



PYTHON PROGRAMMING

Basic of Python:

1. Variables declaration

For integer: int.
For float: float.
For string: str.

Note: Python automatically initialize the type of variables but when we ask input from user then it only take the input in string format so by using type casting we convert the string according to our requirement.

The screenshot shows a PyCharm interface. The code editor contains the following Python script:

```
1 a=10
2 b=20.89
3 c="samundar"
4 print(type(a))
5 print(type(b))
6 print(type(c))
```

The run output window shows the execution results:

```
C:\Users\samun\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/samun/.PyCharmCE2019.3/config/scratches/scratch.py
<class 'int'>
<class 'float'>
<class 'str'>

Process finished with exit code 0
```

A yellow callout box points to the line `print(type(a))` with the text "Type function : this will help to find the type of the variable".

2. Taking input from user and perform typecasting

- The `input()` function is used to take input from the user. By default, it takes string as the input so if we want to take integer or float then we used the typecasting for that

The screenshot shows a PyCharm interface. The code editor contains the following Python script:

```
1 # traditional way of taking input
2 print("ENTER THE VALUE OF A ")
3 a=input()
4 print(a)
5 print(type(a))
6 a=int(a)
7 print(type(a))

9 # Standard way of taking input from the user
10 b=int(input("ENTER THE VALUE OF B"))
11 print(b)
12 print(type(b))
```

The run output window shows the execution results:

```
ENTER THE VALUE OF A
12
12
<class 'str'>
<class 'int'>
ENTER THE VALUE OF B10
10
<class 'int'>
```

A yellow callout box points to the line `b=int(input("ENTER THE VALUE OF B"))` with the text "Type casting performed in one line".



PYTHON PROGRAMMING

3.List in python

- It is a collection of the data that may be of different types. It will act as an array but the major difference is that an array contains group of similar data type whereas a list contains group of similar as well as different data types.
- The data items present between the **big braces []** and we can access the element using the indices number.
- List are **mutable** (can change).

Methods:

creating a empty list say **number= []** (creating a empty list)

Here we are creating a list say **number= [2,7,9,11,3]** (when data is assigned at same time)

- **number. Sort()** - it will sort the number.
- **number. reverse()** - it will reverse the list .
- **max(number)** - it will find the maximum number from the list.
- **min(number)** - it will find the minimum number from the list.
- **len(number)** - it will find the length of the list.
-

To alter the data of the list:

- **number append(data)** - it will insert data in list(at the last) .
- **number pop(data)** - it will remove data from the list(at the last) .
- **number. insert(index, data)** - it will insert data in list to the index which is mentioned.
- **number. remove(data)** - it will delete data from list wherever the data is.

```
File Edit View Navigate Code Befactor Run Tools VCS Window Help rough [C:\Users\samun\Desktop\rough] - C:\Users\samun\PyCharmCE2019.3\config\scratches\scratch.py - PyCharm
```

```
Project: scratch
```

```
scratch.py
```

```
1 num=[2,5,7,11,1,6]      # creating a list and assigning the data to the list
2 print(num)
3 num.sort()                #perform sorting
4 print(num)
5 num.reverse()              #perform reverse
6 print(num)
7 num.append(100)            #perform append
8 print(num)
9 num.pop()                  #perform pop
10 print(num)
11 num.insert(1,"A")         #perform insert
12 print(num)
13 num.remove(11)            #perform delete
14 print(num)
```

```
Run: scratch
```

```
C:\Users\samun\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/samun/.PyCharmCE2019.3/config/scratches/scratch.py
```

```
[2, 5, 7, 11, 1, 6]
[1, 2, 5, 6, 7, 11]
[11, 7, 6, 5, 2, 1]
[11, 7, 6, 5, 2, 1, 100]
[11, 7, 6, 5, 2, 1]
[11, 'A', 7, 6, 5, 2, 1]
['A', 7, 6, 5, 2, 1]
```

```
Event Log
```

```
15:1 CRLF UTF-8 4 spaces
```

4.tuple in python:

- It is a collection of data of different type same as list but it can't have same method as the list have.
- It is immutable in nature means its data can't be altered.

creating a empty list say **number= ()** (creating a empty tuple)

Here we are creating a list say **number= (2,7,9,11,3)** (when data is assigned at same time)



5.Dictionary in python

- It is a data type in python which contains a key -value pair to store the data.
 - The data can be accessed by the help of the key.
 - It is mutable in nature.

Methods:

creating a empty dictionary say food= {} (creating a empty dictionary)

here we are creating a dictionary food={"Bengal": "fish", "Bihar": "Litti-chokka", "Chennai": "idli-dosa"}

To alter the data of the Dictionary:

- **food.update({key:value})** - it will insert data in dictionary .
 - **del food[key]** - it will delete data from dictionary.

 - **food.get(key)** - it will fetch the value of key from dictionary.
 - **food.keys()** - it will fetch the keys of dictionary.
 - **food.values()** - it will fetch the value of key from dictionary.
 - **food.items()** - it will fetch the key-value pair from dictionary.

```
File Edit View Navigate Code Befactor Run Tools VCS Window Help rough [C:\Users\samun\Desktop\rough] - C:\Users\samun\PyCharmCE2019.3\config\scratches\scratch.py - PyCharm
Scratches > scratch.py
scratch.py X
1 food={'bengal':'fish','bihar':'litti-chokha','chennai':'ildi-dosa'}      #this will create the dictionary
2 print(food)
3 food.update({"usa":"pizza"})          #this will add data to the dictionary
4 print(food)
5 del food['bihar']                   #this will delete data from dictionary
6 print(food)
7 print("\n")
8 print(food.keys())                  #to get keys of dictionary
9 print(food.values())                #to get values of dictionary
10 print(food.items())                 #to get keys-values pair of dictionary
11 |
```

Run scratch ×
C:\Users\samun\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/samun/.PyCharmCE2019.3/config/scratches/scratch.py
{'bengal': 'fish', 'bihar': 'litti-chokha', 'chennai': 'ildi-dosa'}
{'bengal': 'fish', 'bihar': 'litti-chokha', 'chennai': 'ildi-dosa', 'usa': 'pizza'}
{'bengal': 'fish', 'chennai': 'ildi-dosa', 'usa': 'pizza'}

dict_keys(['bengal', 'chennai', 'usa'])
dict_values(['fish', 'ildi-dosa', 'pizza'])
dict_items([('bengal', 'fish'), ('chennai', 'ildi-dosa'), ('usa', 'pizza')])

Note: There is a special method called **copy()** use to copy one dictionary and store its value to another we require to use **copy()** function because when we copy directly one dictionary to other dictionary then if we want some changes in other dictionary then it will also reflect to that dictionary from where the data had been copied



PYTHON PROGRAMMING

6.Exception handling in python

There are some situation in which the program will throw an error or not based on the user input. If the program runs well then there is no problem but if the program has some error then the execution gets stopped at that place where error occurs. So avoid such situations and run our program smoothly we use exception handling.

First benefit of using exception handling is that we know where the error occurs and second benefit is that our program doesn't stop while error encounter.

Types of exception:

1.Built-in exception: - These are the exceptions which are already available in python language.

2.User defined exception: - A programmer can create his own exception called user-defined exception.

Some points to remember:

- 1.we can write several except blocks for a single try block.
- 2.we can write multiple except blocks to handle multiple exceptions.
- 3.we can write try block without any except block.
- 4.but we can't write except block without any try block.
- 5.finally block will always be executed whether there is an exception or not.
- 6.Else and finally block is optional.

