NIZAM COLLEGE
(AUTONOMOUS)

MCA II Semester
Data Engg. With Python (LCC252)
Lab Record

OSMANIA UNIVERSITY
HYDERABAD
2024

NIZAM COLLEGE

(Autonomous)

Basheerbagh, Hyderabad.

Ph. No: 040-23233317

CERTIFICATE

Department of Informatics

This is to certify that _____

bearing Hall Ticket No. _____

student of MCA II Semester has completed **Data Engg. With Python (LCC252)**

Lab Record work during the year 2023-2024.

Internal Examiner External Examiner

Head of the Department

Table of Contents

1. Write a program to demonstrate different numbers data types in python

```
print(2 + 3)   # addition(+)
print(3 - 1)   # subtraction(-)
print(2 * 3)   # multiplication(*)
print(3 / 2)   # division(/)
print(3 ** 2)  # exponential(**)
print(3 % 2)   # modulus(%)
print(3 // 2)  # Floor division operator(//)

# Checking data types

print(type(10))              # Int
print(type(3.14))            # Float
print(type(1 + 3j))          # Complex
print(type('Asabeneh'))      # String
print(type([1, 2, 3]))       # List
print(type({'name':'Asabeneh'})) # Dictionary
print(type({9.8, 3.14, 2.7}))   # Set
print(type((9.8, 3.14, 2.7)))   # Tuple

Int = 10
Float = 3.14
Complex = 1 + 3j
String = 'Asabeneh'
List = [1, 2, 3]
Dictionary = {'name':'Asabeneh'}
Set = {9.8, 3.14, 2.7}
Tuple = (9.8, 3.14, 2.7)
print(type(Int))          # Int
print(type(Float))        # Float
print(type(Complex))      # Complex
print(type(String))       # String
print(type(List))         # List
print(type(Dictionary))   # Dictionary
print(type(Set))          # Set
print(type(Tuple))        # Tuple
```

Output

```
5
2
6
1.5
9
```

1
1
<class 'int'>
<class 'float'>
<class 'complex'>
<class 'str'>
<class 'list'>
<class 'dict'>
<class 'set'>
<class 'tuple'>

<class 'int'>
<class 'float'>
<class 'complex'>
<class 'str'>
<class 'list'>
<class 'dict'>
<class 'set'>
<class 'tuple'>

2. Write a python program to work with variables

```
first_name = 'Asabeneh'
last_name = 'Yetayeh'
country = 'Finland'
city = 'Helsinki'
age = 50
is_married = True
skills = ['HTML', 'CSS', 'JS', 'React', 'Python']
person_info = {
    'firstname':'Asabeneh',
    'lastname':'Yetayeh',
    'country':'Finland',
    'city':'Helsinki'
    }

# Printing the values stored in the variables

print('First name:', first_name)
print('First name length:', len(first_name))
print('Last name: ', last_name)
print('Last name length: ', len(last_name))
print('Country: ', country)
print('City: ', city)
print('Age: ', age)
print('Married: ', is_married)
print('Skills: ', skills)
print('Person information: ', person_info)

# Declaring multiple variables in one line

first_name, last_name, country, age, is_married = 'Asabeneh', 'Yetayeh', 'Helsink', 50, True

print(first_name, last_name, country, age, is_married)
print('First name:', first_name)
print('Last name: ', last_name)
print('Country: ', country)
print('Age: ', age)
print('Married: ', is_married)
```

Output

```
First name: Asabeneh
First name length: 8
Last name:  Yetayeh
Last name length:  7
Country:  Finland
City:  Helsinki
Age:  50
Married:  True
Skills:  ['HTML', 'CSS', 'JS', 'React', 'Python']
Person information:  {'firstname': 'Asabeneh', 'lastname': 'Yetayeh', 'country': 'Finland', 'city': 'Helsinki'}
Asabeneh Yetayeh Helsink 50 True
First name: Asabeneh
Last name:  Yetayeh
Country:  Helsink
Age:  50
Married:  True
```

3. Write a python program to implement arithmetic operations

```
# Integers
print('Addition: ', 1 + 2)
print('Subtraction: ', 2 - 1)
print('Multiplication: ', 2 * 3)
print ('Division: ', 4 / 2)                    # Division in python gives floating number
print('Division: ', 6 / 2)
print('Division: ', 7 / 2)
print('Division without the remainder: ', 7 // 2)   # gives without the floating number or without the remai
ning
print('Modulus: ', 3 % 2)                      # Gives the remainder
print ('Division without the remainder: ', 7 // 3)
print('Exponential: ', 3 ** 2)                 # it means 3 * 3

# Floating numbers
print('Floating Number,PI', 3.14)
print('Floating Number, gravity', 9.81)

# Complex numbers
print('Complex number: ', 1 + 1j)
print('Multiplying complex number: ',(1 + 1j) * (1-1j))

# Declaring the variable at the top first
a = 3 # a is a variable name and 3 is an integer data type
b = 2 # b is a variable name and 3 is an integer data type

# Arithmetic operations and assigning the result to a variable
total = a + b
diff = a - b
product = a * b
division = a / b
remainder = a % b
floor_division = a // b
exponential = a ** b

# I should have used sum instead of total but sum is a built-
in function try to avoid overriding builtin functions
print(total) # if you don't label your print with some string, you never know from where is  the result is co
ming
print('a + b = ', total)
print('a - b = ', diff)
print('a * b = ', product)
print('a / b = ', division)
print('a % b = ', remainder)
print('a // b = ', floor_division)
```

```
print('a ** b = ', exponential)

# Declaring values and organizing them together
num_one = 3
num_two = 4

# Arithmetic operations
total = num_one + num_two
diff = num_two - num_one
product = num_one * num_two
div = num_two / num_two
remainder = num_two % num_one

# Printing values with label
print('total: ', total)
print('difference: ', diff)
print('product: ', product)
print('division: ', div)
print('remainder: ', remainder)

# Calculating area of a circle
radius = 10                         # radius of a circle
area_of_circle = 3.14 * radius ** 2        # two * sign means exponent or power
print('Area of a circle:', area_of_circle)

# Calculating area of a rectangle
length = 10
width = 20
area_of_rectangle = length * width
print('Area of rectangle:', area_of_rectangle)

# Calculating a weight of an object
mass = 75
gravity = 9.81
weight = mass * gravity
print(weight, 'N')

print(3 > 2)     # True, because 3 is greater than 2
print(3 >= 2)    # True, because 3 is greater than 2
print(3 < 2)     # False,  because 3 is greater than 2
print(2 < 3)     # True, because 2 is less than 3
print(2 <= 3)    # True, because 2 is less than 3
print(3 == 2)    # False, because 3 is not equal to 2
print(3 != 2)    # True, because 3 is not equal to 2
print(len('mango') == len('avocado'))  # False
print(len('mango') != len('avocado'))  # True
print(len('mango') < len('avocado'))   # True
```

```
print(len('milk') != len('meat'))      # False
print(len('milk') == len('meat'))      # True
print(len('tomato') == len('potato'))  # True
print(len('python') > len('dragon'))   # False

# Boolean comparison
print('True == True: ', True == True)
print('True == False: ', True == False)
print('False == False:', False == False)
print('True and True: ', True and True)
print('True or False:', True or False)

# Another way comparison
print('1 is 1', 1 is 1)                 # True - because the data values are the same
print('1 is not 2', 1 is not 2)          # True - because 1 is not 2
print('A in Asabeneh', 'A' in 'Asabeneh') # True - A found in the string
print('B in Asabeneh', 'B' in 'Asabeneh') # False -there is no uppercase B
print('coding' in 'coding for all') # True - because coding for all has the word coding
print('a in an:', 'a' in 'an')      # True
print('4 is 2 ** 2:', 4 is 2 ** 2)   # True

print(3 > 2 and 4 > 3) # True - because both statements are true
print(3 > 2 and 4 < 3) # False - because the second statement is false
print(3 < 2 and 4 < 3) # False - because both statements are false
print(3 > 2 or 4 > 3)  # True - because both statements are true
print(3 > 2 or 4 < 3)  # True - because one of the statement is true
print(3 < 2 or 4 < 3)  # False - because both statements are false
print(not 3 > 2)     # False - because 3 > 2 is true, then not True gives False
print(not True)      # False - Negation, the not operator turns true to false
print(not False)     # True
print(not not True)  # True
print(not not False) # False
```

Output

```
Addition:  3
Subtraction:  1
Multiplication:  6
Division:  2.0
Division:  3.0
Division:  3.5
Division without the remainder:  3
Modulus:  1
Division without the remainder:  2
Exponential:  9
Floating Number,PI 3.14
Floating Number, gravity 9.81
```

Complex number: (1+1j)
Multiplying complex number: (2+0j)
5
a + b = 5
a - b = 1
a * b = 6
a / b = 1.5
a % b = 1
a // b = 1
a ** b = 9
total: 7
difference: 1
product: 12
division: 1.0
remainder: 1
Area of a circle: 314.0
Area of rectangle: 200
735.75 N
True
True
False
True
True
False
True
False
True
True
False
True
True
False
True == True: True
True == False: False
False == False: True
True and True: True
True or False: True
1 is 1 True
1 is not 2 True
A in Asabeneh True
B in Asabeneh False
True
a in an: True
4 is 2 ** 2: True
True
False
False
True
True

```
False
False
False
True
True
False
<>:103: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:104: SyntaxWarning: "is not" with a literal. Did you mean "!="?
<>:109: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:103: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:104: SyntaxWarning: "is not" with a literal. Did you mean "!="?
<>:109: SyntaxWarning: "is" with a literal. Did you mean "=="?
<ipython-input-5-db6733de76d2>:103: SyntaxWarning: "is" with a literal. Did you mean "=="?
  print('1 is 1', 1 is 1)                # True - because the data values are the same
<ipython-input-5-db6733de76d2>:104: SyntaxWarning: "is not" with a literal. Did you mean "!="?
  print('1 is not 2', 1 is not 2)          # True - because 1 is not 2
<ipython-input-5-db6733de76d2>:109: SyntaxWarning: "is" with a literal. Did you mean "=="?
  print('4 is 2 ** 2:', 4 is 2 ** 2)   # True
```

Note: This is a new warning added in Python 3.8. From the release notes: The compiler now produces a SyntaxWarning when identity checks (is and is not) are used with certain types of literals (e.g. strings, numbers).

4. Write a python program to work with conditionals

```
#If
a = 3
if a > 0:
    print('A is a positive number')

#If Else

a = 3
if a < 0:
    print('A is a negative number')
else:
    print('A is a positive number')

#If Elif Else

a = 0
if a > 0:
    print('A is a positive number')
elif a < 0:
    print('A is a negative number')
else:
    print('A is zero')

#Short Hand If

a = 3
print('A is positive') if a > 0 else print('A is negative') # first condition met, 'A is positive' will be printed

#Nested Conditions

a = 0
if a > 0:
    if a % 2 == 0:
        print('A is a positive and even integer')
    else:
        print('A is a positive number')
elif a == 0:
    print('A is zero')
else:
    print('A is a negative number')

#If Condition and Logical Operators

a = 0
if a > 0 and a % 2 == 0:
```

```
        print('A is an even and positive integer')
elif a > 0 and a % 2 !=  0:
     print('A is a positive integer')
elif a == 0:
    print('A is zero')
else:
    print('A is negative')

#If and Or Logical Operators

user = 'James'
access_level = 3
if user == 'admin' or access_level >= 4:
        print('Access granted!')
else:
    print('Access denied!')
```

Output

```
A is a positive number
A is a positive number
A is zero
A is positive
A is zero
A is zero
Access denied!
```

5. Write a python program to demonstrate Loops

```
#While Loop

count = 0
print('\nWhile Loop')
while count < 5:
    print(count,end=' ')
    count = count + 1

#While Loop with Else

count = 0
print('\nWhile Loop with Else')
while count < 5:
    print(count,end=' ')
    count = count + 1
else:
    print(count)

#For Loop
#For loop – Numbers

print('\nFor loop – Numbers')
numbers = [0, 1, 2, 3, 4, 5]
for number in numbers: # number is temporary name to refer to the list's items, valid only inside this loop
    print(number,end=' ')       # the numbers will be printed line by line, from 0 to 5
language = 'Python'

#For loop – String

print('\nFor loop – String')
for letter in language:
    print(letter,end=' ')

#For loop – Range

print('\nFor loop – Range')
for i in range(len(language)):
    print(language[i],end=' ')

#For loop – Tuple

numbers = (0, 1, 2, 3, 4, 5)
print('\nFor loop – Tuple')
for number in numbers:
```

```
    print(number,end=' ')

#For loop – Dictionary

person = {
    'first_name':'Asabeneh',
    'last_name':'Yetayeh',
    'age':50,
    'country':'Finland',
    'is_marred':True,
    'skills':['JavaScript', 'React', 'Node', 'MongoDB', 'Python'],
    'address':{
        'street':'Space street',
        'zipcode':'02210'
    }
}
print('\nFor loop – Dictionary(Key)')
for key in person:
    print(key,end=' ')
print('\nFor loop – Dictionary(Key,Value)')
for key, value in person.items():
    print(key, value) # this way we get both keys and values printed out

#For loop – Set

it_companies = {'Facebook', 'Google', 'Microsoft', 'Apple', 'IBM', 'Oracle', 'Amazon'}
print('\nFor loop – Set')
for company in it_companies:
    print(company,end=' ')

#Break – While Loop

count = 0
print('\nBreak – While Loop')
while count < 5:
    print(count,end=' ')
    count = count + 1
    if count == 3:
        break
print("After break while loop")

#Continue – While Loop

count = 0
print('\nContinue – While Loop')
while count < 5:
    count = count + 1
```

```
    if count == 3:
        continue
    print(count,end=' ')

#Break – For Loop

numbers = (0,1,2,3,4,5)
print('\nBreak – For Loop')
for number in numbers:
    print(number,end=' ')
    if number == 3:
        break
print("After break for loop")

#Continue – For Loop

numbers = (0,1,2,3,4,5)
print('\nContinue – For Loop')
for number in numbers:
    print(number,end=' ')
    number += 1
    if number == 3:
        continue

#print('Next number should be ', number + 1) if number != 5 else print("loop's end")
#The Range Function

lst = list(range(11))
print('\nThe Range Function-range(11) for list')
print(lst) # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
st = set(range(1, 11))    # 2 arguments indicate start and end of the sequence, step set to default 1
print('\nThe Range Function-range(1,11) for set')
print(st,end=' ') # {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

lst = list(range(0,11,2))
print('\nThe Range Function-range(0,11,2) for list')
print(lst,end=' ') # [0, 2, 4, 6, 8, 10]
st = set(range(0,11,2))
print('\nThe Range Function-range(0,11,2) for set')
print(st,end=' ') #  {0, 2, 4, 6, 8, 10}
# syntax - for iterator in range(start, end, step):
print('\nThe Range Function-range(11) with for loop')
for number in range(11):
    print(number,end=' ')   # prints 0 to 10, not including 11
#Nested For Loop- We can write loops inside a loop. Example:

person = {
```

```
    'first_name': 'Asabeneh',
    'last_name': 'Yetayeh',
    'age': 50,
    'country': 'Finland',
    'is_marred': True,
    'skills': ['JavaScript', 'React', 'Node', 'MongoDB', 'Python'],
    'address': {
        'street': 'Space street',
        'zipcode': '02210'
    }
}
print('\nThe Nested For loop')
for key in person:
    if key == 'skills':
        for skill in person['skills']:
            print(skill,end=' ')

#For Else- If we want to execute some message when the loop ends, we use else. Example:

print('\nThe For loop with else')
for number in range(11):
    print(number,end=' ')   # prints 0 to 10, not including 11
else:
    print('The loop stops at', number)

#Pass-
 In python when statement is required (after semicolon), but we don't like to execute any code there, we c
an write the word pass to avoid errors.
#Also we can use it as a placeholder, for future statements. Example:

print('\nFor and Pass')
for number in range(6):
    pass
```

