in subjects:
            if sub in book_dict[standard]:
                total += book_dict[standard][sub]
            else:
```

```
                print(f"Invalid subject skipped: {sub}")

    return total


def get_notebooks_total(notebook_dict):
    total = 0
    choice = input("Do you want to buy notebooks? (yes/no): ").lower()

    if choice == "yes":
        notebooks = input(
            """
Choose notebooks:
1 square 100 pages
2 square 200 pages
3 4lines 100 pages
4 4lines 200 pages
5 single_line 100 pages
6 single_line 200 pages
7 A4 notebook 100 pages
8 A4 notebook 200 pages

Enter notebook numbers (separated by space):
"""
        ).split()

        for nb in notebooks:
            if nb in notebook_dict:
                total += notebook_dict[nb]
            else:
                print(f"Invalid notebook skipped: {nb}")

    return total


def main():
    book_dict = {
        "1-4": {
            "hindi": 60, "marathi": 60, "english": 80,
            "science": 90, "maths": 100
        },
        "5-8": {
            "hindi": 100, "marathi": 100, "english": 100,
            "science": 120, "maths": 140
        },
        "9-10": {
            "hindi": 150, "marathi": 150, "english": 150,
            "science": 200, "maths": 250
        }
    }

    notebook_dict = {
        "1": 40, "2": 70, "3": 30, "4": 50,
        "5": 30, "6": 50, "7": 100, "8": 180
    }

    standard = get_standard()
    if standard is None:
        return

    total = 0
    total += get_books_total(standard, book_dict)
    total += get_notebooks_total(notebook_dict)

    print(f"\nTotal amount to pay: ₹{total}")


# ---- RUN PROGRAM ----
main()
```

15. Create an interest bucket for Fix Deposit in the bank. Ask user to enter start date and end date for the FD and check which bucket list it belongs to and assign the interest rate

Tenors	Existing for Public w.e.f. 22.02.2019	Revised For Public w.e.f. 09.05.2019	Existing for Senior Citizens w.e.f. 22.02.2019	Revised for Senior Citizens w.e.f. 09.05.2019
7 days to 45 days	5.75	5.75	6.25	6.25
46 days to 179 days	6.25	6.25	6.75	6.75
180 days to 210 days	6.35	6.35	6.85	6.85
211 days to less than 1 year	6.40	6.40	6.90	6.90
1 year to less than 2 year	6.80	7.00	7.30	7.50
2 years to less than 3 years	6.80	6.75	7.30	7.25
3 years to less than 5 years	6.80	6.70	7.30	7.20
5 years and up to 10 years	6.85	6.60	7.35	7.10

```
from datetime import datetime

interest_buckets = [
    (7, 45, 5.75, 6.25),
    (46, 179, 6.25, 6.75),
    (180, 210, 6.35, 6.85),
    (211, 364, 6.40, 6.90),
    (365, 729, 7.00, 7.50),
    (730, 1094, 6.75, 7.25),
    (1095, 1824, 6.70, 7.20),
    (1825, 3650, 6.60, 7.10)
]


start_date = input("Enter Start Date (dd-mm-yyyy): ")
end_date = input("Enter End Date (dd-mm-yyyy): ")
senior = input("Senior Citizen? (Y/N): ").strip().upper()


start = datetime.strptime(start_date, "%d-%m-%Y")
end = datetime.strptime(end_date, "%d-%m-%Y")


total_days = (end - start).days

if total_days <= 0:
    print("Invalid date range")
    exit()


rate = None

for min_days, max_days, normal_rate, senior_rate in interest_buckets:
    if min_days <= total_days <= max_days:
        if senior == "Y":
            rate = senior_rate
        else:
            rate = normal_rate
        break


print("\n------ FD Summary ------")
print("Total Days:", total_days)

if rate:
    print("Applicable Interest Rate:", rate, "%")
else:
    print("No applicable interest bucket found")
```

Q.16

```
string = "In most organized forms of writing, such as essays, paragraphs contain a topic sentence. This topic sentence of the p
```

Q.17

```
a=100
type(a)
```

```
int
```

```
# conver to string

str_a=str(a)
print(str_a, type(str_a))
```

```
100 <class 'str'>
```

```
list_a=list(a)
print(list_a, type(list_a))