Syntax:

```
try:  
    statement.....  
except Exception class name:  
    statement.....  
  
else:  
    statement.....  
finally:  
    statement.....
```

How to write the Exception:

Built-in Exception:

First we have to find what type of possible errors are going to occur in the program according to which we have to implement the handling procedure. Here some of the exceptions are mentioned:

1. **type error:** This type of error occurs when we are performing some operation between two different types of variables (e.g. addition of an integer and a character)

<pre>input: try: a=int(input("please enter a number : ")) b=input("enter another number : ") c=a+b except TypeError as e: print(e) print("success")</pre>	Output please enter a number :12 enter another number : rt unsupported operand type(s) for +: 'int' and 'str' success
---	--

2. **Zero division error:** This error occurs when we perform the division of a number by zero

<pre>input: a=int(input("please enter a number")) b=int(input("please enter a number")) try: a/b except ZeroDivisionError as e: print(e) print("success")</pre>	Output please enter a number :12 enter another number : rt unsupported operand type(s) for +: 'int' and 'str' success
---	--



PYTHON PROGRAMMING

3. **value error:** this error occur a function receive an argument that has a correct type but value provide to it by user is incorrect (e.g. when user try to insert a string when the input takes only int value)

input:	Output
<pre>while(1): try: a=int(input("please enter a number")) b=int(input("enter another number")) break except ValueError as e: print(e) print("sum is : ",a+b)</pre>	<pre>please enter a number: we invalid literal for int() with base 10: 'we' please enter a number12 enter another number34 sum is : 46</pre>

4. **Keyboard interrupt:** suppose if we do not want to execute a particular piece of code than by help of the keyboardinterrupt we can skip that code.

input:	Output
<pre>try: inp=input("enter anything") print("this statement run if there is no keyboard interrupt") except KeyboardInterrupt: print("keybord interrupt occur") print("SUCCESS")</pre>	<pre>enter anything #after pressing ctrl+c keyboard interrupt occur SUCCESS</pre>

User defined Exception:

There are 3 steps to create a user defined exception:

1. Creating exception class
2. Raising exception
3. Handling exception

input:	Output
<pre>class MyException(Exception): pass def checkbalance(): money=10000 print("you have ",money," in your account") withdrawl=int(input("enter the withdrawal amount : ")) global balance balance=money-withdrawl if(balance<2000): raise MyException("balance insufficient") try: checkbalance() print("transaction sucessfull available balance : ",balance) except MyException as e: print(e) print("this is going to work every time to show the exception handling flow")</pre>	<pre>you have 10000 in your account enter the withdrawal amount : 9000 balance insufficient this is going to work every time to show the exception handling flow</pre>



PYTHON PROGRAMMING

7.File handling in python

Python help us to read and write or create a file. The various Acess Modes are:

- “r” : opens a file for read only.
- “w” : opens a file for write only.
- “r+” : opens a file for both read and write. Does not overwrite (doesn’t delete data when executed)
- “w+”: opens a file for both read and write overwrite the file if file exist otherwise creates a new file.
- “a” : it is append mode used to only append any data at the last of file not for read.
- “a+”: it will append data and also allow to read the data.

The open() function:

To open any file we use open() function.

Syntax:

```
fileobj=open("file name", "access mode ")
```

NOTE: Here we create a object which helps us in using the various method related to file handling

Method for reading and writing file:

- 1.) **read()** Method : This method is used to read a string from a open file
syntax: content=fileobject.read()
- 2.) **write()**Method :this method is used to write string to an open file.The access mode must be in write mode.
syntax: fileobject.write(string)
- 3.) **Close()**method :it will close the file after which the data cannot be written to the opened file
syntax: fileobject.close()

NOTE: We will perform iteration and able to access each word from the text file and later we will modify it

input:

```
fileobj=open('sam.txt',"r+")
content=fileobj.read()           #it will create a file object from which other method is called
print(content)
fileobj.write("\n hello samundar how are you ?")
fileobj.close()

#again we are accessing the file

print("-----again Access the same file -----")
obj2=open("sam.txt","r+")
content2=obj2.read()
print(content2)
```

Output:

```
C:\Users\samun\AppData\Local\Programs\Python\Python38-32\python.exe
C:/Users/samun/.PyCharmCE2019.3/config/scratches/scratch.py
hello my name is samundar singh
i am a good boy
i am from jharkhand

-----again Access the same file -----
hello my name is samundar singh
i am a good boy
i am from jharkhand
```

```
hello samundar how are you ?
```

```
Process finished with exit code 0
```



PYTHON PROGRAMMING

8.Recursion in python

When a function call itself then this process called a recursion. since the function call itself so there are infinite calling if we do not stop the calling of function so there must be some way which interrupt the calling of function. This interrupt logic is called **Base case**. Generally in base case there is a **if statement** which return some value and thus the calling breaks.

Syntax:

```
def function(argument):
    if(argument==0):
        return 1
    else:
        function(logic) ← Here we write the logic which call the function recursively

function() ← Here the function called
```

Function defined Base case Here we write the logic which call the function recursively Here the function called

For better understanding a example to show the implementation is shown here:

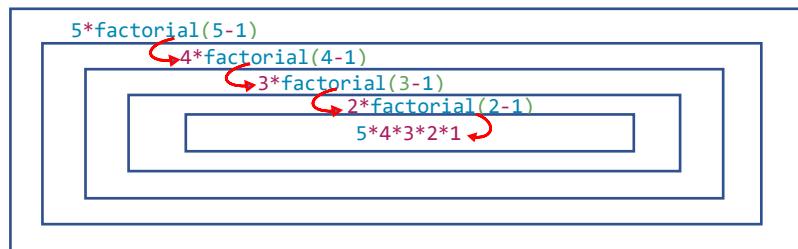
Example for finding factorial of a number using recursion

```
input
def factorial(n):
    if(n==1):
        return 1
    else:
        return n*factorial(n-1)

n=int(input("enter the number : "))
result=factorial(n)
print("factorial of ",n," is ",result)
```

Output

enter the number : 5
factorial of 5 is 120



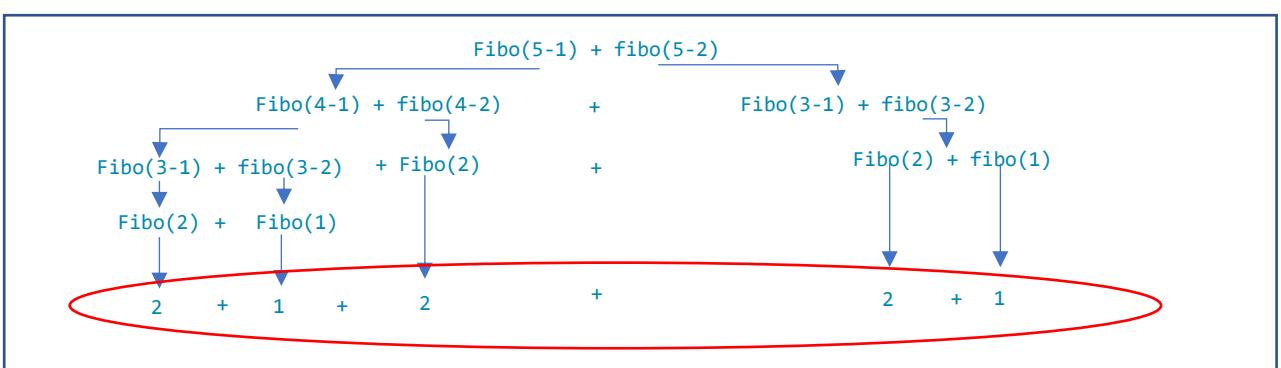
Example for finding Fibonacci series using recursion

```
input
def fibo(n):
    if(n==1):
        return 1
    elif(n==2):
        return 2
    else:
        return fibo(n-1)+fibo(n-2)

n=int(input("enter the number : "))
```