Output

```
While Loop
0 1 2 3 4
While Loop with Else
0 1 2 3 4 5

For loop – Numbers
0 1 2 3 4 5
For loop – String
P y t h o n
For loop – Range
P y t h o n
```

For loop – Tuple
0 1 2 3 4 5
For loop – Dictionary(Key)
first_name last_name age country is_marred skills address
For loop – Dictionary(Key,Value)
first_name Asabeneh
last_name Yetayeh
age 50
country Finland
is_marred True
skills ['JavaScript', 'React', 'Node', 'MongoDB', 'Python']
address {'street': 'Space street', 'zipcode': '02210'}

For loop – Set
Amazon Facebook Oracle Microsoft Apple IBM Google
Break – While Loop
0 1 2 After break while loop

Continue – While Loop
1 2 4 5
Break – For Loop
0 1 2 3 After break for loop

Continue – For Loop
0 1 2 3 4 5
The Range Function-range(11) for list
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

The Range Function-range(1,11) for set
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
The Range Function-range(0,11,2) for list
[0, 2, 4, 6, 8, 10]
The Range Function-range(0,11,2) for set
{0, 2, 4, 6, 8, 10}
The Range Function-range(11) with for loop
0 1 2 3 4 5 6 7 8 9 10
The Nested For loop
JavaScript React Node MongoDB Python
The For loop with else
0 1 2 3 4 5 6 7 8 9 10 The loop stops at 10

For and Pass

6. Write a python program to check whether a given number is Armstrong number or not

```
# Python program to check if the number is an Armstrong number or not

# take input from the user
num = int(input("Enter a number: "))

# initialize sum
sum = 0

# find the sum of the cube of each digit
temp = num
while temp > 0:
   digit = temp % 10
   sum += digit ** 3
   temp //= 10

# display the result
if num == sum:
   print(num,"is an Armstrong number")
else:
   print(num,"is not an Armstrong number")
```

Ouptut

Enter a number: 456

456 is not an Armstrong number

Enter a number: 153

153 an Armstrong number

7. Write a python program to generate prime numbers between different intervals

```
# Python program to display all the prime numbers within an interval

lower = 900
upper = 1000

print("Prime numbers between", lower, "and", upper, "are:")

for num in range(lower, upper + 1):
   # all prime numbers are greater than 1
   if num > 1:
      for i in range(2, num):
         if (num % i) == 0:
            break
      else:
         print(num)
```

Output

```
Prime numbers between 900 and 1000 are:
907
911
919
929
937
941
947
953
967
971
977
983
991
997
```

8. Write a python program to create, append and remove list

```
from builtins import list
#Lists
empty_list = list() # this is an empty list, no item in the list
print(len(empty_list)) # 0

fruits = ['banana', 'orange', 'mango', 'lemon']                    # list of fruits
vegetables = ['Tomato', 'Potato', 'Cabbage','Onion', 'Carrot']     # list of vegetables
animal_products = ['milk', 'meat', 'butter', 'yoghurt']            # list of animal products
web_techs = ['HTML', 'CSS', 'JS', 'React','Redux', 'Node', 'MongDB'] # list of web technologies
countries = ['Finland', 'Estonia', 'Denmark', 'Sweden', 'Norway']

# Print the lists and it length
print('Fruits:', fruits)
print('Number of fruits:', len(fruits))
print('Vegetables:', vegetables)
print('Number of vegetables:', len(vegetables))
print('Animal products:',animal_products)
print('Number of animal products:', len(animal_products))
print('Web technologies:', web_techs)
print('Number of web technologies:', len(web_techs))
print('Number of countries:', len(countries))

# Modifying list

fruits = ['banana', 'orange', 'mango', 'lemon']
first_fruit = fruits[0] # we are accessing the first item using its index
print(first_fruit)      # banana
second_fruit = fruits[1]
print(second_fruit)     # orange
last_fruit = fruits[3]
print(last_fruit) # lemon
# Last index
last_index = len(fruits) - 1
last_fruit = fruits[last_index]

# Accessing items
fruits = ['banana', 'orange', 'mango', 'lemon']
last_fruit = fruits[-1]
second_last = fruits[-2]
print(last_fruit)       # lemon
```

```
print(second_last)      # mango

# Slicing items
fruits = ['banana', 'orange', 'mango', 'lemon']
all_fruits = fruits[0:4] # it returns all the fruits
# this is also give the same result as the above
all_fruits = fruits[0:] # if we don't set where to stop it takes all the rest
orange_and_mango = fruits[1:3] # it does not include the end index
orange_mango_lemon = fruits[1:]

fruits = ['banana', 'orange', 'mango', 'lemon']
all_fruits = fruits[-4:] # it returns all the fruits
# this is also give the same result as the above
orange_and_mango = fruits[-3:-1] # it does not include the end index
orange_mango_lemon = fruits[-3:]


fruits = ['banana', 'orange', 'mango', 'lemon']
fruits[0] = 'Avocado'
print(fruits)       #  ['avocado', 'orange', 'mango', 'lemon']
fruits[1] = 'apple'
print(fruits)       #  ['avocado', 'apple', 'mango', 'lemon']
last_index = len(fruits) - 1
fruits[last_index] = 'lime'
print(fruits)       #  ['avocado', 'apple', 'mango', 'lime']

# checking items
fruits = ['banana', 'orange', 'mango', 'lemon']
does_exist = 'banana' in fruits
print(does_exist)  # True
does_exist = 'lime' in fruits
print(does_exist)  # False

# Append
fruits = ['banana', 'orange', 'mango', 'lemon']
fruits.append('apple')
print(fruits)          # ['banana', 'orange', 'mango', 'lemon', 'apple']
fruits.append('lime')   # ['banana', 'orange', 'mango', 'lemon', 'apple', 'lime]
print(fruits)

# insert
fruits = ['banana', 'orange', 'mango', 'lemon']
fruits.insert(2, 'apple') # insert apple between orange and mango
```

```
print(fruits)          # ['banana', 'orange', 'apple', 'mango', 'lemon']
fruits.insert(3, 'lime')   # ['banana', 'orange', 'apple', 'lime','mango',  'lemon',]
print(fruits)

# remove
fruits = ['banana', 'orange', 'mango', 'lemon']
fruits.remove('banana')
print(fruits)  # ['orange', 'mango', 'lemon']
fruits.remove('lemon')
print(fruits)  # ['orange', 'mango']

# pop
fruits = ['banana', 'orange', 'mango', 'lemon']
fruits.pop()
print(fruits)      # ['banana', 'orange', 'mango']

fruits.pop(0)
print(fruits)      # ['orange', 'mango']

# del
fruits = ['banana', 'orange', 'mango', 'lemon']
del fruits[0]
print(fruits)      # ['orange', 'mango', 'lemon']

del fruits[1]
print(fruits)      # ['orange', 'lemon']
del fruits
#print(fruits)      # This should give: NameError: name 'fruits' is not defined

# clear
fruits = ['banana', 'orange', 'mango', 'lemon']
fruits.clear()
print(fruits)      # []

# copying a lits

fruits = ['banana', 'orange', 'mango', 'lemon']
fruits_copy = fruits.copy()
print(fruits_copy)      # ['banana', 'orange', 'mango', 'lemon']

# join
positive_numbers = [1, 2, 3,4,5]
```

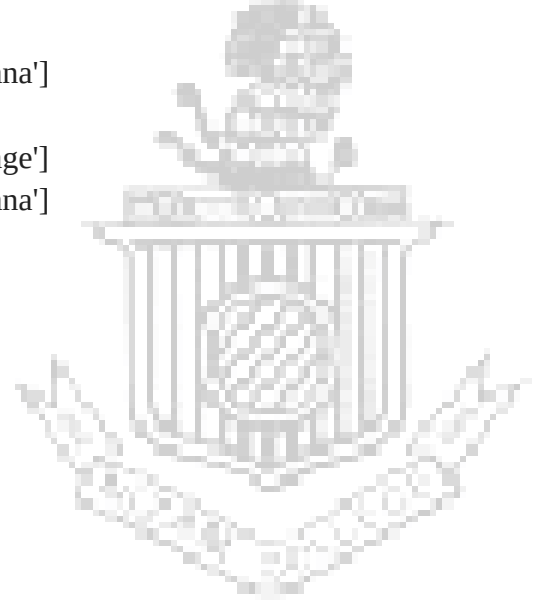
```
zero = [0]
negative_numbers = [-5,-4,-3,-2,-1]
integers = negative_numbers + zero + positive_numbers
print(integers)
fruits = ['banana', 'orange', 'mango', 'lemon']
vegetables = ['Tomato', 'Potato', 'Cabbage','Onion', 'Carrot']
fruits_and_vegetables = fruits + vegetables
print(fruits_and_vegetables )

# join with extend
num1 = [0, 1, 2, 3]
num2= [4, 5,6]
num1.extend(num2)
print('Numbers:', num1)
negative_numbers = [-5,-4,-3,-2,-1]
positive_numbers = [1, 2, 3,4,5]
zero = [0]

negative_numbers.extend(zero)
negative_numbers.extend(positive_numbers)
print('Integers:', negative_numbers)
fruits = ['banana', 'orange', 'mango', 'lemon']
vegetables = ['Tomato', 'Potato', 'Cabbage','Onion', 'Carrot']
fruits.extend(vegetables)
print('Fruits and vegetables:', fruits )

# count
fruits = ['banana', 'orange', 'mango', 'lemon']
print(fruits.count('orange'))   # 1
ages = [22, 19, 24, 25, 26, 24, 25, 24]
print(ages.count(24))          # 3

# index
fruits = ['banana', 'orange', 'mango', 'lemon']
print(fruits.index('orange'))   # 1
ages = [22, 19, 24, 25, 26, 24, 25, 24]
print(ages.index(24))
# Reverse
fruits = ['banana', 'orange', 'mango', 'lemon']
fruits.reverse()
print(fruits)
ages = [22, 19, 24, 25, 26, 24, 25, 24]
```

```
ages.reverse()
print(ages)

# sort
fruits = ['banana', 'orange', 'mango', 'lemon']
fruits.sort()
print(fruits)
fruits.sort(reverse=True)
print(fruits)
ages = [22, 19, 24, 25, 26, 24, 25, 24]
ages.sort()
print(ages)
ages.sort(reverse=True)
print(ages)
```

Output

```
0
Fruits: ['banana', 'orange', 'mango', 'lemon']
Number of fruits: 4
Vegetables: ['Tomato', 'Potato', 'Cabbage', 'Onion', 'Carrot']
Number of vegetables: 5
Animal products: ['milk', 'meat', 'butter', 'yoghurt']
Number of animal products: 4
Web technologies: ['HTML', 'CSS', 'JS', 'React', 'Redux', 'Node', 'MongDB']
Number of web technologies: 7
Number of countries: 5
banana
orange
lemon
lemon
mango
['Avocado', 'orange', 'mango', 'lemon']
['Avocado', 'apple', 'mango', 'lemon']
['Avocado', 'apple', 'mango', 'lime']
True
False
['banana', 'orange', 'mango', 'lemon', 'apple']
['banana', 'orange', 'mango', 'lemon', 'apple', 'lime']
['banana', 'orange', 'apple', 'mango', 'lemon']
['banana', 'orange', 'apple', 'lime', 'mango', 'lemon']
['orange', 'mango', 'lemon']
```

['orange', 'mango']
['banana', 'orange', 'mango']
['orange', 'mango']
['orange', 'mango', 'lemon']
['orange', 'lemon']
[]
['banana', 'orange', 'mango', 'lemon']
[-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5]
['banana', 'orange', 'mango', 'lemon', 'Tomato', 'Potato', 'Cabbage', 'Onion', 'Carrot']
Numbers: [0, 1, 2, 3, 4, 5, 6]
Integers: [-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5]
Fruits and vegetables: ['banana', 'orange', 'mango', 'lemon', 'Tomato', 'Potato', 'Cabbage', 'Onion', 'Carrot']
1
3
1
2
['lemon', 'mango', 'orange', 'banana']
[24, 25, 24, 26, 25, 24, 19, 22]
['banana', 'lemon', 'mango', 'orange']
['orange', 'mango', 'lemon', 'banana']
[19, 22, 24, 24, 24, 25, 25, 26]
[26, 25, 25, 24, 24, 24, 22, 19]