## we cant change
```

```
# convert to tuple
tuple_a=tuple(a)
print(tuple_a, type(tuple_a))

# we cant change to tuple
```

```
# change to dictionary

dict_a = dict(a)

# we cant change
```

```
# convert to set
set_a = set(a)

# we cant change , error will occure
```

```
# convert to float

float_a = float(a)
print(float_a, type(float_a))
```

8. Create city = “Pune”

· Convert to int

· Convert float

· Convert list

· Convert tuple

· Convert dict

· Convert set

· Obsert errors and note it down for all conversions

```
city='Pune'
type(city)
```

```
str
```

```
# · Convert to int

int_city=int(city)
print(int_city)


# we cant convert
```

```
# · Convert float

float_city=float(city)
print(float_city)


# cant convert
```

```
# convert to list

list_city=list(city)
print(list_city)
```

```
['P', 'u', 'n', 'e']
```

```
# convert to tuple

tuple_city=tuple(city)
print(tuple_city)
```

```
('P', 'u', 'n', 'e')
```

```
# convert to dict.

dict_city=dict(city)
print(dict_city)
```

```
# convert to set

set_city=set(city)
print(set_city)
```

```
{'u', 'e', 'n', 'P'}
```

9. Create a list as marks = [20,18,15,17,18]

· Convert to int

· Convert float

· Convert list

· Convert tuple

· Convert dict

· Convert set

· observe : errors and note it down for all conversions

```
marks=[20,18,15,17,18]
```

```
# convert to int

int_marks=int(marks)
print(int_marks)

# we cant change
```

```
# convert to float

float_marks=float(marks)
print(float_marks)

# we cant
```

```
# convert to list

list_marks=list(marks)
print(list_marks)
```

```
[20, 18, 15, 17, 18]
```

```
# convert to tuple

tuple_marks=tuple(marks)
print(tuple_marks)
```

```
(20, 18, 15, 17, 18)
```

```
# convert to dict

dict_marks=dict(marks)
print(dict_marks)

# we cant change to dict from list
```

```
# convert to set

set_marks=set(marks)
print(set_marks)
```

List operations

10. Create the empty list snames

· Add value 20 in the list using append

· Add value 30 in the list using extend

· Add values in the list using append

· Add value “WORK" in the list using extend

· Create a new list combo having the values [1, ‘a’, ‘b’ ,2 , 3]

· Add sname to combo using addition operator

· Add combo to snames using append

· Add combo to snames using extend.

```
snames=[]

snames.append(20)
print(snames)

snames.extend([30])
print(snames)

snames.append([40,50])
print(snames)

snames.extend(['WORK'])
print(snames)

new_list=[1,'a','b',2,3]

combo=snames+new_list
print(combo)

snames.append(combo)
print(snames)

snames.extend(combo)
print(snames)
```

```
[20]
[20, 30]
[20, 30, [40, 50]]
[20, 30, [40, 50], 'WORK']
```

```
[20, 30, [40, 50], 'WORK', 1, 'a', 'b', 2, 3]
[20, 30, [40, 50], 'WORK', [20, 30, [40, 50], 'WORK', 1, 'a', 'b', 2, 3]]
[20, 30, [40, 50], 'WORK', [20, 30, [40, 50], 'WORK', 1, 'a', 'b', 2, 3], 20, 30, [40, 50], 'WORK', 1, 'a', 'b', 2, 3]
```

11. Create one list l1 having two elements and l3 having 7 elements. Now at 4th position add l1

```
l1=[1,2]
l2=[1,2,3,4,5,6,7]

l2.insert(4,l1)
print(l2)

l2[3:3] = l1
print(l2)
```

```
[1, 2, 3, 4, [1, 2], 5, 6, 7]
[1, 2, 3, 1, 2, 4, [1, 2], 5, 6, 7]
```

12. Collection is the list having values [1,2,3,['a', 'b', 'c'], 100, 'Nisha', 20.50, 90.10] if the data is in integer or float then multiply with 5.

· From the collection delete the record for "Nisha"

· Find the location of 20.50