Output

enter the number : 5
fibonacci of 5 is 8





PYTHON PROGRAMMING

9.F-string in python

- This function is used when we have to insert variable in a string specially in between coding
- Before declaring a sting we have to write "f" before the quotes.
- This f string is useful in data insert and retrieve from data base

```
input  
a=(input("enter name :"))  
b=int(input("enter age : "))  
c=input("enter the address : ")  
  
print(f"my name is {a} and i am {b} years old. i live in {c}")  
  
print('my name is %s and i am %d years old. i live in %s'%(a,b,c))
```

Using f sting
Normal method

10.passing multiple argument in function

- We use *var to take multiple argument in function.
- The *var store the multiple data inside the function as a tuple of data.
- Always write the *var as the last argument of the function.
- We can pass data in function in any type like list, tuple, multiple declare variables etc.
- We can pass the dictionary also in the function by using ** var2.
- The *var is always written before the **var2.

Example : passing multiple argument :

input	Output
#creating a list of student def student(*data1): print('the student are :') for i in data1: print(i,end=" ") student("samundar","rahul","rakul")	the student are : samundar rahul rakul

11.The Enumerate function

There are some situation in which we need the index value as well as the data of a given list ,in such situation we have to use two loop one for the index and another for the value in order to skip the loop creation we use enumerate function it will return the index as well as value.

input	Output
li=["apple","mango","banana","litchi"] for index,value in enumerate(li): print(f"The index is {index} and the value is {value}")	The index is 0 and the value is litchi The index is 1 and the value is banana The index is 2 and the value is mango The index is 3 and the value is apple

NOTE: In enumerate function the index is recorded from the end it means that the end element has index 0 and the first element has indexed last



PYTHON PROGRAMMING

12. Modules in python

We can create our own modules (user-defined module) in python.

- First, we create a file name with the extension.
- Then we write “**import file name**” in file name mention the name of file from where function is imported
- we can access the function of the another file by using (.dot e.g. module name. function name()).

This is the file where module is defined:

```
File Edit View Navigate Code Befactor Run Tools VCS Window Help rough [C:\Users\samun\Desktop\rough] - C:\Users\samun\PyCharmCE2019.3\config\scratches\sam.py - PyCharm
Scratches > sam.py
Project scratch.py sam.py
1 def add(x,y):
2     return x+y
3 def sub(x,y):
4     return x-y
5 def mul(x,y):
6     return x*y
7
```

This is the file where we accessed the module:

```
File Edit View Navigate Code Befactor Run Tools VCS Window Help rough [C:\Users\samun\Desktop\rough] - C:\Users\samun\PyCharmCE2019.3\config\scratches\scratch.py - PyCharm
Scratches > scratch.py sam.py
Project scratch.py sam.py
1 import sam
2 a=int(input("enter a number : "))
3 b=int(input("enter another number : "))
4
5 print(f"addition of {a} and {b} is {sam.add(a,b)}")
6 print(f"subtraction of {a} and {b} is {sam.sub(a,b)}")
7
```

Run: scratch > sam

```
C:/Users/samun/AppData/Local/Programs/Python/Python38-32/python.exe C:/Users/samun/.PyCharmCE2019.3/config/scratches/scratch.py
enter a number : 12
enter another number : 8
addition of 12 and 8 is 20
subtraction of 12 and 8 is 4

Process finished with exit code 0
```

Terminal Python Console Run TODO Event Log

**V.imp

NOTE: There are situations that if there is another code except the function in the file then whenever we import the file then the program will execute all the lines of imported file first but we do not want that we only want to use a particular function so here we use this:

- Simply type main in pycharm and press enter it will write this code: `if __name__ == '__main__'`
- we have to write code inside this block in order to avoid the above problem.
- This sentence will just an if block which executes only when you are in the same file in which you are executing the program, if you are in another file then the code that is written inside these main block will not be executed.



PYTHON PROGRAMMING

13.The Join function in python

There are various scenario in which we use the join function suppose we fetch data from data base and try to make in such a format which can be easily loaded in Hadoop and other database.in such case we can modify our fetched data and made it into a simple format.

The screenshot shows the PyCharm IDE interface. In the top navigation bar, the file 'scratch.py' is selected. The code in the editor is:

```
1 li=['apple','mango','banana','chickoo','orange','litchi']
2 print(li)
3 a=' @ '.join(li)
4 print(a)
5
```

In the 'Run' tab, the output window shows the execution of the script:

```
C:\Users\samun\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/samun/.PyCharmCE2019.3/config/scratches/scratch.py
['apple', 'mango', 'banana', 'chickoo', 'orange', 'litchi']
apple @ mango @ banana @ chickoo @ orange @ litchi

Process finished with exit code 0
```

13.The map function in python

If there is a list and we want to modify each and every data of list then this can be done by map function.

Syntax: `var = list(map(fun,list_name))`

!warning: do not put braces here

- First, we have to create a function which changes the data of the function note the function must take argument and also return the result.
- The map function put each data in that function and whatever the function return data stored in another list.
- The map function always returns object data type so we have to typecast it into the list format.
- The result must be stored in another variable (or list).it means it create a separate list with modified data.

The screenshot shows the PyCharm IDE interface. In the top navigation bar, the file 'scratch.py' is selected. The code in the editor is:

```
1 def square(x):
2     return x*x
3 def cube(x):
4     return x*x*x
5
6
7 li=[1,2,3,4,5,6,7,8,9,10]
8 print('the normal list is : ',li)
9 a=list(map(square,li))
10 b=list(map(cube,li))
11 print("the square of list is: ",a)
12 print("the cube of list is: ",b)
13
```

In the 'Run' tab, the output window shows the execution of the script:

```
C:\Users\samun\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/samun/.PyCharmCE2019.3/config/scratches/scratch.py
the normal list is : [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
the square of list is: [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
the cube of list is: [1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]

Process finished with exit code 0
```



13.The filter function in python

If there is a list and we want some filtered data from it then we use filter function. Except of returning value this function return true or false, so it can only filter the data not modified it.

- First, we have to create a function which Checks the data of the function, note the function must take argument and also return the result in form of true or false.
- The filter function put each data from the list and send it to the function where if the function return true then only it keep the data and stored it into a new list.
- It cannot modify the data of the list

The screenshot shows the PyCharm IDE interface. In the top navigation bar, the file 'scratch.py' is selected. The code editor contains the following Python script:

```
1 def great(x):
2     return(x>5)
3
4 li=[1,2,3,4,5,6,7,8,9,10]
5 print('the given list is : ',li)
6 a=list(filter(great,li))
7 print('the filtered list is : ',a)
```

In the bottom right corner of the code editor, the status bar shows the path: rough [C:\Users\samun\Desktop\rough] - C:\Users\samun\PyCharmCE2019.3\config\scratches\scratch.py - PyCharm. Below the code editor is the Run tool window. The 'Run' tab is selected, and the command listed is: C:\Users\samun\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/samun/.PyCharmCE2019.3/config/scratches/scratch.py. The output pane displays the following text:

```
the given list is : [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
the filtered list is : [6, 7, 8, 9, 10]

Process finished with exit code 0
```

The status bar at the bottom right indicates the current time as 7:21, encoding as CRLF, character set as UTF-8, and indentation as 4 spaces.