9. Write a program to demonstrate working with tuples in python

```
from builtins import tuple
#tuple
# syntax
empty_tuple = ()
# or using the tuple constructor
empty_tuple = tuple()
# syntax
tpl = ('item1', 'item2','item3')
fruits = ('banana', 'orange', 'mango', 'lemon')

# tuple length
tpl = ('item1', 'item2', 'item3')
len(tpl)

# accessing tuple items
tpl = ('item1', 'item2', 'item3')
first_item = tpl[0]
second_item = tpl[1]
fruits = ('banana', 'orange', 'mango', 'lemon')
first_fruit = fruits[0]
second_fruit = fruits[1]
last_index =len(fruits) - 1
last_fruit = fruits[last_index]

# Negative Indexing
tpl = ('item1', 'item2', 'item3','item4')
first_item = tpl[-4]
second_item = tpl[-3]
fruits = ('banana', 'orange', 'mango', 'lemon')
first_fruit = fruits[-4]
second_fruit = fruits[-3]
last_fruit = fruits[-1]
#Slicing tuples
tpl = ('item1', 'item2', 'item3','item4')
all_items = tpl[0:4]        # all items
all_items = tpl[0:]        # all items
middle_two_items = tpl[1:3]  # does not include item at index 3
fruits = ('banana', 'orange', 'mango', 'lemon')
all_fruits = fruits[0:4]    # all items
```

```
all_fruits= fruits[0:]      # all items
orange_mango = fruits[1:3]  # doesn't include item at index 3
orange_to_the_rest = fruits[1:]
#Range of Negative Indexes
tpl = ('item1', 'item2', 'item3','item4')
all_items = tpl[-4:]        # all items
middle_two_items = tpl[-3:-1]  # does not include item at index 3 (-1)
fruits = ('banana', 'orange', 'mango', 'lemon')
all_fruits = fruits[-4:]    # all items
orange_mango = fruits[-3:-1]  # doesn't include item at index 3
orange_to_the_rest = fruits[-3:]
#Changing Tuples to Lists
tpl = ('item1', 'item2', 'item3','item4')
lst = list(tpl)
fruits = ('banana', 'orange', 'mango', 'lemon')
fruits = list(fruits)
fruits[0] = 'apple'
print(fruits)     # ['apple', 'orange', 'mango', 'lemon']
fruits = tuple(fruits)
print(fruits)     # ('apple', 'orange', 'mango', 'lemon')
#Checking an Item in a Tuple
tpl = ('item1', 'item2', 'item3','item4')
'item2' in tpl # True
fruits = ('banana', 'orange', 'mango', 'lemon')
print('orange' in fruits) # True
print('apple' in fruits) # False
#fruits[0] = 'apple' # TypeError: 'tuple' object does not support item assignment
#Joining Tuples
tpl1 = ('item1', 'item2', 'item3')
tpl2 = ('item4', 'item5','item6')
tpl3 = tpl1 + tpl2
fruits = ('banana', 'orange', 'mango', 'lemon')
vegetables = ('Tomato', 'Potato', 'Cabbage','Onion', 'Carrot')
fruits_and_vegetables = fruits + vegetables
#Deleting Tuples
tpl1 = ('item1', 'item2', 'item3')
del tpl1
fruits = ('banana', 'orange', 'mango', 'lemon')
del fruits
```

Output

['apple', 'orange', 'mango', 'lemon']
('apple', 'orange', 'mango', 'lemon')
True
False

10. Write a program to demonstrate dictionaries in python

```
#Creating a Dictionary
empty_dict = {}
# Dictionary with data values
dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
person = {
    'first_name':'Asabeneh',
    'last_name':'Yetayeh',
    'age':250,
    'country':'Finland',
    'is_marred':True,
    'skills':['JavaScript', 'React', 'Node', 'MongoDB', 'Python'],
    'address':{
        'street':'Space street',
        'zipcode':'02210'
    }
    }
#Dictionary Length
dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
print(len(dct)) # 4
person = {
    'first_name':'Asabeneh',
    'last_name':'Yetayeh',
    'age':250,
    'country':'Finland',
    'is_marred':True,
    'skills':['JavaScript', 'React', 'Node', 'MongoDB', 'Python'],
    'address':{
        'street':'Space street',
        'zipcode':'02210'
    }
    }
print(len(person)) # 7
#Accessing Dictionary Items
dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
print(dct['key1']) # value1
print(dct['key4']) # value4
person = {
    'first_name':'Asabeneh',
    'last_name':'Yetayeh',
    'age':250,
```

```
    'country':'Finland',
    'is_marred':True,
    'skills':['JavaScript', 'React', 'Node', 'MongoDB', 'Python'],
    'address':{
        'street':'Space street',
        'zipcode':'02210'
    }
    }
print(person['first_name']) # Asabeneh
print(person['country'])    # Finland
print(person['skills'])     # ['JavaScript', 'React', 'Node', 'MongoDB', 'Python']
print(person['skills'][0])  # JavaScript
print(person['address']['street']) # Space street
#print(person['city'])       # Error
#Accessing an item by key name raises an error if the key does not exist. To avoid this error first we have
 to check if a key exist or we can use the get method. The get method returns None, which is a NoneType
object data type, if the key does not exist.
person = {
    'first_name':'Asabeneh',
    'last_name':'Yetayeh',
    'age':250,
    'country':'Finland',
    'is_marred':True,
    'skills':['JavaScript', 'React', 'Node', 'MongoDB', 'Python'],
    'address':{
        'street':'Space street',
        'zipcode':'02210'
    }
    }
print(person.get('first_name')) # Asabeneh
print(person.get('country'))    # Finland
print(person.get('skills')) #['HTML','CSS','JavaScript', 'React', 'Node', 'MongoDB', 'Python']
print(person.get('city'))   # None
#Adding Items to a Dictionary
dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
dct['key5'] = 'value5'
person = {
    'first_name':'Asabeneh',
    'last_name':'Yetayeh',
    'age':250,
    'country':'Finland',
    'is_marred':True,
```

```
    'skills':['JavaScript', 'React', 'Node', 'MongoDB', 'Python'],
    'address':{
        'street':'Space street',
        'zipcode':'02210'
        }
}
person['job_title'] = 'Instructor'
person['skills'].append('HTML')
print(person)
#Modifying Items in a Dictionary
dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
dct['key1'] = 'value-one'
person = {
    'first_name':'Asabeneh',
    'last_name':'Yetayeh',
    'age':250,
    'country':'Finland',
    'is_marred':True,
    'skills':['JavaScript', 'React', 'Node', 'MongoDB', 'Python'],
    'address':{
        'street':'Space street',
        'zipcode':'02210'
    }
    }
person['first_name'] = 'Eyob'
person['age'] = 252
#Checking Keys in a Dictionary
dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
print('key2' in dct) # True
print('key5' in dct) # False
#Removing Key and Value Pairs from a Dictionary
dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
dct.pop('key1') # removes key1 item
dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
dct.popitem() # removes the last item
del dct['key2'] # removes key2 item
person = {
    'first_name':'Asabeneh',
    'last_name':'Yetayeh',
    'age':250,
    'country':'Finland',
    'is_marred':True,
```

```
    'skills':['JavaScript', 'React', 'Node', 'MongoDB', 'Python'],
    'address':{
        'street':'Space street',
        'zipcode':'02210'
    }
    }
person.pop('first_name')       # Removes the firstname item
person.popitem()               # Removes the address item
#del person['is_married']       # Removes the is_married item
#Changing Dictionary to a List of Items
dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
print(dct.items()) # dict_items([('key1', 'value1'), ('key2', 'value2'), ('key3', 'value3'), ('key4', 'value4')])
#Clearing a Dictionary
dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
print(dct.clear()) # None
#Deleting a Dictionary
dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
del dct
#Copy a Dictionary
dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
dct_copy = dct.copy() # {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
#Getting Dictionary Keys as a List
dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
keys = dct.keys()
print(keys)     # dict_keys(['key1', 'key2', 'key3', 'key4'])
#Getting Dictionary Values as a List
dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
values = dct.values()
print(values)
```

Output

```
4
7
value1
value4
Asabeneh
Finland
['JavaScript', 'React', 'Node', 'MongoDB', 'Python']
JavaScript
Space street
Asabeneh
```

Finland
['JavaScript', 'React', 'Node', 'MongoDB', 'Python']
None
{'first_name': 'Asabeneh', 'last_name': 'Yetayeh', 'age': 250, 'country': 'Finland', 'is_marred': True, 'skills': ['JavaScript', 'React', 'Node', 'MongoDB', 'Python', 'HTML'], 'address': {'street': 'Space street', 'zipcode': '02210'}, 'job_title': 'Instructor'}
True
False
dict_items([('key1', 'value1'), ('key2', 'value2'), ('key3', 'value3'), ('key4', 'value4')])
None
dict_keys(['key1', 'key2', 'key3', 'key4'])
dict_values(['value1', 'value2', 'value3', 'value4'])

11. Write a python program on string operations

```
# Single line comment
letter = 'P'              # A string could be a single character or a bunch of texts
print(letter)             # P
print(len(letter))        # 1
greeting = 'Hello, World!'  # String could be  a single or double quote,"Hello, World!"
print(greeting)           # Hello, World!
print(len(greeting))      # 13
sentence = "I hope you are enjoying 30 days of python challenge"
print(sentence)

# Multiline String
multiline_string = "'I am a teacher and enjoy teaching.
I didn't find anything as rewarding as empowering people.
That is why I created 30 days of python.'"
print(multiline_string)
# Another way of doing the same thing
multiline_string = """I am a teacher and enjoy teaching.
I didn't find anything as rewarding as empowering people.
That is why I created 30 days of python."""
print(multiline_string)

# String Concatenation
first_name = 'Asabeneh'
last_name = 'Yetayeh'
space = ' '
full_name = first_name  +  space + last_name
print(full_name) # Asabeneh Yetayeh
# Checking length of a string using len() builtin function
print(len(first_name))  # 8
print(len(last_name))   # 7
print(len(first_name) > len(last_name)) # True
print(len(full_name)) # 15

#### Unpacking characters
language = 'Python'
a,b,c,d,e,f = language # unpacking sequence characters into variables
print(a) # P
print(b) # y
print(c) # t
print(d) # h
print(e) # o
print(f) # n

# Accessing characters in strings by index
language = 'Python'
```

```
first_letter = language[0]
print(first_letter) # P
second_letter = language[1]
print(second_letter) # y
last_index = len(language) - 1
last_letter = language[last_index]
print(last_letter) # n

# If we want to start from right end we can use negative indexing. -1 is the last index
language = 'Python'
last_letter = language[-1]
print(last_letter) # n
second_last = language[-2]
print(second_last) # o

# Slicing

language = 'Python'
first_three = language[0:3] # starts at zero index and up to 3 but not include 3
last_three = language[3:6]
print(last_three) # hon
# Another way
last_three = language[-3:]
print(last_three)   # hon
last_three = language[3:]
print(last_three)   # hon

# Skipping character while splitting Python strings
language = 'Python'
pto = language[0:6:2] #
print(pto) # pto

# Escape sequence
print('I hope every one enjoying the python challenge.\nDo you ?') # line break
print('Days\tTopics\tExercises')
print('Day 1\t3\t5')
print('Day 2\t3\t5')
print('Day 3\t3\t5')
print('Day 4\t3\t5')
print('This is a back slash  symbol (\\)') # To write a back slash
print('In every programming language it starts with \"Hello, World!\"')

## String Methods
# capitalize(): Converts the first character the string to Capital Letter

challenge = 'thirty days of python'
print(challenge.capitalize()) # 'Thirty days of python'
```

```
# count(): returns occurrences of substring in string, count(substring, start=.., end=..)

challenge = 'thirty days of python'
print(challenge.count('y')) # 3
print(challenge.count('y', 7, 14)) # 1
print(challenge.count('th')) # 2`

# endswith(): Checks if a string ends with a specified ending

challenge = 'thirty days of python'
print(challenge.endswith('on'))   # True
print(challenge.endswith('tion')) # False

# expandtabs(): Replaces tab character with spaces, default tab size is 8. It takes tab size argument

challenge = 'thirty\tdays\tof\tpython'
print(challenge.expandtabs())   # 'thirty  days    of      python'
print(challenge.expandtabs(10)) # 'thirty    days      of        python'

# find(): Returns the index of first occurrence of substring

challenge = 'thirty days of python'
print(challenge.find('y'))  # 5
print(challenge.find('th')) # 0

# format()  formats string into nicer output
first_name = 'Asabeneh'
last_name = 'Yetayeh'
job = 'teacher'
country = 'Finland'
sentence = 'I am {} {}. I am a {}. I live in {}.'.format(first_name, last_name, job, country)
print(sentence) # I am Asabeneh Yetayeh. I am a teacher. I live in Finland.

radius = 10
pi = 3.14
area = pi # radius ## 2
result = 'The area of circle with {} is {}'.format(str(radius), str(area))
print(result) # The area of circle with 10 is 314.0

# index(): Returns the index of substring
challenge = 'thirty days of python'
print(challenge.find('y'))  # 5
print(challenge.find('th')) # 0

# isalnum(): Checks alphanumeric character
challenge = 'ThirtyDaysPython'
```

```
print(challenge.isalnum()) # True
challenge = '30DaysPython'
print(challenge.isalnum()) # True
challenge = 'thirty days of python'
print(challenge.isalnum()) # False
challenge = 'thirty days of python 2019'
print(challenge.isalnum()) # False

# isalpha(): Checks if all characters are alphabets
challenge = 'thirty days of python'
print(challenge.isalpha()) # True
num = '123'
print(num.isalpha())      # False

# isdecimal(): Checks Decimal Characters
challenge = 'thirty days of python'
print(challenge.find('y'))  # 5
print(challenge.find('th')) # 0

# isdigit(): Checks Digit Characters
challenge = 'Thirty'
print(challenge.isdigit()) # False
challenge = '30'
print(challenge.isdigit())   # True

# isdecimal():Checks decimal characters
num = '10'
print(num.isdecimal()) # True
num = '10.5'
print(num.isdecimal()) # False

# isidentifier():Checks for valid identifier means it check if a string is a valid variable name
challenge = '30DaysOfPython'
print(challenge.isidentifier()) # False, because it starts with a number
challenge = 'thirty_days_of_python'
print(challenge.isidentifier()) # True

# islower():Checks if all alphabets in a string are lowercase
challenge = 'thirty days of python'
print(challenge.islower()) # True
challenge = 'Thirty days of python'
print(challenge.islower()) # False

# isupper(): returns if all characters are uppercase characters
challenge = 'thirty days of python'
print(challenge.isupper()) #  False
challenge = 'THIRTY DAYS OF PYTHON'
```

```
print(challenge.isupper()) # True

# isnumeric():Checks numeric characters
num = '10'
print(num.isnumeric())      # True
print('ten'.isnumeric())    # False

# join(): Returns a concatenated string
web_tech = ['HTML', 'CSS', 'JavaScript', 'React']
result = '#, '.join(web_tech)
print(result) # 'HTML# CSS# JavaScript# React'

# strip(): Removes both leading and trailing characters
challenge = ' thirty days of python '
print(challenge.strip('y')) # 5

# replace(): Replaces substring inside
challenge = 'thirty days of python'
print(challenge.replace('python', 'coding')) # 'thirty days of coding'

# split():Splits String from Left
challenge = 'thirty days of python'
print(challenge.split()) # ['thirty', 'days', 'of', 'python']

# title(): Returns a Title Cased String
challenge = 'thirty days of python'
print(challenge.title()) # Thirty Days Of Python

# swapcase(): Checks if String Starts with the Specified String
challenge = 'thirty days of python'
print(challenge.swapcase())   # THIRTY DAYS OF PYTHON
challenge = 'Thirty Days Of Python'
print(challenge.swapcase())  # tHIRTY dAYS oF pYTHON

# startswith(): Checks if String Starts with the Specified String
challenge = 'thirty days of python'
print(challenge.startswith('thirty')) # True
challenge = '30 days of python'
print(challenge.startswith('thirty')) # False
```