```
list1=[1,2,3,['a','b','c'],100,'Nisha',20.50,90.10]

print(list1)
print('position of 20.50 :',list1.index(20.50))
for i in range(len(list1)):
    if type(list1[i]) == int or type(list1[i]) == float:
        list1[i] = list1[i] * 5

print(list1)

list1.remove('Nisha')
print(list1)
```

```
[1, 2, 3, ['a', 'b', 'c'], 100, 'Nisha', 20.5, 90.1]
position of 20.50 : 6
[5, 10, 15, ['a', 'b', 'c'], 500, 'Nisha', 102.5, 450.5]
[5, 10, 15, ['a', 'b', 'c'], 500, 102.5, 450.5]
```

13. Create a comprehensive list for square upto 10

```
squares=[ x**2 for x in range (1,11)]
squares
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

14. Create the comprehensive list to find number divisible by 13 till 200

```
squares=[ (x, x%13==0) for x in range (1,20)  ]
print(squares)

# method 2

square=[ x for x in range (1,200) if x%13==0 ]
square
```

```
[(1, False), (2, False), (3, False), (4, False), (5, False), (6, False), (7, False), (8, False), (9, False), (10, False), (11, F
[13, 26, 39, 52, 65, 78, 91, 104, 117, 130, 143, 156, 169, 182, 195]
```

15. Create the list which is divisible by 4 from 300 to 400

```
square=[ x for x in range (300,400) if x%4==0 ]
square
```

```
[300,
 304,
 308,
 312,
 316,
 320,
 324,
 328,
 332,
 336,
 340,
 344,
 348,
 352,
 356,
 360,
 364,
 368,
 372,
 376,
 380,
 384,
 388,
 392,
 396]
```

16. Create a comprehensive list to generate list up to x, y as a number. For example if x = 3 list will be x_list = [0,1,2] and y=2 then y_list = [0,1]

Then generate a new list based on all combination of x and y.

For example: if x = 1 and y =2 then x_list = [0] and y_list = [0,1]

And output will be [[0,0],[0,1]]

If x=2 and y = 2 then output will be [[0,0],[0,1][1,0],[1,1]]

```
# 21. How to create empty set?

set1={}
print(set1)
```

```
{}
```

22. Create the set s1 and s2 and perform set operations like union, intersection, difference, set difference.

```
s1={1,2,3,4,5}
s2={4,5,6,7,8}

print(s1 | s2)
print(s1 & s2)
print(s1 - s2)
print(s2 - s1)
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
{4, 5}
{1, 2, 3}
{8, 6, 7}
```

23. Create l2 as a list and perform set operation on s1 with l2

```
l2=[4,5,6,7]

print(s1 | l2)
print(s1 & l2)
print(s1 - l2)
print(l2 - s1)


# we cant perform set operations on list and set
# error will occure
```

24. Ask user to enter the name and DOB and generate the password based on first name 4 characters and @ddmm. For example name = SURESH and DOB = 05-05-1978 then password will be SURE@0505

```
name=input("Enter the name:")
DOB=input("Enter the DOB in dd-mm-yyyy:")

print(DOB)

password=name[:4]+'@'+DOB[:2]+DOB[3:5]
print("Password is : ",password)
```

```
Enter the name:Rohit
Enter the DOB in dd-mm-yyyy:16-02-2002
16-02-2002
Password is :  Rohi@1602
```

25. Ask user to enter the name and DOB and generate the password based on first name 4 characters and last 4 digits of DOB or year @yyyy. For example name = SURESH and DOB = 05-05-1978 then password will be SURE@1978

```
name=input("Enter the name:")
DOB=input("Enter the DOB in dd-mm-yyyy:")

print(DOB)

password=name[:4]+'@'+DOB[6:]
print("Password is : ",password)
```

```
Enter the name:Rohit
Enter the DOB in dd-mm-yyyy:16-02-2002
16-02-2002
Password is :  Rohi@2002
```

26. Create the format mentioned here.

-
- ○

```
for i in range(5):
  print("* "*i)
```

```
*
* *
* * *
* * * *
```

```
n=5
for i in range(n):
  for j in range(i):
    print("*",end=" ")
  print()
```

```
*
* *
* * *
* * * *
```