PYTHON PROGRAMMING

OOPS in Python:

1. Class in python

Creating a class in python helps in reusability of code. we can also define function inside the class and call it whenever needed.

- In class we can mention the structure or blueprint of the object and also help to set default value
- In class we can also create a number of functions which can be accessed by creating an object and calling that particular function
- The data that is defined as a default inside the class cannot be changed using the object, it can only be changed when we use class name
- If we add anything in the class using object or try to change then the changes only occur for that object only it cannot throw any error it just get added to that object.
- If we define any function inside the class the as a default parameter it pass a parameter **self** which helps to call the function by passing argument as the self object as show in this fig at line(5). We did not have to pass the object name as argument it automatically passed the object.

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help rough [C:\Users\samun\Desktop\rough] - C:\Users\samun\PyCharmCE2019.3\config\scratches\scratch.py - PyCharm
Scratches > scratch.py sam.py
Project
scratch.py sam.py
1 class student:
2     roll=0
3     name=""
4     std=0
5     def detail(self):
6         print(f"Roll.no :{self.roll}\nName :{self.name}\nclass :{self.std} ")
7
8 sam=student() # creating object
9 rahul=student() # creating object
10
11 sam.name="samundar singh"
12 sam.roll=100
13 sam.std=10
14 print(sam.__dict__) #inbuilt function that print the data in a dictionary format
15 print("\n")
16 sam.detail()

Run: scratch > sam
C:\Users\samun\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/samun/.PyCharmCE2019.3/config/scratches/scratch.py
{'name': 'samundar singh', 'roll': 100, 'std': 10}

    Roll.no :100
    Name :samundar singh
    class :10

Process finished with exit code 0
Event Log
17:1 CRLF UTF-8 4 spaces

```

Accessing the function inside of the class:

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help rough [C:\Users\samun\Desktop\rough] - C:\Users\samun\PyCharmCE2019.3\config\scratches\sam.py - PyCharm
Scratches > scratch.py sam.py
Project
scratch.py sam.py
1 class cal:
2     def add(x,y):
3         return x+y
4     def sub(x,y):
5         return x-y
6
7 a=int(input("enter a number : "))
8 b=int(input("enter another number : "))
9
10 print(f"sum of number {cal.add(a,b)}") # direct access the function of class without object
11 print(f"subtract of number {cal.sub(a,b)}") # direct access the function of class without object

Run: scratch > sam
C:\Users\samun\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/samun/.PyCharmCE2019.3/config/scratches/sam.py
enter a number : 12
enter another number : 5
sum of number 17
subtract of number 7

Process finished with exit code 0
Event Log
8:1 CRLF UTF-8 4 spaces

```



PYTHON PROGRAMMING

2. Constructor in python

A constructor is a special type of class method or function which is used to initialize the member of the class when an object of class is created.

- it is automatically called whenever an object is created.

Syntax: `def __init__(self, argument1, argument2.....)`

The screenshot shows two panels of PyCharm. The left panel displays the code for `const.py`:

```

1 class employee:
2     #creating constructor that initialize the data
3     def __init__(self):
4         id=int(input("enter your employee id : "))
5         name=input("enter your name : ")
6         address=input("enter your address : ")
7         phone=int(input("enter your phone : "))
8
9         self.name=name
10        self.id=id
11        self.address=address
12        self.phone= phone
13
14    #data insertion through creating single object
15    ram=employee()
16    print(ram.__dict__)
17
18    #data insertion by creating object through loop
19    for i in range(1,4):
20        print(f"\t employee : {i} through loop ")
21        i=employee()
22        print(i.__dict__)

```

The right panel shows the run output:

```

C:\Users\samun\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/samun/PyCharmCE2019.3/config/scratches/const.py
enter your employee id : 100
enter your name : samundar
enter your address : Lohardaga
enter your phone : 89898989898
{'name': 'samundar', 'id': 100, 'address': 'lohardaga', 'phone': 89898989898}
    employee : 1 through loop
enter your employee id : 101
enter your name : rahul
enter your address : patna
enter your phone : 4567
{'name': 'rahul', 'id': 101, 'address': 'patna', 'phone': 4567}
    employee : 2 through loop
enter your employee id : 110
enter your name : johnny dep
enter your address : usa
enter your phone : 4564323
{'name': 'johnny dep', 'id': 110, 'address': 'usa', 'phone': 4564323}
    employee : 3 through loop
enter your employee id :

```

3. Changing class member data or property using object in python

The traditional way is that if we have to change the data of class then we can only change by means of class name but now by using `@classmethod` property we can change data by object also.

Syntax: `@classmethod`

Mention object name → objectname.function(cls, newdata)

Mention function name that is written below the classmethod → By default this created

The screenshot shows PyCharm with the code for `class_method1.py`:

```

1 class employee:
2     basic=5000
3     HRA=1000
4     def __init__(self):
5         self.basic=employee.basic
6         self.HRA=employee.HRA
7     @classmethod
8     def changebasic(cls,newbasic):
9         cls.basic=newbasic
10
11    ram=employee()
12    print("ram salary detail : ",ram.__dict__)
13    print(f"\t\t\t\t the basic_pay in employee class before change is {employee.basic}")
14
15    ram.changebasic(10000)      #using object we will change the data of the class successfully
16
17    mohan=employee()
18    print("mohan salary detail : ",mohan.__dict__)
19    print(f"\t\t\t\t the basic_pay in employee class After change is {employee.basic}")

```

The run output shows the changes:

```

C:\Users\samun\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/samun/PyCharmCE2019.3/config/scratches/class_method1.py
ram salary detail : {'basic': 5000, 'HRA': 1000}
    the basic_pay in employee class before change is 5000
mohan salary detail : {'basic': 10000, 'HRA': 1000}
    the basic_pay in employee class After change is 10000

```



PYTHON PROGRAMMING

4. Use of static method in python

There are such situations in which we have to create a function inside the class but did not want the ***self*** argument although we can create the same function by neglecting the ***self*** argument but this is not efficient so we create a static method. The function that is defined below this method does not need any argument.

The screenshot shows a PyCharm IDE interface with the following details:

- File Menu:** File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help.
- Project Bar:** Scratches > static method.py
- Code Editor:** Displays the following Python code:

```
1 class student:
2     def __init__(self):
3         self.name=input("enter name of student : ")
4         self.stream=input("enter the stream : ")
5
6     @staticmethod
7     def fun1():
8         print("this is static method")
9
10    def fun2():
11        print(" this is a normal method")
12
13 for i in range(1,3):
14     i=student()
15     i.fun1()
16     i.fun2()
```
- Run Tab:** Shows the command: C:\Users\samun\AppData\Local\Programs\Python\Python38-32\python.exe "C:/Users/samun/.PyCharmCE2019.3/config/scratches/static method.py". The output window displays:

```
enter name of student : sam
enter the stream : mca
this is static method
Traceback (most recent call last):
  File "C:/Users/samun/.PyCharmCE2019.3/config/scratches/static method.py", line 16, in <module>
    i.fun2()
TypeError: fun2() takes 0 positional arguments but 1 was given
```

5. Access specifier in python

- **Public:** The data that is public can access by any anyone. There is no particular indication for it.
 - **Protected:** The data that is protected can be accessed by the class objects and its child class. It is denoted by a single underscore(_ func1) and called by placing the same single underscore(_ func1).
 - **Private:** The data that is private can only accessed by the class members or child class under some conditions. It is denoted by a double underscore(__func1) and called by placing a single underscore class name double underscore function or variable name(A__func1).