Output

```
P
1
Hello, World!
13
I hope you are enjoying 30 days of python challenge
```

I am a teacher and enjoy teaching.
I didn't find anything as rewarding as empowering people.
That is why I created 30 days of python.
I am a teacher and enjoy teaching.
I didn't find anything as rewarding as empowering people.
That is why I created 30 days of python.
Asabeneh Yetayeh
8
7
True
16
P
y
t
h
o
n
P
y
n
n
o
hon
hon
hon
Pto
I hope every one enjoying the python challenge.
Do you ?
Days Topics Exercises
Day 1 3 5
Day 2 3 5
Day 3 3 5
Day 4 3 5
This is a back slash symbol (\)
In every programming language it starts with "Hello, World!"
Thirty days of python
3
1
2
True
False
thirty days of python
thirty days of python
5
0
I am Asabeneh Yetayeh. I am a teacher. I live in Finland.
The area of circle with 10 is 3.14
5
0

True
True
False
False
False
False
5
0
False
True
True
False
False
True
True
False
False
True
True
False
HTML#, CSS#, JavaScript#, React
 thirty days of python
thirty days of coding
['thirty', 'days', 'of', 'python']
Thirty Days Of Python
THIRTY DAYS OF PYTHON
tHIRTY dAYS oF pYTHON
True
False

12. Write a python program to demonstrate Set data type

```
# Creating an empty set-We use curly brackets, {} to create a set or the set() built-in function.

st = {}
# or
st = set()

# Creating a set with initial items

st = {'item1', 'item2', 'item3', 'item4'}
#Example:
fruits = {'banana', 'orange', 'mango', 'lemon'}

#Getting Set's Length-We use len() method to find the length of a set.

st = {'item1', 'item2', 'item3', 'item4'}
len(st)
#Example:
fruits = {'banana', 'orange', 'mango', 'lemon'}
len(fruits)

#Accessing Items in a Set- We use loops to access items.
#Checking an Item- To check if an item exist in a list we use in membership operator.

st = {'item1', 'item2', 'item3', 'item4'}
print("Does set st contain item3? ", 'item3' in st) # Does set st contain item3? True
#Example:
fruits = {'banana', 'orange', 'mango', 'lemon'}
print('mango' in fruits ) # True

#Adding Items to a Set-
Once a set is created we cannot change any items and we can also add additional items.
#Add one item using add()

st = {'item1', 'item2', 'item3', 'item4'}
st.add('item5')
#Example:
fruits = {'banana', 'orange', 'mango', 'lemon'}
fruits.add('lime')

#Add multiple items using update() The update() allows to add multiple items to a set. The update() takes
 a list argument.

st = {'item1', 'item2', 'item3', 'item4'}
st.update(['item5','item6','item7'])
#Example:
```

```
fruits = {'banana', 'orange', 'mango', 'lemon'}
vegetables = ('tomato', 'potato', 'cabbage','onion', 'carrot')
fruits.update(vegetables)

#Removing Items from a Set
#We can remove an item from a set using remove() method. If the item is not found remove() method wil
l raise errors, so it is good to check if the item exist in the given set. However, discard() method doesn't ra
ise any errors.

st = {'item1', 'item2', 'item3', 'item4'}
st.remove('item2')

#The pop() methods remove a random item from a set and it returns the removed item. Example:

fruits = {'banana', 'orange', 'mango', 'lemon'}
fruits.pop()  # removes a random item from the set
#If we are interested in the removed item.
fruits = {'banana', 'orange', 'mango', 'lemon'}
removed_item = fruits.pop()

#Clearing Items in a Set- If we want to clear or empty the set we use clear method.

st = {'item1', 'item2', 'item3', 'item4'}
st.clear()
#Example:
fruits = {'banana', 'orange', 'mango', 'lemon'}
fruits.clear()
print(fruits) # set()

#Deleting a Set- If we want to delete the set itself we use del operator.
st = {'item1', 'item2', 'item3', 'item4'}
del st
#Example:
fruits = {'banana', 'orange', 'mango', 'lemon'}
del fruits

#Converting List to Set-
 We can convert list to set and set to list. Converting list to set removes duplicates and only unique items
will be reserved.

lst = ['item1', 'item2', 'item3', 'item4', 'item1']
st = set(lst)  # {'item2', 'item4', 'item1', 'item3'} -
 the order is random, because sets in general are unordered
#Example:
fruits = ['banana', 'orange', 'mango', 'lemon','orange', 'banana']
fruits = set(fruits) # {'mango', 'lemon', 'banana', 'orange'}
```

```
#Joining Sets- We can join two sets using the union() or update() method.
#Union This method returns a new set
st1 = {'item1', 'item2', 'item3', 'item4'}
st2 = {'item5', 'item6', 'item7', 'item8'}
st3 = st1.union(st2)
#Example:
fruits = {'banana', 'orange', 'mango', 'lemon'}
vegetables = {'tomato', 'potato', 'cabbage','onion', 'carrot'}
print(fruits.union(vegetables)) # {'lemon', 'carrot', 'tomato', 'banana', 'mango', 'orange', 'cabbage', 'potato', '
onion'}

# Update This method inserts a set into a given set

st1 = {'item1', 'item2', 'item3', 'item4'}
st2 = {'item5', 'item6', 'item7', 'item8'}
st1.update(st2) # st2 contents are added to st1
#Example:
fruits = {'banana', 'orange', 'mango', 'lemon'}
vegetables = {'tomato', 'potato', 'cabbage','onion', 'carrot'}
fruits.update(vegetables)
print(fruits) # {'lemon', 'carrot', 'tomato', 'banana', 'mango', 'orange', 'cabbage', 'potato', 'onion'}

#Finding Intersection Items-Intersection returns a set of items which are in both the sets. See the example

st1 = {'item1', 'item2', 'item3', 'item4'}
st2 = {'item3', 'item2'}
st1.intersection(st2) # {'item3', 'item2'}
#Example:
whole_numbers = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
even_numbers = {0, 2, 4, 6, 8, 10}
whole_numbers.intersection(even_numbers) # {0, 2, 4, 6, 8, 10}

python = {'p', 'y', 't', 'h', 'o','n'}
dragon = {'d', 'r', 'a', 'g', 'o','n'}
python.intersection(dragon)     # {'o', 'n'}

#Checking Subset and Super Set- A set can be a subset or super set of other sets:
#Subset: issubset()
#Super set: issuperset

st1 = {'item1', 'item2', 'item3', 'item4'}
st2 = {'item2', 'item3'}
st2.issubset(st1) # True
st1.issuperset(st2) # True
#Example:
whole_numbers = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
even_numbers = {0, 2, 4, 6, 8, 10}
```

```
whole_numbers.issubset(even_numbers) # False, because it is a super set
whole_numbers.issuperset(even_numbers) # True

python = {'p', 'y', 't', 'h', 'o','n'}
dragon = {'d', 'r', 'a', 'g', 'o','n'}
python.issubset(dragon)     # False

#Checking the Difference Between Two Sets-It returns the difference between two sets.
st1 = {'item1', 'item2', 'item3', 'item4'}
st2 = {'item2', 'item3'}
st2.difference(st1) # set()
st1.difference(st2) # {'item1', 'item4'} => st1\st2
#Example:
whole_numbers = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
even_numbers = {0, 2, 4, 6, 8, 10}
whole_numbers.difference(even_numbers) # {1, 3, 5, 7, 9}

python = {'p', 'y', 't', 'o','n'}
dragon = {'d', 'r', 'a', 'g', 'o','n'}
python.difference(dragon)     # {'p', 'y', 't'}  - the result is unordered (characteristic of sets)
dragon.difference(python)     # {'d', 'r', 'a', 'g'}

#Finding Symmetric Difference Between Two Sets-
 It returns the the symmetric difference between two sets. It means that it returns a set that contains all ite
ms from both sets, except items that are present in both sets, mathematically: (A\B) ∪ (B\A)

st1 = {'item1', 'item2', 'item3', 'item4'}
st2 = {'item2', 'item3'}
# it means (A\B)∪(B\A)
st2.symmetric_difference(st1) # {'item1', 'item4'}
#Example:
whole_numbers = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
some_numbers = {1, 2, 3, 4, 5}
whole_numbers.symmetric_difference(some_numbers) # {0, 6, 7, 8, 9, 10}
python = {'p', 'y', 't', 'h', 'o','n'}
dragon = {'d', 'r', 'a', 'g', 'o','n'}
python.symmetric_difference(dragon)  # {'r', 't', 'p', 'y', 'g', 'a', 'd', 'h'}

#Joining Sets-
 If two sets do not have a common item or items we call them disjoint sets. We can check if two sets are j
oint or disjoint using isdisjoint() method.

st1 = {'item1', 'item2', 'item3', 'item4'}
st2 = {'item2', 'item3'}
st2.isdisjoint(st1) # False
#Example:
even_numbers = {0, 2, 4 ,6, 8}
```

```
odd_numbers = {1, 3, 5, 7, 9}
even_numbers.isdisjoint(odd_numbers) # True, because no common item

python = {'p', 'y', 't', 'h', 'o','n'}
dragon = {'d', 'r', 'a', 'g', 'o','n'}
python.isdisjoint(dragon)
```

Output

```
Does set st contain item3?  True
True
set()
{'potato', 'tomato', 'mango', 'carrot', 'cabbage', 'onion', 'orange', 'lemon', 'banana'}
{'potato', 'tomato', 'mango', 'carrot', 'cabbage', 'onion', 'orange', 'lemon', 'banana'}
False
```

13. Write a python program to count the number of characters present in a word

```
string=input("Enter string:")
char=0
word=1
for i in string:
    char=char+1
    if(i==' '):
        word=word+1
print("Number of words in the string:")
print(word)
print("Number of characters in the string:")
print(char)
```

Output

```
Enter string:PGDDS CDE
Number of words in the string:
2
Number of characters in the string:
9
```

14. Write a python program to demonstrate basic array characteristics

```
#Syntax to create python array
#from array import *
#arrayName = array(typecode, [initializers])

import array as arr
a = arr.array('i', [2, 4, 6, 8,10, 12, 14])
print("First element:", a[0])
print("Second element:", a[1])
print("Last element:", a[-1])
# changing first element
a[0] = 100
print(a)

# changing 3rd to 5th element
a[2:5] = arr.array('i', [104, 106, 108])
print(a)
del a[2]
print(a)
```

Output

```
First element: 2
Second element: 4
Last element: 14
array('i', [100, 4, 6, 8, 10, 12, 14])
array('i', [100, 4, 104, 106, 108, 12, 14])
array('i', [100, 4, 106, 108, 12, 14])
```

15. Write a python program to demonstrate array creation techniques

```
from array import *
T = [[11, 12, 5, 2], [15, 6,10], [10, 8, 12, 5], [12,15,8,6]]
print(T[0])
print(T[1][2])

for r in T:
  for c in r:
    print(c,end = " ")
  print()

T.insert(2, [0,5,11,13,6])
for r in T:
  for c in r:
    print(c,end = " ")
  print()

T[2] = [11,9]
T[0][3] = 7
for r in T:
  for c in r:
    print(c,end = " ")
  print()

del T[3]
for r in T:
  for c in r:
    print(c,end = " ")
  print()
```

Output

```
[11, 12, 5, 2]
10
11 12 5 2
15 6 10
10 8 12 5
12 15 8 6
11 12 5 2
15 6 10
0 5 11 13 6
```

10 8 12 5
12 15 8 6
11 12 5 7
15 6 10
11 9
10 8 12 5
12 15 8 6
11 12 5 7
15 6 10
11 9
12 15 8 6