27. Create the format as mentioned below:

**

```
n=5
for i in range(5):
  for j in range(n-i):
    print("*",end=' ')
  print()
```

```
* * * * *
* * * *
* * *
* *
*
```

28. Str_val = “ABCD” then create the format as mentioned below:

A

A B

A B C

A B C D

```
str_val='ABCD'
str1=""

for i in range(len(str_val)):
  str1=str1+str_val[i]
  print(str1)
```

```
A
AB
ABC
ABCD
```

29. Create the format mentioned below:

A

BB

CCC

DDDD

```
s='ABCD'

for i  in range (len (s)):
  print(s[i]*(i+1))
```

```
A
BB
CCC
DDDD
```

30. Create the format mentioned below:

1

22

333

4444

```
s = "1234"

for i in range(len(s)):
    print(s[i] * (i + 1))
```

```
1
22
333
4444
```

31. Val = “ABCD” based on the Val, create the format mentioned below

D

DC

DCB

DCBA

```
str_val='ABCD'
str1=""

for i in range(len(str_val),i):
  str1=str1+str_val[i]
  print(str1)
```

32. Ask user to enter the string. If string is UPGRAD then output must be

D

DA

DAR

DARG

DARGP

DARGPU

```
str_val='UPGRADE'
str1=""

for i in range(len(str_val)):
  str1=str1+str_val[i]
  print(str1)
```

```
U
UP
UPG
UPGR
UPGRA
UPGRAD
UPGRADE
```

33. Create a list of odd numbers from 1 to 10

Using for loop

Using comprehensive list

```
# using for loop

odd_list=[]
for i in range(1,11):
  if i%2!=0:
    odd_list.append(i)

print(odd_list)


# using list comprehensive

odd_list2 = [ x for x in  range(1,11) if x%2!=0]
print(odd_list2)
```

```
[1, 3, 5, 7, 9]
[1, 3, 5, 7, 9]
```

34. Create even number list using for loop from 200 to 250

```
even_list=[]
for i in range(200,251):
  if i%2!=0:
```

```
        even_list.append(i)

    print(even_list)
```

```
[201, 203, 205, 207, 209, 211, 213, 215, 217, 219, 221, 223, 225, 227, 229, 231, 233, 235, 237, 239, 241, 243, 245, 247, 249]
```

35. List2 = [2,70,'work', para, 2.5, [1,2,3], (1,2), {1,2}, {1:'a', 2:'b'}, 3,10,302.5]

Multiply each and every element by 2 and display the answer

```
List2 = [2, 70, 'work', 'para', 2.5, [1,2,3], (1,2), {1,2}, {1:'a', 2:'b'}, 3, 10, 302.5]

result = []

for item in List2:
    if isinstance(item, (int, float, str, list, tuple)):
        result.append(item * 2)
    elif isinstance(item, set):
        result.append(item.union(item))   # duplicate elements
    elif isinstance(item, dict):
        new_dict = {}
        for k, v in item.items():
            new_dict[k * 2] = v * 2
        result.append(new_dict)

print(result)
```

```
[4, 140, 'workwork', 'parapara', 5.0, [1, 2, 3, 1, 2, 3], (1, 2, 1, 2), {1, 2}, {2: 'aa', 4: 'bb'}, 6, 20, 605.0]
```

36. List2 = [2,70,'work', para, 2.5, [1,2,3], (1,2), {1,2}, {1:'a', 2:'b'}, 3,10,302.5]

Multiply each and every element from list2 by 2 and store the answer in list3

```
list2 = [2, 70, 'work', 'para', 2.5, [1,2,3], (1,2), {1,2}, {1:'a', 2:'b'}, 3, 10, 302.5]

list3 = []

for item in list2:
    if isinstance(item, (int, float, str, list, tuple)):
        list3.append(item * 2)

    elif isinstance(item, set):
        list3.append(item.union(item))   # sets don't allow duplicates

    elif isinstance(item, dict):
        new_dict = {}
        for k, v in item.items():
            new_dict[k * 2] = v * 2
        list3.append(new_dict)

print(list3)
```

```
[4, 140, 'workwork', 'parapara', 5.0, [1, 2, 3, 1, 2, 3], (1, 2, 1, 2), {1, 2}, {2: 'aa', 4: 'bb'}, 6, 20, 605.0]
```