The screenshot shows the PyCharm IDE interface with the following details:

- Project:** python sample program
- File:** access_specifier.py
- Code Content:**

```
1 class A:  
2     var1="this is variable of A"  
3     @staticmethod  
4     def _displayA():          #here we create a private function  
5         print("I am in A")  
6 class B(A):  
7     var2 = "this is variable of B"  
8  
9     def _displayB(self):      #here we create a protected function  
10        print("I am in B")  
11 class C(B):  
12     var3 = "this is variable of C"  
13  
14     def displayC(self):  
15         print("I am in C")    #here we create a public function  
16  
17 obj=C()  
18 obj._A__displayA()           #----- Name mangling performed here-----  
19 obj._displayB()  
20 obj.displayC()
```
- Run Output:**

```
I am in A  
I am in B  
I am in C
```

Process finished with exit code 0



PYTHON PROGRAMMING

6. Inheritance in python

We can inherit the property of one class to another class using inheritance.

Inheritance are of many types:



1. Single level Inheritance:

- In single level inheritance there is one parent class and one child
- The child can inherit with only one parent the child may have child but there is no grandparent relation.
- We can also create the constructor of both the child and parent class and also access the function of the parent class with the help of child class object

Example to show single level inheritance:

```
class personaldetail:  
    def __init__(self):  
        print("1. calling the parent class constructor")  
        self.name=input("enter your name : ")  
        self.age=input("enter your age : ")  
        self.address=input("enter your address : ")  
  
    def family(self):  
        self.father=input("enter father name : ")  
        self.mother=input("enter mother name : ")  
  
    def gender(self):  
        self.gender=input("enter the gender : ")  
  
class studentdetail(personaldetail):  
    def __init__(self):  
        personaldetail.__init__(self)      #this line will call the constructor of the parent class  
        print("2. calling child class constructor")  
        self.stream =input("enter stream : ")  
        self.college=input("enter college name : ")  
        print("3. calling function defined in parent class in child class")  
        self.gender()                   # here we are accessing the parent class function using child self  
  
i=studentdetail()  
print("4. calling function of the parent class using child class object ") # the object of child class acess the function of the parent class  
i.family()  
print(i.__dict__)
```

Output:

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help rough [C:\Users\samun\Desktop\rough] - C:\Users\samun\PyCharmCE2019.3\config\scratches\inheritance_.py - PyCharm  
Scratches > inheritance_.py  
Project: inheritance_.py  
scratch.py const.py class_method1.py static method.py inheritance_.py  
Run inheritance_.py  
C:\Users\samun\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/samun/.PyCharmCE2019.3/config/scratches/inheritance_.py  
1. calling the parent class constructor  
enter your name : Abhishek bacchan  
enter your age : 34  
enter your address : mumbai  
2. calling child class constructor  
enter stream : MCA  
enter college name : Anna university  
3. calling function defined in parent class in child class  
enter the gender : Male  
4. calling function of the parent class using child class object  
enter father name : Amitabh bacchan  
enter mother name : jaya bacchan  
{'name': 'Abhishek bacchan', 'age': '34', 'address': 'mumbai', 'stream': 'MCA', 'college': 'Anna university', 'gender': 'Male', 'father': 'Amitabh bacchan', 'mother': 'jaya bacchan'}  
Process finished with exit code 0
```

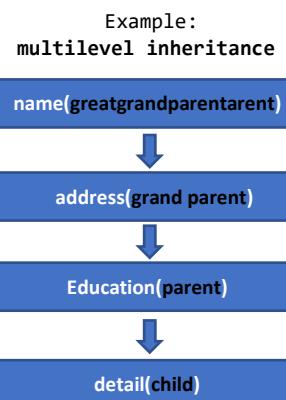
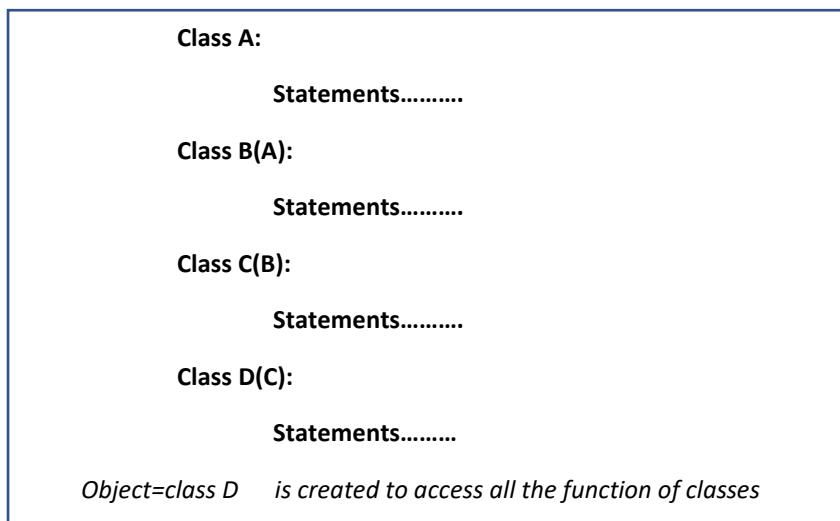


PYTHON PROGRAMMING

2). Multi-level Inheritance:

- In multilevel inheritance we can access the grand parent data using child class object.
- In multilevel there is more than two class and the class are interconnected to each other.
- We can either access the parent class data in child class or we can call the function of the parent class by help of the child class object.

Syntax:



Example to show Multi level inheritance:

```

File Edit View Navigate Code Behavior Run Tools VCS Window Help python sample program [D:\python program\python learning\python sample program] - ...\\inheritance_multilevel.py - PyCharm
python sample program inheritance_multilevel.py
Project inheritance_multilevel.py
1 ❶ class name:
2     def names(self):
3         self.firstname=input("enter first your name : ")
4         self.lastname=input("enter your last name : ")
5 ❷ class address(name):
6     def address(self):
7         self.address=input("enter your address : ")
8
9 ❸ class education(address):
10    def educationdetail(self):
11        self.stream=input("enter your stream : ")
12
13 class detail(education):
14     def __init__(self):
15         self.names()          #accessing a function of great grand parent using grandchild object
16
17
18     def printdata(self):
19         print(self.__dict__)
20
21 ram=detail()
22 ram.printdata()           #Accessing the child class function
23 ram.address()            #accessing grand parent function using child object
24 ram.printdata()
  
```

Output:

```

File Edit View Navigate Code Behavior Run Tools VCS Window Help python sample program [D:\python program\python learning\python sample program] - ...\\inheritance_multilevel.py - PyCharm
python sample program inheritance_multilevel.py
Run: inheritance_multilevel
C:\Users\sumun\AppData\Local\Programs\Python\Python38-32\python.exe "D:/python program/python learning/python sample program/inheritance_multilevel.py"
enter first your name : samundar
enter your last name : singh
{'firstname': 'samundar', 'lastname': 'singh'}
enter your address : Jharkhand
{'firstname': 'samundar', 'lastname': 'singh', 'address': 'Jharkhand'}