16. Write a python program to demonstrate functions

```
#Function to generate full_name without parameters and without return
def generate_full_name ():
    first_name = 'Asabeneh'
    last_name = 'Yetayeh'
    space = ' '
    full_name = first_name + space + last_name
    print(full_name)
print('Function to generate full_name without parameters and without return')
generate_full_name () # calling a function

#Function to generate full_name without parameters and with return
def generate_full_name ():
    first_name = 'Asabeneh'
    last_name = 'Yetayeh'
    space = ' '
    full_name = first_name + space + last_name
    return full_name
print('Function to generate full_name without parameters and with return')
print(generate_full_name())

#Function to add two_numbers without parameters and without return
def add_two_numbers ():
    num_one = 2
    num_two = 3
    total = num_one + num_two
    print(total)
print('Function to add two_numbers without parameters and without return')
add_two_numbers()

#Function to add two_numbers without parameters and with return
def add_two_numbers ():
    num_one = 2
    num_two = 3
    total = num_one + num_two
    return total
print('Function to add two_numbers without parameters and with return')
print(add_two_numbers())

#Function with Parameters
#Function to generate greetings with parameter and with return
def greetings (name):
    message = name + ', welcome to Python for Everyone!'
    return message
print('Function to generate greetings with parameter and with return')
```

```
print(greetings('Asabeneh'))

#Function to add 10 with parameter and with return
def add_ten(num):
    ten = 10
    return num + ten
print('Function to add 10 with parameter and with return')
print(add_ten(90))

#Function to square a number with parameter and with return
def square_number(x):
    return x * x
print('Function to square a number with parameter and with return')
print(square_number(2))

#Function to calculate area of circle with parameter and with return
def area_of_circle (r):
    PI = 3.14
    area = PI * r ** 2
    return area
print('Function to calculate area of circle with parameter and with return')
print(area_of_circle(10))

#Function to find sum of numbers with parameter and with return
def sum_of_numbers(n):
    total = 0
    for i in range(n+1):
        total+=i
    print(total)
print('Function to find sum of numbers with parameter and with return')
print(sum_of_numbers(10)) # 55
print(sum_of_numbers(100)) # 5050

#Function to generate full_name with parameters and with return
def generate_full_name (first_name, last_name):
    space = ' '
    full_name = first_name + space + last_name
    return full_name
print('Function to generate full_name with parameters and with return')
print('Full Name: ', generate_full_name('Asabeneh','Yetayeh'))

#Function to add two numbers with parameters and with return
def sum_two_numbers (num_one, num_two):
    sum = num_one + num_two
    return sum
print('Function to add two numbers with parameters and with return')
print('Sum of two numbers: ', sum_two_numbers(1, 9))
```

```
#Function to calculate age with parameters and with return
def calculate_age (current_year, birth_year):
    age = current_year - birth_year
    return age;
print('Function to calculate age with parameters and with return')
print('Age: ', calculate_age(2023, 1998))

#Function to find the weight of object with parameters and with return
def weight_of_object (mass, gravity):
    weight = str(mass * gravity)+ ' N' # the value has to be changed to a string first
    return weight
print('Function to find the weight of object with parameters and with return')
print('Weight of an object in Newtons: ', weight_of_object(100, 9.81))

#Passing Arguments with Key and Value -
 If we pass the arguments with key and value, the order of the arguments does not matter.
#Function to generate full_name with key, value parameters
def print_fullname(firstname, lastname):
    space = ' '
    full_name = firstname  + space + lastname
    print(full_name)
print('Function to generate full_name with key, value parameters ')
print(print_fullname(firstname = 'Asabeneh', lastname = 'Yetayeh'))

#Function to add two numbers with key, value parameters
def add_two_numbers (num1, num2):
    total = num1 + num2
    print(total)
print('Function to add two numbers with key, value parameters')
print(add_two_numbers(num2 = 3, num1 = 2)) # Order does not matter

#Function Returning a Value - Returning a string Examples
print('Function Returning a Value - Returning a string')
def print_name(firstname):
    return firstname
print_name('Asabeneh') # Asabeneh

def print_full_name(firstname, lastname):
    space = ' '
    full_name = firstname  + space + lastname
    return full_name
print_full_name(firstname='Asabeneh', lastname='Yetayeh')

#Returning a number: Examples
print('Function Returning a Value - Returning a number')
def add_two_numbers (num1, num2):
```

```
    total = num1 + num2
    return total
print(add_two_numbers(2, 3))

def calculate_age (current_year, birth_year):
    age = current_year - birth_year
    return age;
print('Age: ', calculate_age(2019, 2001))

#Returning a boolean: Examples
print('Function Returning a Value - Returning a boolean')
def is_even (n):
    if n % 2 == 0:
        print('even')
        return True    # return stops further execution of the function, similar to break
    return False
print(is_even(10)) # True
print(is_even(7)) # False

#Returning a list: Examples
print('Function Returning a Value - Returning a List')
def find_even_numbers(n):
    evens = []
    for i in range(n + 1):
        if i % 2 == 0:
            evens.append(i)
    return evens
print(find_even_numbers(10))

#Function with Default Parameters Examples
print('Function with Default Parameters')
def greetings (name = 'Peter'):
    message = name + ', welcome to Python for Everyone!'
    return message
print(greetings())
print(greetings('Asabeneh'))

def generate_full_name (first_name = 'Asabeneh', last_name = 'Yetayeh'):
    space = ' '
    full_name = first_name + space + last_name
    return full_name

print(generate_full_name())
print(generate_full_name('David','Smith'))

def calculate_age (birth_year,current_year = 2021):
    age = current_year - birth_year
```

```
    return age;
print('Age: ', calculate_age(2001))

def weight_of_object (mass, gravity = 9.81):
    weight = str(mass * gravity)+ ' N' # the value has to be changed to string first
    return weight
print('Weight of an object in Newtons: ', weight_of_object(100)) # 9.81 -
 average gravity on Earth's surface
print('Weight of an object in Newtons: ', weight_of_object(100, 1.62)) # gravity on the surface of the Mo
on

#Arbitrary Number of Arguments Examples
def sum_all_nums(*nums):
    total = 0
    for num in nums:
        total += num     # same as total = total + num
    return total
print('Arbitrary Number of Arguments')
print(sum_all_nums(2, 3, 5)) # 10

#Default and Arbitrary Number of Parameters in Functions
def generate_groups (team,*args):
    print(team)
    for i in args:
        print(i)
print('Default and Arbitrary Number of Parameters in Functions')
print(generate_groups('Team-1','Asabeneh','Brook','David','Eyob'))

#Function as a Parameter of another Function
def square_number (n):
    return n * n
def do_something(f, x):
    return f(x)
print('Function as a Parameter of another Function')
print(do_something(square_number, 3))
```

Output

```
Function to generate full_name without parameters and without return
Asabeneh Yetayeh
Function to generate full_name without parameters and with return
Asabeneh Yetayeh
Function to add two_numbers without parameters and without return
5
Function to add two_numbers without parameters and with return
5
Function to generate greetings with parameter and with return
```

```
Asabeneh, welcome to Python for Everyone!
Function to add 10 with parameter and with return
100
Function to square a number with parameter and with return
4
Function to calculate area of circle with parameter and with return
314.0
Function to find sum of numbers with parameter and with return
55
None
5050
None
Function to generate full_name with parameters and with return
Full Name:  Asabeneh Yetayeh
Function to add two numbers with parameters and with return
Sum of two numbers:  10
Function to calculate age with parameters and with return
Age:  25
Function to find the weight of object with parameters and with return
Weight of an object in Newtons:  981.0 N
Function to generate full_name with key, value parameters
Asabeneh Yetayeh
None
Function to add two numbers with key, value parameters
5
None
Function Returning a Value - Returning a string
Function Returning a Value - Returning a number
5
Age:  18
Function Returning a Value - Returning a boolean
even
True
False
Function Returning a Value - Returning a List
[0, 2, 4, 6, 8, 10]
Function with Default Parameters
Peter, welcome to Python for Everyone!
Asabeneh, welcome to Python for Everyone!
Asabeneh Yetayeh
David Smith
Age:  20
Weight of an object in Newtons:  981.0 N
Weight of an object in Newtons:  162.0 N
Arbitrary Number of Arguments
10
Default and Arbitrary Number of Parameters in Functions
Team-1
Asabeneh
```

Brook
David
Eyob
None
Function as a Parameter of another Function
9

17. Demonstrate about the built-in functions

In Python, we have lots of built-in functions. Built-in functions are globally available for your use that means you can make use of the built-in functions without importing or configuring. Some of the most commonly used Python built-in functions are the following:

print(), len(), type(), int(), float(), str(), input(), list(), dict(), min(), max(), sum(), sorted(), open(), file(),

help(), and *dir()*.

In the following table you will see an exhaustive list of Python built-in functions taken from [python documentation]

		Built-in Functions		
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

```
help('keywords')
```

Output

Here is a list of the Python keywords. Enter any keyword to get more help.

```
False               break               for                 not
None                class               from                or
True                continue            global              pass
__peg_parser__      def                 if                  raise
and                 del                 import              return
as                  elif                in                  try
assert              else                is                  while
async               except              lambda              with
await               finally             nonlocal            yield
```

```
help('str')
```

Output

Help on class str in module builtins:

```
class str(object)
 |  str(object='') -> str
 |  str(bytes_or_buffer[, encoding[, errors]]) -> str
 |
 |  Create a new string object from the given object. If encoding or
 |  errors is specified, then the object must expose a data buffer
 |  that will be decoded using the given encoding and error handler.
 |  Otherwise, returns the result of object.__str__() (if defined)
 |  or repr(object).
 |  encoding defaults to sys.getdefaultencoding().
 |  errors defaults to 'strict'.
 |
 |  Methods defined here:
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __contains__(self, key, /)
 |      Return key in self.
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __format__(self, format_spec, /)
 |      Return a formatted version of the string as described by format_spec.
 |
```

```
|  __ge__(self, value, /)
|      Return self>=value.
|
|  __getattribute__(self, name, /)
|      Return getattr(self, name).
|
|  __getitem__(self, key, /)
|      Return self[key].
|
|  __getnewargs__(...)
|
|  __gt__(self, value, /)
|      Return self>value.
|
|  __hash__(self, /)
|      Return hash(self).
|
|  __iter__(self, /)
|      Implement iter(self).
|
|  __le__(self, value, /)
|      Return self<=value.
|
|  __len__(self, /)
|      Return len(self).
|
|  __lt__(self, value, /)
|      Return self<value.
|
|  __mod__(self, value, /)
|      Return self%value.
|
|  __mul__(self, value, /)
|      Return self*value.
|
|  __ne__(self, value, /)
|      Return self!=value.
|
|  __repr__(self, /)
|      Return repr(self).
|
|  __rmod__(self, value, /)
|      Return value%self.
|
|  __rmul__(self, value, /)
|      Return value*self.
|
|  __sizeof__(self, /)
|      Return the size of the string in memory, in bytes.
```

```
|
| __str__(self, /)
|     Return str(self).
|
| capitalize(self, /)
|     Return a capitalized version of the string.
|
|     More specifically, make the first character have upper case and the rest lower
|     case.
|
| casefold(self, /)
|     Return a version of the string suitable for caseless comparisons.
|
| center(self, width, fillchar=' ', /)
|     Return a centered string of length width.
|
|     Padding is done using the specified fill character (default is a space).
|
| count(...)
|     S.count(sub[, start[, end]]) -> int
|
|     Return the number of non-overlapping occurrences of substring sub in
|     string S[start:end].  Optional arguments start and end are
|     interpreted as in slice notation.
|
| encode(self, /, encoding='utf-8', errors='strict')
|     Encode the string using the codec registered for encoding.
|
|     encoding
|       The encoding in which to encode the string.
|     errors
|       The error handling scheme to use for encoding errors.
|       The default is 'strict' meaning that encoding errors raise a
|       UnicodeEncodeError.  Other possible values are 'ignore', 'replace' and
|       'xmlcharrefreplace' as well as any other name registered with
|       codecs.register_error that can handle UnicodeEncodeErrors.
|
| endswith(...)
|     S.endswith(suffix[, start[, end]]) -> bool
|
|     Return True if S ends with the specified suffix, False otherwise.
|     With optional start, test S beginning at that position.
|     With optional end, stop comparing S at that position.
|     suffix can also be a tuple of strings to try.
|
| expandtabs(self, /, tabsize=8)
|     Return a copy where all tab characters are expanded using spaces.
|
|     If tabsize is not given, a tab size of 8 characters is assumed.
```

```
|
| find(...)
|     S.find(sub[, start[, end]]) -> int
|
|     Return the lowest index in S where substring sub is found,
|     such that sub is contained within S[start:end].  Optional
|     arguments start and end are interpreted as in slice notation.
|
|     Return -1 on failure.
|
| format(...)
|     S.format(*args, **kwargs) -> str
|
|     Return a formatted version of S, using substitutions from args and kwargs.
|     The substitutions are identified by braces ('{' and '}').
|
| format_map(...)
|     S.format_map(mapping) -> str
|
|     Return a formatted version of S, using substitutions from mapping.
|     The substitutions are identified by braces ('{' and '}').
|
| index(...)
|     S.index(sub[, start[, end]]) -> int
|
|     Return the lowest index in S where substring sub is found,
|     such that sub is contained within S[start:end].  Optional
|     arguments start and end are interpreted as in slice notation.
|
|     Raises ValueError when the substring is not found.
|
| isalnum(self, /)
|     Return True if the string is an alpha-numeric string, False otherwise.
|
|     A string is alpha-numeric if all characters in the string are alpha-numeric and
|     there is at least one character in the string.
|
| isalpha(self, /)
|     Return True if the string is an alphabetic string, False otherwise.
|
|     A string is alphabetic if all characters in the string are alphabetic and there
|     is at least one character in the string.
|
| isascii(self, /)
|     Return True if all characters in the string are ASCII, False otherwise.
|
|     ASCII characters have code points in the range U+0000-U+007F.
|     Empty string is ASCII too.
|
```

```
|  isdecimal(self, /)
|      Return True if the string is a decimal string, False otherwise.
|
|      A string is a decimal string if all characters in the string are decimal and
|      there is at least one character in the string.
|
|  isdigit(self, /)
|      Return True if the string is a digit string, False otherwise.
|
|      A string is a digit string if all characters in the string are digits and there
|      is at least one character in the string.
|
|  isidentifier(self, /)
|      Return True if the string is a valid Python identifier, False otherwise.
|
|      Call keyword.iskeyword(s) to test whether string s is a reserved identifier,
|      such as "def" or "class".
|
|  islower(self, /)
|      Return True if the string is a lowercase string, False otherwise.
|
|      A string is lowercase if all cased characters in the string are lowercase and
|      there is at least one cased character in the string.
|
|  isnumeric(self, /)
|      Return True if the string is a numeric string, False otherwise.
|
|      A string is numeric if all characters in the string are numeric and there is at
|      least one character in the string.
|
|  isprintable(self, /)
|      Return True if the string is printable, False otherwise.
|
|      A string is printable if all of its characters are considered printable in
|      repr() or if it is empty.
|
|  isspace(self, /)
|      Return True if the string is a whitespace string, False otherwise.
|
|      A string is whitespace if all characters in the string are whitespace and there
|      is at least one character in the string.
|
|  istitle(self, /)
|      Return True if the string is a title-cased string, False otherwise.
|
|      In a title-cased string, upper- and title-case characters may only
|      follow uncased characters and lowercase characters only cased ones.
|
|  isupper(self, /)
```

```
|     Return True if the string is an uppercase string, False otherwise.
|
|     A string is uppercase if all cased characters in the string are uppercase and
|     there is at least one cased character in the string.
|
| join(self, iterable, /)
|     Concatenate any number of strings.
|
|     The string whose method is called is inserted in between each given string.
|     The result is returned as a new string.
|
|     Example: '.'.join(['ab', 'pq', 'rs']) -> 'ab.pq.rs'
|
| ljust(self, width, fillchar=' ', /)
|     Return a left-justified string of length width.
|
|     Padding is done using the specified fill character (default is a space).
|
| lower(self, /)
|     Return a copy of the string converted to lowercase.
|
| lstrip(self, chars=None, /)
|     Return a copy of the string with leading whitespace removed.
|
|     If chars is given and not None, remove characters in chars instead.
|
| partition(self, sep, /)
|     Partition the string into three parts using the given separator.
|
|     This will search for the separator in the string.  If the separator is found,
|     returns a 3-tuple containing the part before the separator, the separator
|     itself, and the part after it.
|
|     If the separator is not found, returns a 3-tuple containing the original string
|     and two empty strings.
|
| removeprefix(self, prefix, /)
|     Return a str with the given prefix string removed if present.
|
|     If the string starts with the prefix string, return string[len(prefix):].
|     Otherwise, return a copy of the original string.
|
| removesuffix(self, suffix, /)
|     Return a str with the given suffix string removed if present.
|
|     If the string ends with the suffix string and that suffix is not empty,
|     return string[:-len(suffix)]. Otherwise, return a copy of the original
|     string.
|
```

```
| replace(self, old, new, count=-1, /)
|     Return a copy with all occurrences of substring old replaced by new.
|
|       count
|         Maximum number of occurrences to replace.
|         -1 (the default value) means replace all occurrences.
|
|     If the optional argument count is given, only the first count occurrences are
|     replaced.
|
| rfind(...)
|     S.rfind(sub[, start[, end]]) -> int
|
|     Return the highest index in S where substring sub is found,
|     such that sub is contained within S[start:end].  Optional
|     arguments start and end are interpreted as in slice notation.
|
|     Return -1 on failure.
|
| rindex(...)
|     S.rindex(sub[, start[, end]]) -> int
|
|     Return the highest index in S where substring sub is found,
|     such that sub is contained within S[start:end].  Optional
|     arguments start and end are interpreted as in slice notation.
|
|     Raises ValueError when the substring is not found.
|
| rjust(self, width, fillchar=' ', /)
|     Return a right-justified string of length width.
|
|     Padding is done using the specified fill character (default is a space).
|
| rpartition(self, sep, /)
|     Partition the string into three parts using the given separator.
|
|     This will search for the separator in the string, starting at the end. If
|     the separator is found, returns a 3-tuple containing the part before the
|     separator, the separator itself, and the part after it.
|
|     If the separator is not found, returns a 3-tuple containing two empty strings
|     and the original string.
|
| rsplit(self, /, sep=None, maxsplit=-1)
|     Return a list of the words in the string, using sep as the delimiter string.
|
|       sep
|         The delimiter according which to split the string.
|         None (the default value) means split according to any whitespace,
```

```
|         and discard empty strings from the result.
|       maxsplit
|         Maximum number of splits to do.
|         -1 (the default value) means no limit.
|
|     Splits are done starting at the end of the string and working to the front.
|
| rstrip(self, chars=None, /)
|     Return a copy of the string with trailing whitespace removed.
|
|     If chars is given and not None, remove characters in chars instead.
|
| split(self, /, sep=None, maxsplit=-1)
|     Return a list of the words in the string, using sep as the delimiter string.
|
|     sep
|       The delimiter according which to split the string.
|       None (the default value) means split according to any whitespace,
|       and discard empty strings from the result.
|     maxsplit
|       Maximum number of splits to do.
|       -1 (the default value) means no limit.
|
| splitlines(self, /, keepends=False)
|     Return a list of the lines in the string, breaking at line boundaries.
|
|     Line breaks are not included in the resulting list unless keepends is given and
|     true.
|
| startswith(...)
|     S.startswith(prefix[, start[, end]]) -> bool
|
|     Return True if S starts with the specified prefix, False otherwise.
|     With optional start, test S beginning at that position.
|     With optional end, stop comparing S at that position.
|     prefix can also be a tuple of strings to try.
|
| strip(self, chars=None, /)
|     Return a copy of the string with leading and trailing whitespace removed.
|
|     If chars is given and not None, remove characters in chars instead.
|
| swapcase(self, /)
|     Convert uppercase characters to lowercase and lowercase characters to uppercase.
|
| title(self, /)
|     Return a version of the string where each word is titlecased.
|
|     More specifically, words start with uppercased characters and all remaining
```

```
|  cased characters have lower case.
|
| translate(self, table, /)
|  Replace each character in the string using the given translation table.
|
|    table
|      Translation table, which must be a mapping of Unicode ordinals to
|      Unicode ordinals, strings, or None.
|
|  The table must implement lookup/indexing via __getitem__, for instance a
|  dictionary or list.  If this operation raises LookupError, the character is
|  left untouched.  Characters mapped to None are deleted.
|
| upper(self, /)
|  Return a copy of the string converted to uppercase.
|
| zfill(self, width, /)
|  Pad a numeric string with zeros on the left, to fill a field of the given width.
|
|  The string is never truncated.
|
| ----------------------------------------------------------------------
| Static methods defined here:
|
| __new__(*args, **kwargs) from builtins.type
|  Create and return a new object.  See help(type) for accurate signature.
|
| maketrans(...)
|  Return a translation table usable for str.translate().
|
|  If there is only one argument, it must be a dictionary mapping Unicode
|  ordinals (integers) or characters to Unicode ordinals, strings or None.
|  Character keys will be then converted to ordinals.
|  If there are two arguments, they must be strings of equal length, and
|  in the resulting dictionary, each character in x will be mapped to the
|  character at the same position in y. If there is a third argument, it
|  must be a string, whose characters will be mapped to None in the result.
```