37. Create a function to accept marks from user utilize exception concept to validate proper marks.

```
def check_marks():

  Marks=input("Enter the marks: ")
  if (not Marks.isalpha() and not Marks.isalpha()) or Marks.isdigit():
    Marks=int(Marks)
    if Marks>100 or Marks<0:
      return "Enter correct marks"
    elif Marks >= 80:
      return "Distiction"
    elif Marks >= 70 :
      return "First class"
    elif Marks >= 50 :
      return "Second Class"
    elif Marks >= 35:
      return "Pass"
```

```
    else:
      return "Fail"
  else:
    return "Enter a correct Marks"
check_marks()
```

```
Enter the marks: 77
'First class'
```

```
# USING EXEPTION HANDLING


def check_marks():
    try:
        marks = int(input("Enter the marks: "))

        if marks > 100 or marks < 0:
            return "Enter correct marks"
        elif marks >= 80:
            return "Distinction"
        elif marks >= 70:
            return "First Class"
        elif marks >= 50:
            return "Second Class"
        elif marks >= 35:
            return "Pass"
        else:
            return "Fail"

    except ValueError:
        return "Enter a correct marks"

check_marks()
```

```
Enter the marks: 88
'Distinction'
```

38. Create a function to validate user first name/last name. User first name/last name should contain only characters. No special characters, numbers, space in name

```
def check_name():
  name=input("Enter the name: ")
  if name.isalpha():

    return 'Valid Name'
  else:
    return 'Enter the correct name'

check_name()
```

```
Enter the name: rohit
'Valid Name'
```

39. Create a function to accept mobile number. Mobile number should contain 10 digits. No Special character, alphabets and space.

```
def Check_No():
  no=input("Enter the name: ")
  if no.isdigit() and len(no)==10:
    no=int()
    return 'Accept the Mobile No.'
  else:
    return 'Enter the Correct No.'

Check_No()
```

```
Enter the name: 9876543210
'Accept the Mobile No.'
```

40. Create a function to generate auto-password based on specific person details. Ask user to enter name, DOB. And password must be First name 4 characters and year of birth.

```
def Check_Password():

  name=input ("Enter the name: ")
  dob=input("Enter the dob in dd-mm-yyyy :")

  if not name.isalpha() or len(dob)!=10 or dob[2]!='-' or dob[5]!='-':
    return "Please enter in proper format"
  else:
    password =  name[:4]+dob[6:]
    return password

Check_Password()
```

```
Enter the name: rohit
Enter the dob in dd-mm-yyyy :16-02-2002
'rohi2002'
```

41. Create a empty dictionary and ask user to enter values as name, DOB, mobile number add all the details in dictionary with customer number as 1 for first time. If user try to enter another value, then number should increase as 2 with new details and previous values should not change.

For example:

{}

{1:{name : "Sachin", "DOB": "21-06-1965" , "mobile": "1234123423"}}

{1:{name : "Sachine", "DOB": "21-06-1965" , "mobile": "1234123423"},

2: {name : "Sumedh", "DOB": "02-02-2002" , "mobile": "1234123433"}}

```
customers = {}
cust_no = 1

while True:
    name = input("Enter Name: ")
    dob = input("Enter DOB (DD-MM-YYYY): ")
    mobile = input("Enter Mobile: ")

    customers[cust_no] = {"name": name, "DOB": dob, "mobile": mobile}
    cust_no += 1

    more = input("Add more? (y/n): ")
    if more.lower() != 'y':
        break

print(customers)
0
```

```
Enter Name: rohit
Enter DOB (DD-MM-YYYY): 16-02-2002
Enter Mobile: 9876543211
Add more? (y/n): y
Enter Name: shraddha
Enter DOB (DD-MM-YYYY): 03-03-2002
Enter Mobile: 9876543219
Add more? (y/n): n
{1: {'name': 'rohit', 'DOB': '16-02-2002', 'mobile': '9876543211'}, 2: {'name': 'shraddha', 'DOB': '03-03-2002', 'mobile': '9876
```