Process finished with exit code 0
  
```

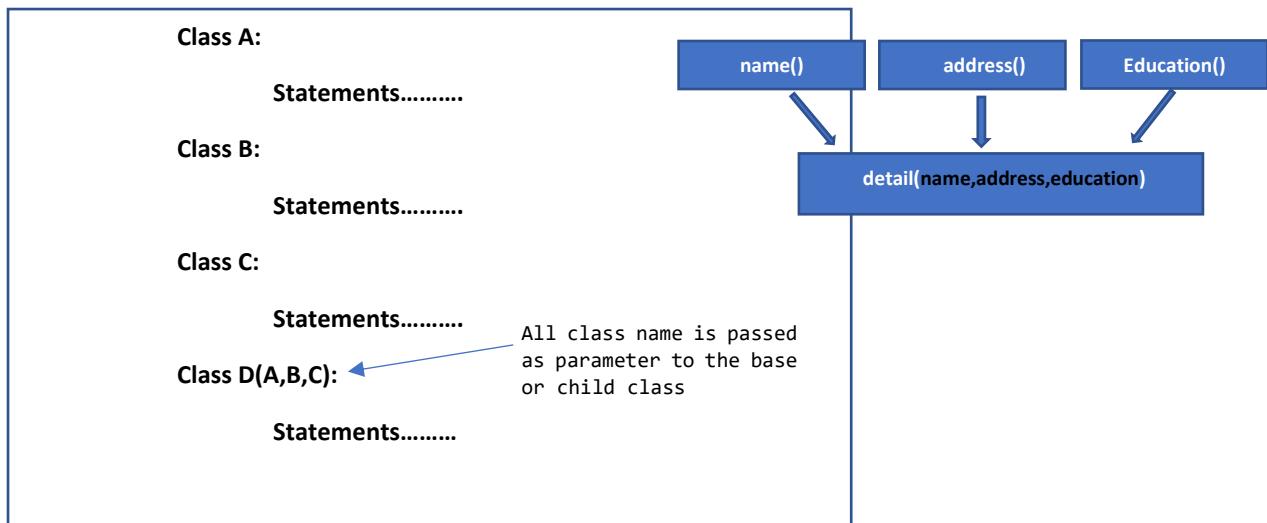


PYTHON PROGRAMMING

3). Multiple Inheritance:

- In multiple inheritance one class can inherit property of other class.
- There is no need that the classes are interconnected to each other.

Syntax:



Example to show Multiple inheritance:

```

File Edit View Navigate Code Befactor Run Tools VCS Window Help python sample program [D:\python program\python learning\python sample program] - inheritance_multiple.py - PyCharm
python sample program inheritance_multiple.py
Project inheritance_multilevel.py inheritance_multiple.py
1 class name:
2     def names(self):
3         self.firstname = input("enter first your name : ")
4         self.lastname = input("enter your last name : ")
5
6 class address:
7     def addressdetail(self):
8         self.address = input("enter your address : ")
9
10 class education:
11     def educationdetail(self):
12         self.stream = input("enter your stream : ")
13
14 class detail(name, address, education): # accessing a function of other class by passing the class name in the argument
15     def __init__(self):
16         self.names()
17         self.addressdetail()
18         self.educationdetail()
19
20     def printdata(self):
21         print(self.__dict__)
22
23 ram = detail()
24 ram.printdata()

```

Output:

```

File Edit View Navigate Code Befactor Run Tools VCS Window Help python sample program [D:\python program\python learning\python sample program] - inheritance_multiple.py - PyCharm
python sample program inheritance_multiple.py
Project inheritance_multilevel.py inheritance_multiple.py
Run: inheritance_multiple
C:\Users\suman\AppData\Local\Programs\Python\Python38-32\python.exe "D:/python program/python learning/python sample program/inheritance_multiple.py"
enter first your name : samundar
enter your last name : singh
enter your address : Jharkhand
enter your stream : Mca
{'firstname': 'samundar', 'lastname': 'singh', 'address': 'Jharkhand', 'stream': 'Mca'}
Process finished with exit code 0

```

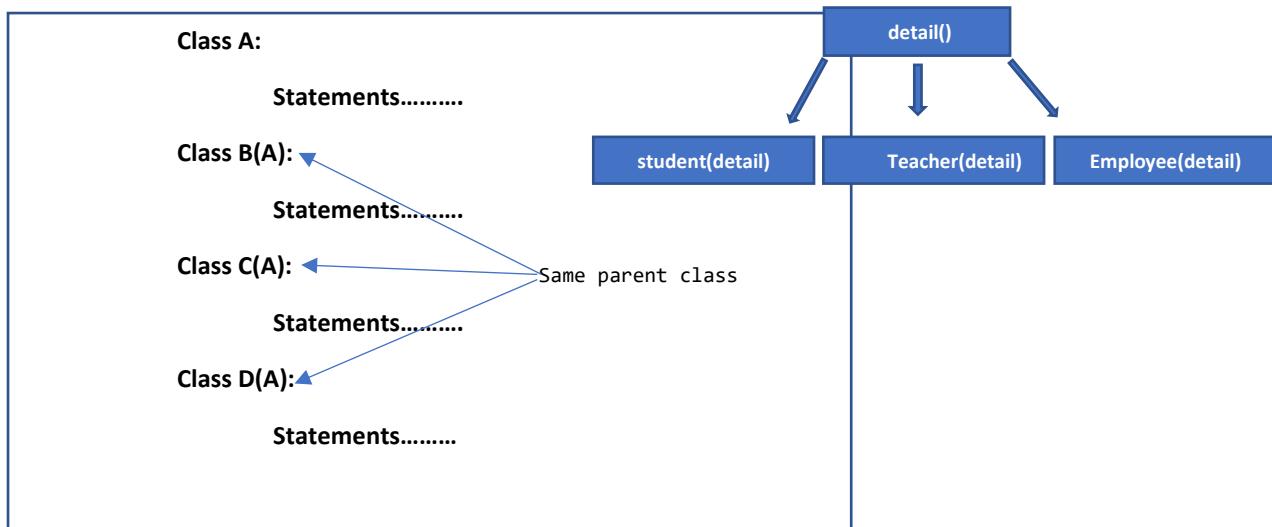


PYTHON PROGRAMMING

2). Hierarchical Inheritance:

- In hierarchical inheritance there is only one parent class and multiple child class all the child class inherit property of the parent class.

Syntax:



Example to show Hierarchical inheritance:

```

class detail:
    def __init__(self):
        self.name=input("enter your name : ")
        self.age=input("enter your age : ")
        self.address=input("enter your address : ")

    def print(self):
        print("student data : ",self.__dict__)

class student(detail):
    def printstudent(self):
        print("student data : ",self.__dict__)

class teacher(detail):
    def printteacher(self):
        print("teacher data : ",self.__dict__)

class employee(detail):
    def printemployee(self):
        print("employee data : ",self.__dict__)

print(" select option : \n 1.student   \n 2.teacher   \n 3.employee ")
choice=int(input("enter your choice to insert the data"))

if (choice==1):
    obj=student()
    obj.printstudent()
elif(choice==2):
    obj = teacher()
    obj.printteacher()
elif(choice==3):
    obj = employee()
    obj.printemployee()
else:
    print("invalid choice")
  
```

Output:

```

Run: inheritance_hierarchy
C:\Users\suman\AppData\Local\Programs\Python\Python38-32\python.exe "D:/python program/python learning/python sample program/inheritance_hierarchy.py"
select option :
1.student
2.teacher
3.employee
enter your choice to insert the data 2
enter your name : sanjay pandey
enter your age : 45
enter your address : Patna
teacher data : {'name': 'sanjay pandey', 'age': '45', 'address': 'Patna'}

Process finished with exit code 0
  
```



PYTHON PROGRAMMING

7. Method overriding and super() keyword

Before starting this first we have to understand the flow of execution of the class.

When we create a class and perform inheritance on it then whenever the object of child class is created than the constructor of the child class is first executed and if any variable or method is called through the object than first it find the variable in the constructor then outside of constructor and after then its parent class...and so on