18. Write a python program to design simple calculator using functions

```
from builtins import int
while True:
    # Print options for the user
    print("Enter '+' to add two numbers")
    print("Enter '-' to subtract two numbers")
    print("Enter '*' to multiply two numbers")
    print("Enter '/' to divide two numbers")
    print("Enter 'quit' to end the program")

    # Get user input
    user_input = input(": ")

    # Check if the user wants to quit
    if user_input == "quit":
        break
    # Check if the user input is a valid operator
    elif user_input in ["+", "-", "*", "/"]:
        # Get first number
        num1 = int(input("Enter a number: "))
        # Get second number
        num2 = int(input("Enter another number: "))

        # Perform the operation based on the user input
        if user_input == "+":
            result = num1 + num2
            print(num1, "+", num2, "=", result)
        elif user_input == "-":
            result = num1 - num2
            print(num1, "-", num2, "=", result)
        elif user_input == "*":
            result = num1 * num2
            print(num1, "*", num2, "=", result)
        elif user_input == "/":
            result = num1 / num2
            print(num1, "/", num2, "=", result)
        else:
            # In case of invalid input
            print("Invalid Input")
```

Ouput

```
Enter '+' to add two numbers
Enter '-' to subtract two numbers
Enter '*' to multiply two numbers
Enter '/' to divide two numbers
Enter 'quit' to end the program
: +
Enter a number: 5
Enter another number: 8
5 + 8 = 13
Enter '+' to add two numbers
Enter '-' to subtract two numbers
Enter '*' to multiply two numbers
Enter '/' to divide two numbers
Enter 'quit' to end the program
: q
Enter '+' to add two numbers
Enter '-' to subtract two numbers
Enter '*' to multiply two numbers
Enter '/' to divide two numbers
Enter 'quit' to end the program
: quit
```

19. Write a python program to find factorial of a number using recursion

```
# Factorial of a number using recursion
def recur_factorial(n):
   if n == 1:
      return n
   else:
      return n*recur_factorial(n-1)

num = 7

# check if the number is negative
if num < 0:
   print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
   print("The factorial of 0 is 1")
else:
   print("The factorial of", num, "is", recur_factorial(num))
```

Output

The factorial of 7 is 5040

20. Write a python program to check whether a string is palindrome or not

```
# function which return reverse of a string

def isPalindrome(s):
  return s == s[::-1]


# Driver code
s = "malayalam"
ans = isPalindrome(s)

if ans:
  print("Yes")
else:
  print("No")
```

Output

Yes

21. Write a python program to demonstrate modules

#A module is a file containing a set of codes or a set of functions which can be included to an application. A module could be a file containing a single variable, a function or a big code base.

#Creating a Module -
To create a module we write our codes in a python script and we save it as a .py file. Create a file named mymodule.py inside your project folder. Let us write some code in this file.

```
# mymodule.py file
def generate_full_name(firstname, lastname):
    space = ' '
    fullname = firstname + space + lastname
    return fullname

def sum_two_nums(num1, num2):
    return num1 + num2

gravity = 9.81
person = {
    "firstname": "Asabeneh",
    "age": 250,
    "country": "Finland",
    "city": 'Helsinki'
}
```

#Importing a Module - To import the file we use the *import* keyword and the name of the file only.

```
import mymodule
print(mymodule.generate_full_name('Asabeneh', 'Yetayeh')) # Asabeneh Yetayeh


#Import Functions from a Module - We can have many functions in a file and we can
import all the functions differently.

from mymodule import generate_full_name, sum_two_nums, person, gravity
print(generate_full_name('Asabneh','Yetayeh'))
print(sum_two_nums(1,9))
mass = 100;
weight = mass * gravity
print(weight)
print(person['firstname'])
```

#Import Functions from a Module and Renaming - During importing we can rename the name of the module.

```
from mymodule import generate_full_name as fullname, sum_two_nums as total, person as p, gravity as g
print(fullname('Asabneh','Yetayeh'))
print(total(1, 9))
mass = 100;
weight = mass * g
print(weight)
print(p)
print(p['firstname'])
```

Output

```
Asabeneh Yetayeh
Asabneh Yetayeh
10
981.0
Asabeneh
Asabneh Yetayeh
10
981.0
{'firstname': 'Asabeneh', 'age': 250, 'country': 'Finland', 'city': 'Helsinki'}
Asabeneh
```

#Import Built-in Modules - Some of the common built-in modules*: math, datetime, os,sys, random, statistics, collections, json, re*

#OS Module - Using python *os* module it is possible to automatically perform many operating system tasks. The OS module in Python provides functions for creating, changing current working directory, and removing a directory (folder), fetching its contents, changing and identifying the current directory.

```
import os
# Creating a directory
os.mkdir('directory_name')
# Changing the current directory
#os.chdir('path')
# Getting current working directory
os.getcwd()
# Removing directory
os.rmdir('directory_name')
```

#Sys Module - The sys module provides functions and variables used to manipulate different parts of the Python runtime environment. Function sys.argv returns a list of command line arguments passed to a Python script. The item at index 0 in this list is always the name of the script, at index 1 is the argument passed from the command line.

```
import sys
#print(sys.argv[0], argv[1],sys.argv[2])  # this line would print out: filename argument1 argument2
print('Welcome {}. Enjoy  {} challenge!'.format('John', 'Python'))
#Now to check how this script works I wrote in command line:
```

```
#python script.py Asabeneh 30DaysOfPython
#The result:
#Welcome Asabeneh. Enjoy  30DayOfPython challenge!
#Some useful sys commands:
# to exit sys
#sys.exit()
# To know the largest integer variable it takes
sys.maxsize
# To know environment path
sys.path
# To know the version of python you are using
sys.version
```

#Statistics Module - The statistics module provides functions for mathematical statistics of numeric data. The popular statistical functions which are defined in this module: ***mean***, ***median***, ***mode***, ***stdev*** etc.

```
from statistics import * # importing all the statistics modules
ages = [20, 20, 4, 24, 25, 22, 26, 20, 23, 22, 26]
print(mean(ages))       # ~22.9
print(median(ages))     # 23
print(mode(ages))       # 20
print(stdev(ages))      # ~2.3
```

#Math Module - Module containing many mathematical operations and constants.

```
import math
print(math.pi)          # 3.141592653589793, pi constant
print(math.sqrt(2))     # 1.4142135623730951, square root
print(math.pow(2, 3))   # 8.0, exponential function
print(math.floor(9.81)) # 9, rounding to the lowest
print(math.ceil(9.81))  # 10, rounding to the highest
print(math.log10(100))  # 2, logarithm with 10 as base
from math import pi
print(pi)
#It is also possible to import multiple functions at once
from math import pi, sqrt, pow, floor, ceil, log10
print(pi)               # 3.141592653589793
print(sqrt(2))          # 1.4142135623730951
print(pow(2, 3))        # 8.0
print(floor(9.81))      # 9
print(ceil(9.81))       # 10
print(math.log10(100))  # 2
#But if we want to import all the function in math module we can use * .
from math import *
print(pi)               # 3.141592653589793, pi constant
```

```
print(sqrt(2))          # 1.4142135623730951, square root
print(pow(2, 3))        # 8.0, exponential
print(floor(9.81))      # 9, rounding to the lowest
print(ceil(9.81))       # 10, rounding to the highest
print(math.log10(100))  # 2
#When we import we can also rename the name of the function.
from math import pi as  PI
print(PI) # 3.141592653589793
```

#String Module - A string module is a useful module for many purposes. The example below shows some use of the string module.

```
import string
print(string.ascii_letters) # abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
print(string.digits)        # 0123456789
print(string.punctuation)   # !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

#Random Module - By now you are familiar with importing modules. The *random* module has lots of functions but in this section we will only use *random* and *randint*.

```
from random import random, randint
print(random())  # it doesn't take any arguments; it returns a value between 0 and 0.9999
print(randint(5, 20))
```

Output

```
Welcome John. Enjoy  Python challenge!
21.09090909090909
22
20
6.106628291529549
3.141592653589793
1.4142135623730951
8.0
9
10
2.0
3.141592653589793
3.141592653589793
1.4142135623730951
8.0
9
10
2.0
3.141592653589793
```

1.4142135623730951
8.0
9
10
2.0
3.141592653589793
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
0.21960524635597978
7

22. Write a python program on classes and objects

#Python is an object oriented programming language. Everything in Python is an object, with its properties and methods. A number, string, list, dictionary, tuple, set etc. used in a program is an object of a corresponding built-in class. We create class to create an object. Let us check if everything in python is a class:

```
>>> num = 10
>>> type(num)
<class 'int'>
>>> string = 'string'
>>> type(string)
<class 'str'>
>>> boolean = True
>>> type(boolean)
<class 'bool'>
>>> lst = []
>>> type(lst)
<class 'list'>
>>> tpl = ()
>>> type(tpl)
<class 'tuple'>
>>> set1 = set()
>>> type(set1)
<class 'set'>
>>> dct = {}
>>> type(dct)
<class 'dict'>
```

```
#Creating a Class -
 To create a class we need the key word class followed by the name and colon. Class name should be Ca
melCase.

class Person:
  pass
print(Person)
#Creating an Object - We can create an object by calling the class.

p = Person()
print(p)
#Class Constructor
class Person:
    def __init__ (self, name):
```

```
        # self allows to attach parameter to the class
          self.name =name

p = Person('Asabeneh')
print(p.name)
print(p)

class Person:
    def __init__(self, firstname, lastname, age, country, city):
        self.firstname = firstname
        self.lastname = lastname
        self.age = age
        self.country = country
        self.city = city


p = Person('Asabeneh', 'Yetayeh', 50, 'Finland', 'Helsinki')
print(p.firstname)
print(p.lastname)
print(p.age)
print(p.country)
print(p.city)
#Object Methods - Objects can have methods. The methods are functions which belong to the object.

class Person:
    def __init__(self, firstname, lastname, age, country, city):
        self.firstname = firstname
        self.lastname = lastname
        self.age = age
        self.country = country
        self.city = city
    def person_info(self):
      return f'{self.firstname} {self.lastname} is {self.age} years old. He lives in {self.city}, {self.country
}'

p = Person('Asabeneh', 'Yetayeh', 50, 'Finland', 'Helsinki')
print(p.person_info())
#Object Default Methods
class Person:
    def __init__(self, firstname='Asabeneh', lastname='Yetayeh', age=50, country='Finland', city='Helsin
ki'):
        self.firstname = firstname
        self.lastname = lastname
        self.age = age
        self.country = country
        self.city = city
```

```
    def person_info(self):
      return f'{self.firstname} {self.lastname} is {self.age} years old. He lives in {self.city}, {self.country
}.'

p1 = Person()
print(p1.person_info())
p2 = Person('John', 'Doe', 30, 'Nomanland', 'Noman city')
print(p2.person_info())
```

Output

```
<class '__main__.Person'>
<__main__.Person object at 0x7f5bb8138e20>
Asabeneh
<__main__.Person object at 0x7f5bb8138e80>
Asabeneh
Yetayeh
50
Finland
Helsinki
Asabeneh Yetayeh is 50 years old. He lives in Helsinki, Finland
Asabeneh Yetayeh is 50 years old. He lives in Helsinki, Finland.
John Doe is 30 years old. He lives in Noman city, Nomanland.
```

23. Write a program to implement the inheritance