42. Based on the above example create the dictionary and save the same in a cust_info.txt or cust_info.log

```
with open("cust_info.txt", "w") as f:
    f.write(str(customers))
```

43. Based on the above example read the file cust_info.txt . check if dictionary any information is available in the file. If there is information then read the dictionary store into one variable and then append new information of customer if added.

```

```

```
import os

if os.path.exists("cust_info.txt"):
    with open("cust_info.txt", "r") as f:
        data = f.read()
        customers = eval(data) if data else {}
else:
    customers = {}

cust_no = max(customers.keys()) + 1 if customers else 1

name = input("Enter Name: ")
dob = input("Enter DOB: ")
mobile = input("Enter Mobile: ")

customers[cust_no] = {"name": name, "DOB": dob, "mobile": mobile}

with open("cust_info.txt", "w") as f:
    f.write(str(customers))
```

```
Enter Name: rohit
Enter DOB: 16-02-2002
Enter Mobile: 9876543211
```

44. Create a table cust_info as sr_no, name, DOB, mobile. Ask user to enter the information from python code. Validate all fields and after validation insert records in the table.

```
#import mysql.connector
import re

conn = mysql.connector.connect(host="localhost", user="root", password="root", database="test")
cur = conn.cursor()

name = input("Enter Name: ")
dob = input("Enter DOB (YYYY-MM-DD): ")
mobile = input("Enter Mobile: ")

if not re.match(r"\d{10}$", mobile):
    print("Invalid Mobile")
else:
    cur.execute("INSERT INTO cust_info (name, DOB, mobile) VALUES (%s,%s,%s)", (name, dob, mobile))
    conn.commit()
    print("Inserted Successfully")
```

45. Dict1= {“Key”: {“subkey”:20} , “k2”:{“sub2” : 5}, “k3” : {“sub4” :16}, “k4” : {“sub4” : 6}}

Sort elements based on values

Output must be {, “k2”:{“sub2” : 5}, “k4” : {“sub4” : 6}, “k3” : {“sub4” : 16}, “Key”:{“subkey”:20}}

```
Dict1 = {"Key":{"subkey":20}, "k2":{"sub2":5}, "k3":{"sub4":16}, "k4":{"sub4":6}}

sorted_dict = dict(sorted(Dict1.items(), key=lambda x: list(x[1].values())[0]))
print(sorted_dict)
```

```
{'k2': {'sub2': 5}, 'k4': {'sub4': 6}, 'k3': {'sub4': 16}, 'Key': {'subkey': 20}}
```

46. Create a function to calculate age till now.

```
from datetime import datetime

def calculate_age(dob):
    dob = datetime.strptime(dob, "%d-%m-%Y")
    today = datetime.today()
    return today.year - dob.year - ((today.month, today.day) < (dob.month, dob.day))

print(calculate_age("21-06-2000"))
```

```
25
```

47. Create a function to check age eligibility for given customer based on DOB. Function will take two input DOB and ELIGIBILITY age.

```
def check_eligibility(dob, eligibility_age):
    age = calculate_age(dob)
    if age >= eligibility_age:
        return "Eligible"
    return "Not Eligible"

print(check_eligibility("21-06-2000", 18))
```

```
Eligible
```

48. Create a function to check if string is palindrome or not ? For example, if input is NITIN then reverse of the string is same then it is palindrome. If input is ANIL then reverse is LINA which is not same then it is not palindrome.

```
def is_palindrome(s):
    return s == s[::-1]

print(is_palindrome("NITIN"))
```

```
True
```

49. Create a function to generate a Fibonacci Series. 0 1 1 2 3 5 8 13 21 34 ..... upto 100

```
def fibonacci():
    a, b = 0, 1
    while a <= 100:
        print(a, end=" ")
        a, b = b, a+b

fibonacci()
```

```
0 1 1 2 3 5 8 13 21 34 55 89
```

50. Write a code to generate factorial of the number For example: factorial of 5 = 5! = 5*4*3*2*1

```
def factorial(n):
    fact = 1
    for i in range(1, n+1):
        fact *= i
    return fact

print(factorial(5))
```

```
120
```