```

1  class A:
2      var="this is variable of class A"
3      def __init__(self):
4          self.var = "constructor variable of A"
5          print("i am in constructor of class A")
6
7  class B(A):
8      var="this is variable of class B"
9      def __init__(self):
10         self.var = "constructor variable of b"
11         def display():
12             print("i am in constructor of class B")
13
14 class C(B):
15     var="variable of class c"
16     def __init__(self):
17         self.var="constructor variable of C"
18         def display():
19             print("I am in constructor of class C")
20
21 obj=C()
22 print(obj.var)
23

```

The constructor of class C override the constructor of both A and B class
Hence only C constructor runs

the variable inside the constructor will override the variable that is out side of the constructor

Run: overriding_
C:\Users\samun\AppData\Local\Programs\Python\Python38-32\python.exe "D:/python program/python learning/python sample program/overriding_.py"
constructor variable of C
Process finished with exit code 0

```

1  class A:
2      var="this is variable of class A"
3      def __init__(self):
4          self.var = "constructor variable of A"
5          print("i am in constructor of class A")
6
7  class B(A):
8      var="this is variable of class B"
9      def __init__(self):
10         self.var = "constructor variable of b"
11         def display():
12             print("i am in constructor of class B")
13
14 class C(B):
15     var="variable of class c"
16     def __init__(self):
17         self.var="constructor variable of C"
18         super().__init__()
19         def display():
20             print("I am in constructor of class C")
21
22 obj=C()
23 print(obj.var)
24

```

This super() keyword will override the constructor of class C and run Class B constructor

Hence the variable is also override

Run: overriding_
C:\Users\samun\AppData\Local\Programs\Python\Python38-32\python.exe "D:/python program/python learning/python sample program/overriding_.py"
constructor variable of b
Process finished with exit code 0

When the object of child class created then the constructor of the child class is executed if the requested variable is inside the constructor then it simply assign that value at the called object but later we invoked super() key thus the value of var is override by the variable of its parent class that's why instead of printing the variable of child class, it will print the value of parent class.



PYTHON PROGRAMMING

8. Function Caching

There are such situations in which a function will require time to process the data for example if we want some work related to database then it is not ideal to open the database again and again because it consumes time suppose if a function takes 3 sec to open a database and we call that function in our program 5 times then the execution time will take $3 \times 5 = 15$ sec to complete the task so there is a special method in python which will reduce this effort.

To achieve the function caching we have to import the `lru_cache` from the `functools` this `lru_cache` is a decorator which accepts `maxsize` as an argument in which we have to mention that how much this will cache the data of function with different arguments.

Syntax:

```
from functools import lru_cache
@lru_cache(maxsize=2)
def fun(n):
    statement.....
    return n
```

Note: in max size we have to mention that for how many various cases the function will store the data

The screenshot shows a PyCharm project window with the file `function_caching.py` open. The code defines a function `fun` that sleeps for `n` seconds and prints the result. It uses the `@lru_cache(maxsize=2)` decorator. The code is as follows:

```
1  import time
2  from functools import lru_cache
3  @lru_cache(maxsize=2)
4  def fun(n):
5      time.sleep(n)
6      return n
7
8  print("running start")
9  fun(3)
10 print("after 3 sec ")
11 fun(2)
12 print("after 2 more sec ")
13 fun(1)
14 print("after 1 sec ")
15 fun(3)
16 print("after 3 sec ")
17 fun(2)
18 print("after 2 sec ")
19 fun(3)
20 print("after 3 sec ")
21 fun(2)
22 print("after 2 sec ")
```

Two notes are present on the right side of the code editor:

- NOTE:** *In max size the value is 2 so it will only store the two values as cache that is for fun(3) and fun(2)*
- And when we give the fun(1) then it cannot be stored so that is not stored in cache so whenever fun(2) or fun(3) encounter than the program does not take time since it is stored in the cache*

The run tab at the bottom shows the output of the program:

```
running start
after 3 sec
after 2 more sec
after 1 sec
after 3 sec
```

9. Function Co-routine

There are situations in which a function will take time to complete its execution and we require that when we call that function again than instead of executing the whole function we execute some part of the function. That time we use co-routine.

For example if we have to access a database and we search a particular data in table and we create a function that accesses a database and search a data in table if we want to again search a data than the same table then same process executed means again we have to access the database and start searching table in database which is not ideal.

Here co-routine helps, the ideal approach is that if we access the database and table only once and search the data whenever needed.

Here the scenario is represented by means of time module. Here we create database and a sleep method to give it a real feel of fetching data from the database.



PYTHON PROGRAMMING

```

1 def search():
2     import time
3     #suppose we access a database having only name
4     database=["samundar","rahul","ramesh"]
5     time.sleep(3)
6
7     while True:
8         text=(yield)
9         if text in database:
10            print("data found")
11        else:
12            print("data not found")
13
14     find=search()
15     next(find)
16     find.send("samundar")
17     find.send("samundar")
18     find.send("rahul")
19     find.send("kishna")
20     find.send("tony")
21     find.send("ramesh")
22

```

Run: functio_co-routine x
C:\Users\samun\AppData\Local\Programs\Python\Python38-32\python.exe "D:/python program/python learning/python sample program/functio_co-routine.py"

data not found
data found
data found
data not found
data not found
data found

23:1 CRLF UTF-8 4 spaces Python 3.8 Event Log

10. Abstract base class in python

This is the class that enforces the child class to have the same function that is in the base class. If the function is not mentioned in the child class then it will show some error.

To achieve abstract base class we have to import ABCmeta from abc class.

```

1 from abc import ABC,abstractmethod
2 class shape(ABC):
3     @abstractmethod
4     def inputsides(self):
5         self.a=int(input("Enter the side"))
6         self.b=int(input("Enter the side"))
7
8 class rectangle(shape):
9     def inputsides(self):
10        self.a = int(input("enter the side : "))
11        self.b = int(input("Enter the side"))
12    def display(self):
13        print("the area of rectangle : ",self.a*self.b)
14
15 class square(shape): #here we can't mention the inputsides function of abstract base class
16     def __init__(self):
17         def display(self):
18             print("the area of square : ")
19
20     re=rectangle()
21     re.inputsides()
22     re.display()
23     sq=square()
24

```

Run: abstract_base_class x
abstract_base_class

27:1 CRLF UTF-8 4 spaces Python 3.8 Event Log

```

1 enter the side : 12
2 Enter the side2
3 the area of rectangle : 24
4 Traceback (most recent call last):
5   File "D:/python program/python Learning/python sample program/abstract_base_class.py", line 24, in <module>
6     sq.square()
7 TypeError: Can't instantiate abstract class square with abstract methods inputsides
8
9 Process finished with exit code 1

```



PYTHON PROGRAMMING

11. OS module in python

It is a built in module of python and the various methods are:

- 1.) **Rename():** To change the name of the file we use rename() method:
Syntax: os.rename(current_file_name, new_file_name)
- 2.) **Remove():** To remove the file we use remove() method:
Syntax: os.remove(file_name)
- 3.) **mkdir():** To create folder we use mkdir() method:
Syntax: os.mkdir("newdir")
- 4.) **chdir():** To change the current directory we use chdir() method:
Syntax: os.chdir("newdir")
- 5.) **getcwd():** To get current working directory we use getcwd() method:
Syntax: os.getcwd()

This module is helpful when we create any app and that app can create a folder and store data in it or to search a particular file in a particular location.

12. request module in python

When we create any app that will communicate with any website than in order to establish communication between the app and the website, we use request method

Note: first install the request method using the command : **pip install requests**

Here by using the get method we are going to hit an api which is already created we get that API from here:

This site have In-built API :- Financial Modeling Prep API Documentation

The screenshot shows the PyCharm IDE interface with a Python script named 'request_module.py'. The code in the editor is as follows:

```
1 import requests
2 r=requests.get("https://financialmodelingprep.com/api/v3/search?query=AA&limit=10&exchange=NASDAQ&apikey=demo")
3 print(r.text)
4
```

A blue arrow points from the text "This is the link where we have to fetch the data, the request returns response which we can understand by using text keyword after the r." to the URL in the second line of code.

This is the link where we have to fetch the data, the request returns response which we can understand by using text keyword after the r.

The PyCharm interface includes a Project tree on the left, a Run/Debug tool window at the bottom, and various status bars at the bottom right.



PYTHON PROGRAMMING

For post method we have to use a dictionary to send the data, here again we use a API to perform the post method in post method of request two argument are given that is the url where data have to be sent and the data

```
import requests
url="https://reqres.in/api/users"
data={
    "name": "samundar",
    "job": "programmer"
}
r=requests.post(url=url, data=data)
print(r.status_code)
print(r.text)
```

In this api whatever they asked we have fill the same thing in same format

Give it a try

Request /api/users

{
 "name": "morpheus",
 "job": "leader"
}

Response
201

{
 "name": "morpheus",
 "job": "leader",
 "id": "612",
 "createdAt": "2020-07-02T11:39:46.565Z"
}



PYTHON PROGRAMMING

13. JSON module in python

There is an in-built module n python that support the json encoding and decoding function.

Some of the important method of JSON module are as:

1. Python to JSON (encoding) using dumps:

The following python object converted to json object as given below:

Python	JSON
dict	Object
list	Array
unicode	String
number - int, long	number - int
float	number - real
True	True
False	False
None	Null

Python object is converted to JSON object with the help of **dumps()** method . **dumps()** method converts dictionary object of python into JSON string data format.The converted data cannot be written to any file. If we want to write the converted data into a .json file then we use **dump()** function.

Program to convert python object to json object (using dumps):

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help python sample program [D:\python program\python learning\python sample program] - ...\\json_module.py - PyCharm
python sample program json_module.py
json_module.py
1 import json
2 #python data
3 python_data={
4     "name": "samundar",
5     "age": 24,                                     #it is a dictionary
6     "marks": [70, 80, 100],                          #it is a list
7     "books": ("harrypotter", "jungle book", "godan"), #it is a tuple
8     "married": False
9 }
10 print("python data : \n ", python_data)
11 json_data=json.dumps(python_data)
12 print("json data : \n ", json_data)

Run: json_module ...
C:\Users\samun\AppData\Local\Programs\Python\Python38-32\python.exe "D:/python program/python learning/python sample program/json_module.py"
python data :
{'name': 'samundar', 'age': 24, 'marks': [70, 80, 100], 'books': ('harrypotter', 'jungle book', 'godan'), 'married': False}
json data :
{"name": "samundar", "age": 24, "marks": [70, 80, 100], "books": ["harrypotter", "jungle book", "godan"], "married": false}

Process finished with exit code 0
See the difference (tuple vs array)
```



PYTHON PROGRAMMING

If we want to write the above data into a json file then we use `dump()` method show in this program below:

Syntax:

```
import json
With open("your_json_filename.json", "Access mode") as ANY_NAME
Python_dictionary_name={ .....}
.....}
Json.dump(Python_dictionary_name, ANY_NAME, indent=4, separators( , , ':') )
```

See the file is empty

```
1 import json
2 #python data
3 with open('sam.json', 'w') as file_write:
4     python_data={
5         "name": "samundar",
6         "age": 24,
7         "marks": [70, 80, 100],
8         "books": ("harrypotter", "jungle book", "godan"),
9         "married": False
10    }
11    json.dump(python_data, file_write, indent=4, separators=(',', ' : '))
12    print("data is sucessfully written in the file")
13
14 #this will write the data in your json file
15 #NOTE: use append "a" in the file access mode in order to append data to afile
16
17
```

After code run the json data filled into the file

```
1 import json
2 #python data
3 with open('sam.json', 'w') as file_write:
4     python_data={
5         "name": "samundar",
6         "age": 24,
7         "marks": [70, 80, 100],
8         "books": ("harrypotter", "jungle book", "godan"),
9         "married": False
10    }
11    json.dump(python_data, file_write, indent=4, separators=(',', ' : '))
12    print("data is sucessfully written in the file")
13
14 #this will write the data in your json file
15 #NOTE: use append "a" in the file access mode in order to append data to afile
16
17
```

NOTE: 1. The `indent = 4` and `separators = (" , ", " : ")` parameter is just to show the data in pretty format
2. In order to append the data we use “`a`” append method in access mode



PYTHON PROGRAMMING

2.) JSON to Python (decoding or parsing):

The following JSON object converted to python object as given below

JSON	Python
Object	dict
Array	list
String	unicode
number - int	number - int, long
number - real	float
True	True
False	False
Null	None

JSON string decoding is done with the help of inbuilt method **loads()** & **load()** of JSON library in Python.
load() is used parse json data to python data it cannot write any data to the file

program to convert python object to json object (using dumps):

The screenshot shows the PyCharm IDE interface. The code editor contains a Python script named `json_module2.py` with the following content:

```
1 import json
2
3 json_data = '{ "name": "ramesh", '
4     '"age": 23, '
5     '"marks": [100, 230, 334], '
6     '"married": true, '
7     '"subject": ["maths", "english"] }'
8 #data parsing take place here
9 pyth_obj=json.loads(json_data)
10 print(pyth_obj)
```

The run tab shows the output of the script:

```
C:\Users\samun\AppData\Local\Programs\Python\Python38-32\python.exe "D:/python program/python learning/python sample program/json_module2.py"
{'name': 'ramesh', 'age': 23, 'marks': [100, 230, 334], 'married': True, 'subject': ['maths', 'english']}
Process finished with exit code 0
```

Note: 1. Whatever data is given as json data must be given as string.

2. Since the data is in json format it means it is either in key value pair or array so it can be converted to dictionary or list.



PYTHON PROGRAMMING

If we want to read data from a json file then we use load() method show in this program below:

The screenshot shows the PyCharm IDE interface. On the left, the Project tool window lists files: json_module.py, json_dump file write.py, json_module2.py, and json_load file read.py. The json_load file read.py file is open in the main editor area, containing the following code:

```
1 import json
2 with open('sam.json') as file_obj:
3     python_data = json.load(file_obj)
4 print(python_data)
```

To the right of the editor is the Python Console window, which displays the contents of the 'sam.json' file:

```
1 {
2     "name": "samundar",
3     "age": 24,
4     "marks": [
5         70,
6         80,
7         100
8     ],
9     "books": [
10         "harrypotter",
11         "jungle book",
12         "godan"
13     ],
14     "married": false
15 }
```

The screenshot shows the PyCharm IDE interface after running the code. A blue arrow points from the explanatory text below to the Run tool window at the bottom.

After code run the json data is read as python object

The Run tool window shows the command run and the output:

```
C:\Users\sumun\AppData\Local\Programs\Python\Python38-32\python.exe "D:/python program/python learning/python sample program/json_load file read.py"
```

The output shows the JSON data as a Python object:

```
{"name": "samundar", "age": 24, "marks": [70, 80, 100], "books": ["harrypotter", "jungle book", "godan"], "married": false}
```

Process finished with exit code 0



TO BE CONTINUE.....



PYTHON PROGRAMMING