```
#Inheritance
class Person:
    def __init__(self, firstname='Asabeneh', lastname='Yetayeh', age=250, country='Finland', city='Helsi
nki'):
        self.firstname = firstname
        self.lastname = lastname
        self.age = age
        self.country = country
        self.city = city
        self.skills = []

    def person_info(self):
      return f'{self.firstname} {self.lastname} is {self.age} years old. He lives in {self.city}, {self.country
}.'
    def add_skill(self, skill):
        self.skills.append(skill)

p1 = Person()
print(p1.person_info())
p1.add_skill('HTML')
p1.add_skill('CSS')
p1.add_skill('JavaScript')
p2 = Person('John', 'Doe', 30, 'Nomanland', 'Noman city')
print(p2.person_info())
print(p1.skills)
print(p2.skills)

class Student(Person):
   pass
s1 = Student('Eyob', 'Yetayeh', 30, 'Finland', 'Helsinki')
s2 = Student('Lidiya', 'Teklemariam', 28, 'Finland', 'Espoo')
print(s1.person_info())
s1.add_skill('JavaScript')
s1.add_skill('React')
s1.add_skill('Python')
print(s1.skills)

print(s2.person_info())
s2.add_skill('Organizing')
s2.add_skill('Marketing')
```

```
s2.add_skill('Digital Marketing')
print(s2.skills)
#Overriding parent method

class Student(Person):
    def __init__ (self, firstname='Asabeneh', lastname='Yetayeh',age=250, country='Finland', city='Helsinki', gender='male'):
        self.gender = gender
        super().__init__(firstname, lastname,age, country, city)

    def person_info(self):
        gender = 'He' if self.gender =='male' else 'She'
        return f'{self.firstname} {self.lastname} is {self.age} years old. {gender} lives in {self.city}, {self.country}.'

s1 = Student('Eyob', 'Yetayeh', 30, 'Finland', 'Helsinki','male')
s2 = Student('Lidiya', 'Teklemariam', 28, 'Finland', 'Espoo', 'female')
print(s1.person_info())
s1.add_skill('JavaScript')
s1.add_skill('React')
s1.add_skill('Python')
print(s1.skills)

print(s2.person_info())
s2.add_skill('Organizing')
s2.add_skill('Marketing')
s2.add_skill('Digital Marketing')
print(s2.skills)
```

Output

```
Asabeneh Yetayeh is 250 years old. He lives in Helsinki, Finland.
John Doe is 30 years old. He lives in Noman city, Nomanland.
['HTML', 'CSS', 'JavaScript']
[]
Eyob Yetayeh is 30 years old. He lives in Helsinki, Finland.
['JavaScript', 'React', 'Python']
Lidiya Teklemariam is 28 years old. He lives in Espoo, Finland.
['Organizing', 'Marketing', 'Digital Marketing']
Eyob Yetayeh is 30 years old. He lives in Helsinki, Finland.
['JavaScript', 'React', 'Python']
Lidiya Teklemariam is 28 years old. She lives in Espoo, Finland.
```

['Organizing', 'Marketing', 'Digital Marketing']

24. Write a python program to demonstrate indexing in numpy

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
print(arr)
print(type(arr))
#Use a tuple to create a NumPy array
arr = np.array((1, 2, 3, 4, 5))
print(arr)

#Create a 0-D array with value 42
arr = np.array(42)
print(arr)

#Create a 1-D array containing the values 1,2,3,4,5
arr = np.array([1, 2, 3, 4, 5])
print(arr)

#Create a 2-D array containing two arrays with the values 1,2,3 and 4,5,6
arr = np.array([[1, 2, 3], [4, 5, 6]])
print(arr)

#Create a 3-D array with two 2-D arrays, both containing two arrays with the values 1,2,3 and 4,5,6
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])
print(arr)

#Check how many dimensions the arrays have
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)

#Get the first element from the following array
arr = np.array([1, 2, 3, 4])
print(arr[0])

#Get the second element from the following array
arr = np.array([1, 2, 3, 4])
```

```
print(arr[1])

#Get third and fourth elements from the following array and add them
arr = np.array([1, 2, 3, 4])
print(arr[2] + arr[3])

#Access the element on the first row, second column
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])
print('2nd element on 1st row: ', arr[0, 1])

#Access the element on the 2nd row, 5th column
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])
print('5th element on 2nd row: ', arr[1, 4])

#Negative indexing - Print the last element from the 2nd dim
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])
print('Last element from 2nd dim: ', arr[1, -1])

#Slicing - Slice elements from index 1 to index 5 from the following array
arr = np.array([1, 2, 3, 4, 5, 6, 7])
print(arr[1:5])

#Slice elements from index 4 to the end of the array
arr = np.array([1, 2, 3, 4, 5, 6, 7])
print(arr[4:])

#Slice elements from the beginning to index 4 (not included)
arr = np.array([1, 2, 3, 4, 5, 6, 7])
print(arr[:4])

#Slice from the index 3 from the end to index 1 from the end
arr = np.array([1, 2, 3, 4, 5, 6, 7])
print(arr[-3:-1])

#Return every other element from index 1 to index 5
arr = np.array([1, 2, 3, 4, 5, 6, 7])
print(arr[1:5:2])

#Return every other element from the entire array
arr = np.array([1, 2, 3, 4, 5, 6, 7])
print(arr[::2])
```

```
#Slicing 2-D Arrays
#From the second element, slice elements from index 1 to index 4 (not included)
arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])
print(arr[1, 1:4])

#From both elements, return index 2
arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])
print(arr[0:2, 2])

#From both elements, slice index 1 to index 4 (not included), this will return a 2-D array
arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])
print(arr[0:2, 1:4])
```

Output

```
[1 2 3 4 5]
<class 'numpy.ndarray'>
[1 2 3 4 5]
42
[1 2 3 4 5]
[[1 2 3]
 [4 5 6]]
[[[1 2 3]
  [4 5 6]]

 [[1 2 3]
  [4 5 6]]]
0
1
2
3
1
2
7
2nd element on 1st row:  2
5th element on 2nd row:  10
Last element from 2nd dim:  10
[2 3 4 5]
[5 6 7]
[1 2 3 4]
[5 6]
[2 4]
```

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

25. Write a python program to demonstrate list comprehension

```
#List Comprehension
#List comprehension in Python is a compact way of creating a list from a sequence. It is a short way to cr
eate a new list. List comprehension is considerably faster than processing a list using the for loop.

# syntax
#[i for i in iterable if expression]

#Example:2 - For instance if you want to change a string to a list of characters.
# One way
language = 'Python'
lst = list(language) # changing the string to list
print(type(lst))     # list
print(lst)          # ['P', 'y', 't', 'h', 'o', 'n']

# Second way: list comprehension
lst = [i for i in language]
print(type(lst)) # list
print(lst)       # ['P', 'y', 't', 'h', 'o', 'n']

#Example:2 - For instance if you want to generate a list of numbers - Generating numbers
numbers = [i for i in range(11)]  # to generate numbers from 0 to 10
print(numbers)                  # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# It is possible to do mathematical operations during iteration
squares = [i * i for i in range(11)]
print(squares)                  # [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

# It is also possible to make a list of tuples
numbers = [(i, i * i) for i in range(11)]
print(numbers)                          # [(0, 0), (1, 1), (2, 4), (3, 9), (4, 16), (5, 25)]

#Example:2 - List comprehension can be combined with if expression - Generating even numbers
even_numbers = [i for i in range(21) if i % 2 == 0]  # to generate even numbers list in range 0 to 21
print(even_numbers)                  # [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

# Generating odd numbers
odd_numbers = [i for i in range(21) if i % 2 != 0]  # to generate odd numbers in range 0 to 21
print(odd_numbers)                   # [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
# Filter numbers: let's filter out positive even numbers from the list below
numbers = [-8, -7, -3, -1, 0, 1, 3, 4, 5, 7, 6, 8, 10]
positive_even_numbers = [i for i in range(21) if i % 2 == 0 and i > 0]
print(positive_even_numbers)                 # [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

# Flattening a three dimensional array
```

```
list_of_lists = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
flattened_list = [ number for row in list_of_lists for number in row]
print(flattened_list)    # [1, 2, 3, 4, 5, 6, 7, 8, 9]

#Lambda Function -
 Lambda function is a small anonymous function without a name. It can take any number of arguments, b
ut can only have one expression. Lambda function is similar to anonymous functions in JavaScript. We n
eed it when we want to write an anonymous function inside another function.
#Creating a Lambda Function -
 To create a lambda function we use lambda keyword followed by a parameter(s), followed by an express
ion. See the syntax and the example below. Lambda function does not use return but it explicitly returns t
he expression.
# syntax
arg1 = int(input('Enter a value for num1 - '))
arg2 = int(input('Enter a value for num2 - '))
arg3 = int(input('Enter a value for num3 - '))
x = lambda param1, param2, param3: param1 + param2 + param2
print(x(arg1, arg2, arg3))

#Example: Named function
def add_two_nums(a, b):
    return a + b

print(add_two_nums(2, 3))     # 5
# Lets change the above function to a lambda function
add_two_nums = lambda a, b: a + b
print(add_two_nums(2,3))    # 5

# Self invoking lambda function
(lambda a, b: a + b)(2,3) # 5 - need to encapsulate it in print() to see the result in the console

square = lambda x : x ** 2
print(square(3))    # 9
cube = lambda x : x ** 3
print(cube(3))    # 27

# Multiple variables
multiple_variable = lambda a, b, c: a ** 2 - 3 * b + 4 * c
print(multiple_variable(5, 5, 3)) # 22

#Lambda Function Inside Another Function
#Using a lambda function inside another function.

def power(x):
    return lambda n : x ** n

cube = power(2)(3)   # function power now need 2 arguments to run, in separate rounded brackets
```

```
print(cube)         # 8
two_power_of_five = power(2)(5)
print(two_power_of_five)  # 32
```

Output

```
<class 'list'>
['P', 'y', 't', 'h', 'o', 'n']
<class 'list'>
['P', 'y', 't', 'h', 'o', 'n']
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
[(0, 0), (1, 1), (2, 4), (3, 9), (4, 16), (5, 25), (6, 36), (7, 49), (8, 64), (9, 81), (10, 100)]
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
[1, 2, 3, 4, 5, 6, 7, 8, 9]
Enter a value for num1 - 7
Enter a value for num2 - 8
Enter a value for num3 - 9
23
5
5
9
27
22
8
32
```

26. Write a python program to work with date and time

```
#Python datetime
#Python has got datetime module to handle date and time.

import datetime
print(dir(datetime))
['MAXYEAR', 'MINYEAR', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '
__package__', '__spec__', 'date', 'datetime', 'datetime_CAPI', 'sys', 'time', 'timedelta', 'timezone', 'tzinfo']

#With dir or help builtin commands it is possible to know the available functions in a certain module. As
you can see, in the datetime module there are many functions, but we will focus on date, datetime, time a
nd timedelta.

#Getting datetime Information
from datetime import datetime
now = datetime.now()
print(now)                  # 2021-07-08 07:34:46.549883
day = now.day               # 8
month = now.month           # 7
year = now.year             # 2021
hour = now.hour             # 7
minute = now.minute         # 38
second = now.second
timestamp = now.timestamp()
print(day, month, year, hour, minute)
print('timestamp', timestamp)
print(f'{day}/{month}/{year}, {hour}:{minute}')  # 8/7/2021, 7:38
#Timestamp or Unix timestamp is the number of seconds elapsed from 1st of January 1970 UTC.

#Formatting Date Output Using strftime
from datetime import datetime
new_year = datetime(2020, 1, 1)
print(new_year)     # 2020-01-01 00:00:00
day = new_year.day
month = new_year.month
year = new_year.year
hour = new_year.hour
minute = new_year.minute
second = new_year.second
print(day, month, year, hour, minute) #1 1 2020 0 0
print(f'{day}/{month}/{year}, {hour}:{minute}')  # 1/1/2020, 0:0
#Formatting date time using strftime method

from datetime import datetime
# current date and time
now = datetime.now()
```

```
t = now.strftime("%H:%M:%S")
print("time:", t)
time_one = now.strftime("%m/%d/%Y, %H:%M:%S")
# mm/dd/YY H:M:S format
print("time one:", time_one)
time_two = now.strftime("%d/%m/%Y, %H:%M:%S")
# dd/mm/YY H:M:S format
print("time two:", time_two)

from datetime import datetime
date_string = "5 December, 2019"
print("date_string =", date_string)
date_object = datetime.strptime(date_string, "%d %B, %Y")
print("date_object =", date_object)

#Using date from datetime
from datetime import date
d = date(2020, 1, 1)
print(d)
print('Current date:', d.today())    # 2019-12-05
# date object of today's date
today = date.today()
print("Current year:", today.year)   # 2019
print("Current month:", today.month) # 12
print("Current day:", today.day)     # 5

#Time Objects to Represent Time
from datetime import time
# time(hour = 0, minute = 0, second = 0)
a = time()
print("a =", a)
# time(hour, minute and second)
b = time(10, 30, 50)
print("b =", b)
# time(hour, minute and second)
c = time(hour=10, minute=30, second=50)
print("c =", c)
# time(hour, minute, second, microsecond)
d = time(10, 30, 50, 200555)
print("d =", d)

#Difference Between Two Points in Time Using
today = date(year=2019, month=12, day=5)
new_year = date(year=2020, month=1, day=1)
time_left_for_newyear = new_year - today
# Time left for new year:  27 days, 0:00:00
print('Time left for new year: ', time_left_for_newyear)
```

```
t1 = datetime(year = 2019, month = 12, day = 5, hour = 0, minute = 59, second = 0)
t2 = datetime(year = 2020, month = 1, day = 1, hour = 0, minute = 0, second = 0)
diff = t2 - t1
print('Time left for new year:', diff) # Time left for new year: 26 days, 23: 01: 00

#Difference Between Two Points in Time Using timedelata
from datetime import timedelta
t1 = timedelta(weeks=12, days=10, hours=4, seconds=20)
t2 = timedelta(days=7, hours=5, minutes=3, seconds=30)
t3 = t1 - t2
print("t3 =", t3)
```

Output

```
['MAXYEAR', 'MINYEAR', '__all__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__',
'__name__', '__package__', '__spec__', 'date', 'datetime', 'datetime_CAPI', 'sys', 'time', 'timedelta',
'timezone', 'tzinfo']
2023-04-17 04:32:03.543562
17 4 2023 4 32
timestamp 1681705923.543562
17/4/2023, 4:32
2020-01-01 00:00:00
1 1 2020 0 0
1/1/2020, 0:0
time: 04:32:03
time one: 04/17/2023, 04:32:03
time two: 17/04/2023, 04:32:03
date_string = 5 December, 2019
date_object = 2019-12-05 00:00:00
2020-01-01
Current date: 2023-04-17
Current year: 2023
Current month: 4
Current day: 17
a = 00:00:00
b = 10:30:50
c = 10:30:50
d = 10:30:50.200555
Time left for new year:  27 days, 0:00:00
Time left for new year: 26 days, 23:01:00
t3 = 86 days, 22:56:50
```