51. Write a program to find largest number in the list.

```
lst = [10, 20, 5, 45, 30]
print(max(lst))
```

```
45
```

52. Write a program to check frequency of each element in the list.

```
lst = [1,2,2,3,3,3,4]

freq = {}
for i in lst:
    freq[i] = freq.get(i,0) + 1

print(freq)
```

```
{1: 1, 2: 2, 3: 3, 4: 1}
```

53. There are two string l1 =[1,2,3,4,5] and l2 =[3,2,8,7,9] then write a program to find common elements in the list.

```
l1 = [1,2,3,4,5]
l2 = [3,2,8,7,9]

common = list(set(l1) & set(l2))
print(common)
```

```
[2, 3]
```

54. Create 3 tables in mysql having the columns as mentioned

1. user : uid, mobile, passkey, date
2. Product : pid, pname
3. Order : oid, uid, pid

user having attribute uid, mobile

method user sign up

sign in

view_products

view_orders

place_order

Based on the OOPs concept create classes, methods, attributes and insert information

```
import mysql.connector

class User:
    def __init__(self, uid, mobile, password):
        self.uid = uid
        self.mobile = mobile
        self.password = password

    def sign_up(self):
        cur.execute("INSERT INTO user (uid, mobile, passkey, date) VALUES (%s,%s,%s,NOW())",
                    (self.uid, self.mobile, self.password))
        conn.commit()
        print("Signup Successful")

    def sign_in(self):
        cur.execute("SELECT * FROM user WHERE uid=%s AND passkey=%s",
                    (self.uid, self.password))
        if cur.fetchone():
            print("Login Successful")
        else:
            print("Invalid Login")

    def view_products(self):
        cur.execute("SELECT * FROM Product")
        for row in cur.fetchall():
            print(row)

    def place_order(self, pid):
        cur.execute("INSERT INTO Order (uid, pid) VALUES (%s,%s)",
                    (self.uid, pid))
        conn.commit()
        print("Order Placed")

    def view_orders(self):
        cur.execute("SELECT * FROM Order WHERE uid=%s", (self.uid,))
        for row in cur.fetchall():
            print(row)


conn = mysql.connector.connect(host="localhost", user="root", password="root", database="test")
cur = conn.cursor()
```

56. Build a function to print bill. Generate the bill as mentioned below.

```
menu=input("Enter the menue:")
quantity=int(input("Enter the quantity:"))
price=int(input("Enter the price:"))



print('-----------------------------------------------')
print("|{:^46}".format("Welcome Hotel Bhagyashree ","       |"))
print('-----------------------------------------------')
print("| {:<3} {:<15} {:<10} {:<10} |".format("sr", "Menu", "qunt", "price"))
print("| {:<3} {:<15} {:<10} {:<10} |".format(1, menu, quantity, price))


print('-----------------------------------------------')
print('|Total',price*quantity,"                                |" )

print('-----------------------------------------------')
```

```
Enter the menue:Idli
Enter the quantity:2
Enter the price:60
-----------------------------------------------
|          Welcome Hotel Bhagyashree
-----------------------------------------------
| sr  Menu            qunt       price      |
| 1   Idli            2          60         |
-----------------------------------------------
|Total 120                                   |
-----------------------------------------------
```

```
hotel_name = input("Enter Hotel Name: ")
menu = input("Enter Menu Item: ")
quantity = int(input("Enter Quantity: "))
price = int(input("Enter Price per Item: "))

total = quantity * price

print("\n" + "-" * 50)
print("|{:^48}|".format("Welcome " + hotel_name))
print("-" * 50)
print("| {:<3} {:<15} {:<10} {:<10}      |".format("sr", "Menu", "qunt", "price"))
print("| {:<3} {:<15} {:<10} {:<10}      |".format(1, menu, quantity, price))
print("-" * 50)
print("| {:<30} {:>15} |".format("Total", total))
print("-" * 50)
```

```
Enter Hotel Name: Hotel Bhagyashree
Enter Menu Item: Idli
Enter Quantity: 2
Enter Price per Item: 60

-------------------------------------------------
|           Welcome Hotel Bhagyashree           |
-------------------------------------------------
| sr  Menu            qunt       price          |
```