27. Write a python program to implement file concept

```
#File Handling
#File handling is an import part of programming which allows us to create, read, update and delete files. I
n Python to handle data we use open() built-in function.
# Syntax
'"open('filename', mode) # mode(r, a, w, x, t,b)  could be to read, write, update
"r" - Read - Default value. Opens a file for reading, it returns an error if the file does not exist
"a" - Append - Opens a file for appending, creates the file if it does not exist
"w" - Write - Opens a file for writing, creates the file if it does not exist
"x" - Create - Creates the specified file, returns an error if the file exists
"t" - Text - Default value. Text mode
"b" - Binary - Binary mode (e.g. images)'"
#Opening Files for Reading
#The default mode of open is reading, so we do not have to specify 'r' or 'rt'. I have created and saved a fil
e named myfile.txt in the files directory.

f = open('/content/drive/MyDrive/myfile.txt')
print(f) # <_io.TextIOWrapper name='./files/reading_file_example.txt' mode='r' encoding='UTF-8'>
#As you can see in the example above, I printed the opened file and it gave some information about it.
#Opened file has different reading methods: read(), readline, readlines. An opened file has to be closed wi
th close() method.

#read(): read the whole text as string. If we want to limit the number of characters we want to read, we ca
n limit it by passing int value to the read(number) method.
f = open('/content/drive/MyDrive/myfile.txt')
txt = f.read()
print(type(txt))
print(txt)
f.close()
# output

#Instead of printing all the text, let us print the first 10 characters of the text file.
f = open('/content/drive/MyDrive/myfile.txt')
txt = f.read(10)
print(type(txt))
print(txt)
f.close()

#readline(): read only the first line
f = open('/content/drive/MyDrive/myfile.txt')
line = f.readline()
print(type(line))
print(line)
f.close()
```

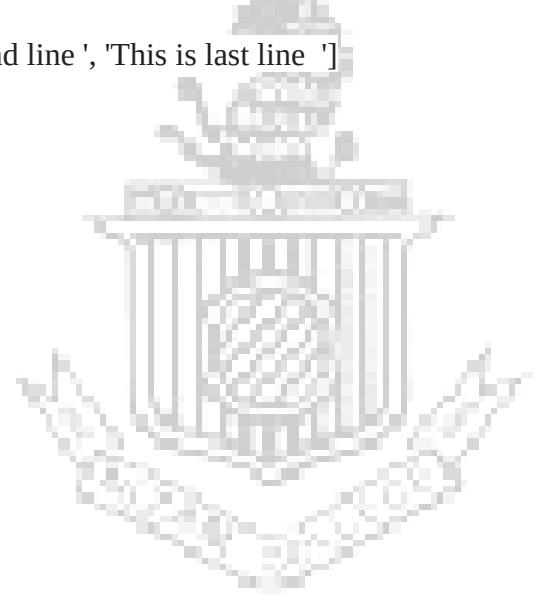
```
#readlines(): read all the text line by line and returns a list of lines
f = open('/content/drive/MyDrive/myfile.txt')
lines = f.readlines()
print(type(lines))
print(lines)
f.close()

#Another way to get all the lines as a list is using splitlines():
f = open('/content/drive/MyDrive/myfile.txt')
lines = f.read().splitlines()
print(type(lines))
print(lines)
f.close()

#After we open a file, we should close it. There is a high tendency of forgetting to close them. There is a new way of opening files using with  closes the files by itself. Let us rewrite the the previous example wit h the with method

with open('/content/drive/MyDrive/myfile.txt') as f:
    lines = f.read().splitlines()
    print(type(lines))
    print(lines)

#Opening Files for Writing and Updating
#To write to an existing file, we must add a mode as parameter to the open() function:
#"a" - append - will append to the end of the file, if the file does not it creates a new file.
#"w" - write - will overwrite any existing content, if the file does not exist it creates.

#Let us append some text to the file we have been reading:
with open('/content/drive/MyDrive/myfile.txt','a') as f:
    f.write('This text has to be appended at the end')
#The method below creates a new file, if the file does not exist:
with open('/content/drive/MyDrive/myfile.txt','w') as f:
    f.write('This text will be written in a newly created file')

#Deleting Files
#if we want to remove a file we use os module.

import os
os.remove('/content/drive/MyDrive/myfile.txt')
#If the file does not exist, the remove method will raise an error, so it is good to use a condition like this:
import os
if os.path.exists('/content/drive/MyDrive/myfile.txt'):
    os.remove('/content/drive/MyDrive/myfile.txt')
else:
    print('The file does not exist')
```

Output

<_io.TextIOWrapper name='/content/drive/MyDrive/myfile.txt' mode='r' encoding='UTF-8'>
<class 'str'>
This is first line
This is second line
This is last line

<class 'str'>
This is fi
<class 'str'>
This is first line

<class 'list'>
['This is first line\n', 'This is second line \n', 'This is last line \n']
<class 'list'>
['This is first line', 'This is second line ', 'This is last line ']
<class 'list'>
['This is first line', 'This is second line ', 'This is last line ']
The file does not exist



28. Write a program to print each line of a file in reverse order

Myfile.txt

This is first line
This is second line
This is last line

Python file 1 – file1.py

```
# Open the file in write mode
f1 = open("output1.txt", "w")


# Open the input file and get # the content into a variable data
with open("myfile.txt", "r") as myfile:
  data = myfile.read()


# For Full Reversing we will store the # value of data into new variable data_1 # in a reverse order using [start: end: step], # where step when passed -1 will reverse # the string
data_1 = data[::-1]


# Now we will write the fully reverse # data in the output1 file using # following command
f1.write(data_1)
print(data_1)
f1.close()
```

Output

enil tsal si sihT
enil dnoces si sihT
enil tsrif si sihT

Python file 2 – file2.py

```
# Open the file in write mode
f2 = open("output2.txt", "w")

# Open the input file again and get # the content as list to a variable data
with open("myfile.txt", "r") as myfile:
   data = myfile.readlines()
```

```
# We will just reverse the # array using following code
data_2 = data[::-1]
# Now we will write the fully reverse # list in the output2 file using # following command
f2.writelines(data_2)
print(data_2)
f2.close()
```

Output

['This is last line', 'This is second line\n', 'This is first line\n']

29. Write a python program on exception handling

```
#Exception Handling
#Python uses try and except to handle errors gracefully. A graceful exit (or graceful handling) of errors is a simple programming idiom -
 a program detects a serious error condition and "exits gracefully", in a controlled manner as a result. Ofte n the program prints a descriptive error message to a terminal or log as part of the graceful exit, this make s our application more robust. The cause of an exception is often external to the program itself. An examp le of exceptions could be an incorrect input, wrong file name, unable to find a file, a malfunctioning IO d evice. Graceful handling of errors prevents our applications from crashing.
#Syntax

'''try:
    code in this block if things go well
except:
    code in this block run if things go wrong'''

#Example
try:
    print(10 + '5')
except:
    print('Something went wrong')

#In the example above the second operand is a string. We could change it to float or int to add it with the number to make it work. But without any changes, the second block, except, will be executed.

#Example
try:
    name = input('Enter your name:')
    year_born = input('Year you were born:')
    age = 2019 - year_born
    print(f'You are {name}. And your age is {age}.')
except:
    print('Something went wrong')

#Something went wrong
#In the above example, the exception block will run and we do not know exactly the problem. To analyze
 the problem, we can use the different error types with except.

#In the following example, it will handle the error and will also tell us the kind of error raised.
try:
    name = input('Enter your name:')
    year_born = input('Year you were born:')
    age = 2019 - year_born
    print(f'You are {name}. And your age is {age}.')
except TypeError:
```

```
    print('Type error occured')
except ValueError:
    print('Value error occured')
except ZeroDivisionError:
    print('zero division error occured')

#Type error occured
#In the code above the output is going to be TypeError. Now, let's add an additional block:

try:
    name = input('Enter your name:')
    year_born = input('Year you born:')
    age = 2019 - int(year_born)
    print('You are {name}. And your age is {age}.',name,age)
except TypeError:
    print('Type error occur')
except ValueError:
    print('Value error occur')
except ZeroDivisionError:
    print('zero division error occur')
else:
    print('I usually run with the try block')
finally:
    print('I alway run.')

#It is also shorten the above code as follows:
try:
    name = input('Enter your name:')
    year_born = input('Year you born:')
    age = 2019 - int(year_born)
    print('You are {name}. And your age is {age}.',name,age)
except Exception as e:
    print(e)
```

Output

```
Something went wrong
Enter your name:john
Year you were born:2003
Something went wrong
Enter your name:john
Year you were born:2003
Type error occured
Enter your name:john
Year you born:2003
You are {name}. And your age is {age}. john 16
I usually run with the try block
```

I alway run.
Enter your name:john
Year you born:2003
You are {name}. And your age is {age}. john 16

30. Demonstrate different kinds of python errors

```
#Python Error Types
#When we write code it is common that we make a typo or some other common error. If our code fails to run, the Python interpreter will display a message, containing feedback with information on where the pr oblem occurs and the type of an error. It will also sometimes gives us suggestions on a possible fix. Unde rstanding different types of errors in programming languages will help us to debug our code quickly and also it makes us better at what we do.
#Let us see the most common error types one by one.

#SyntaxError

print 'hello world'
```

Output

```
'"File "<ipython-input-10-6bf9dfc4c8fe>", line 6
    print 'hello world'
          ^
SyntaxError: Missing parentheses in call to 'print'. Did you mean print('hello world')?"'


#As you can see we made a syntax error because we forgot to enclose the string with parenthesis and Pyt hon already suggests the solution. Let us fix it.
```



```
print('hello world')
```

Output

```
hello world
```

```
#NameError

print(college)
```

Output

```
'"NameError                               Traceback (most recent call last)
<ipython-input-14-73acc31db330> in <cell line: 2>()
      1 #NameError
----> 2 print(college)

NameError: name 'college' is not defined'"

#As you can see from the message above, name age is not defined. Yes, it is true that we did not define a
n age variable but we were trying to print it out as if we had had declared it. Now, lets fix this by declarin
g it and assigning with a value.

#The type of error was a NameError. We debugged the error by defining the variable name.

college = 'PGRRCDE'
print(college)
```

Output

```
PGRRCDE
```

```
#IndexError

numbers = [1, 2, 3, 4, 5]
numbers[5]
```

Output

```
'''IndexError                            Traceback (most recent call last)
<ipython-input-16-efff15c92551> in <cell line: 4>()
      2
      3 numbers = [1, 2, 3, 4, 5]
----> 4 numbers[5]
      5
      6 #In the example above, Python raised an IndexError, because the list has only indexes from 0 to 4 ,
so it was out of range.

IndexError: list index out of range'''

#In the example above, Python raised an IndexError, because the list has only indexes from 0 to 4 , so it
was out of range.
```

```
#ModuleNotFoundError

import maths
```

Output

```
'''---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-17-a668a5fbdb35> in <cell line: 3>()
      1 #ModuleNotFoundError
      2
----> 3 import maths
      4
      5 #In the example above, I added an extra s to math deliberately and ModuleNotFoundError was raise
d. Lets fix it by removing the extra s from math.

ModuleNotFoundError: No module named 'maths'

---------------------------------------------------------------------------
NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.

To view examples of installing some common dependencies, click the
"Open Examples" button below.'''

#In the example above, I added an extra s to math deliberately and ModuleNotFoundError was raised. Let
s fix it by removing the extra s from math.

#We fixed it, so let's use some of the functions from the math module.

import math
```

Output
Not Applicable

```
#AttributeError

import math
math.PI
```

Output

```
'"---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-20-93b9df399274> in <cell line: 4>()
      2
      3 import math
----> 4 math.PI
      5
      6

AttributeError: module 'math' has no attribute 'PI""

#As you can see, I made a mistake again! Instead of pi, I tried to call a PI function from maths module. It raised an attribute error, it means, that the function does not exist in the module. Lets fix it by changing fr om PI to pi.

#Now, when we call pi from the math module we got the result.

math.pi
```

Output

```
3.141592653589793
```

```
#KeyError

users = {'name':'Asab', 'age':250, 'country':'Finland'}

print(users['name'])

print(users['county'])
```

Output

```
"'Asab
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-23-7110f943dfa8> in <cell line: 6>()
      4 print(users['name'])
      5
----> 6 print(users['county'])
      7
      8

KeyError: 'county""

#As you can see, there was a typo in the key used to get the dictionary value. so, this is a key error and th
e fix is quite straight forward. Let's do this!

#We debugged the error, our code ran and we got the value.

users['country']
```

Output

```
Finland
```

```
#TypeError

 4 + '3'
```

Output

```
 '"File "<ipython-input-26-ddeecc2937b2>", line 3
    4 + '3'
    ^
IndentationError: unexpected indent"'
```

```
 #In the example above, a TypeError is raised because we cannot add a number to a string. First solution would be to convert the string to int or float. Another solution would be converting the number to a string (the result then would be '43'). Let us follow the first fix.

 #Error removed and we got the result we expected.

print(4 + int('3'))

print(4 + float('3'))
```

Output

```
7
7.0
```

```
#ImportError

from math import power
```

Output

```
'"---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
<ipython-input-28-bb147d12493e> in <cell line: 3>()
      1 #ImportError
      2
----> 3 from math import power
      4
      5

ImportError: cannot import name 'power' from 'math' (unknown location)

---------------------------------------------------------------------------
NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.

To view examples of installing some common dependencies, click the
"Open Examples" button below.'"

#There is no function called power in the math module, it goes with a different name: pow. Let's correct i
t:
```



```
from math import pow

pow(2,3)
```


Output

```
8.0
```

```
#ValueError

int('12a')
```

Output

```
'"---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-30-b4bde0beee85> in <cell line: 3>()
      1 #ValueError
      2
----> 3 int('12a')
      4
      5 #In this case we cannot change the given string to a number, because of the 'a' letter in it.

ValueError: invalid literal for int() with base 10: '12a'"'

#In this case we cannot change the given string to a number, because of the 'a' letter in it.
```

```
#ZeroDivisionError

1/0
```

Output

```
'''---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-31-30376fdc2170> in <cell line: 3>()
      1 #ZeroDivisionError
      2
----> 3 1/0
      4
      5 #We cannot divide a number by zero.

ZeroDivisionError: division by zero'''

#We cannot divide a number by zero.
```