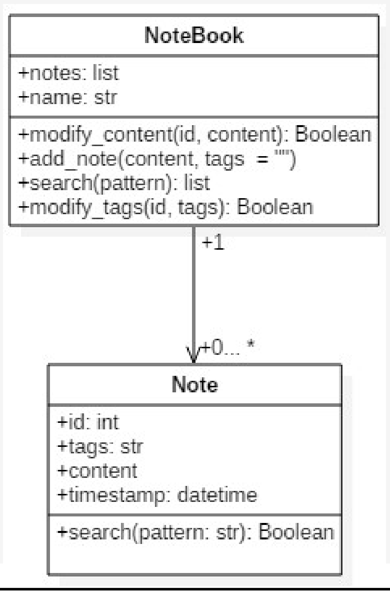
Back to Notes and Notebooks

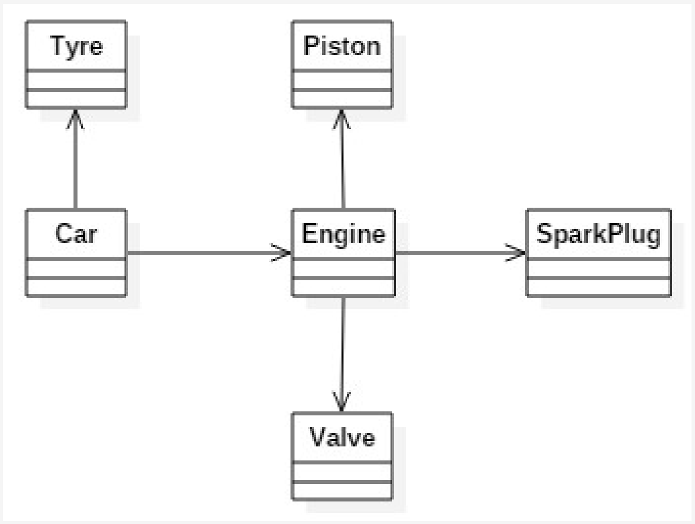
Note vs **Notebook**

A NoteBook contains multiple Note objects.


NoteBook
+notes: list
+name: str
+modify_content(id, content): Boolean
+add_note(content, tags = "")
+search(pattern): list
+modify_tags(id, tags): Boolean
+1
+0... *
Note
+id: int
+tags: str
+content
+timestamp: datetime
+search(pattern: str): Boolean

trainer.cpp@gmail.com

Car Example



NoteBook Methods

- Create a constructor __init__()

  name as argument and initializes **name**

  creates an empty list and assigns to **notes** attribute**.**

- *add_note(content, tags="") -> None*:

  create a new **note** object and pass **id, content, tag** to constructor of **Note**

  Add the new **Note** to the **list** of notes

  The **note id** of a new note is generated depending on length of the existing list

trainer.cpp@gmail.com

NoteBook Methods Continued

- *search(pattern) -> list* :

  create an empty list

  iterate on the n**otes** list and call the **search** method of the **Note** class

  if **search** return T**rue**, add to the list

  return the final **list**

- *modify_content(id, content),->Boolean:*

  Find a matching by **searching** in the **list** on basis of **id**

  On match, change the content and return **True**

  Function should return **False** if no matching note is found

trainer.cpp@gmail.com

NoteBook Methods Continued

- *modify_tag(id, tag)* -> *Boolean:*

  similar to *modify_content* method, but should work on tag instead

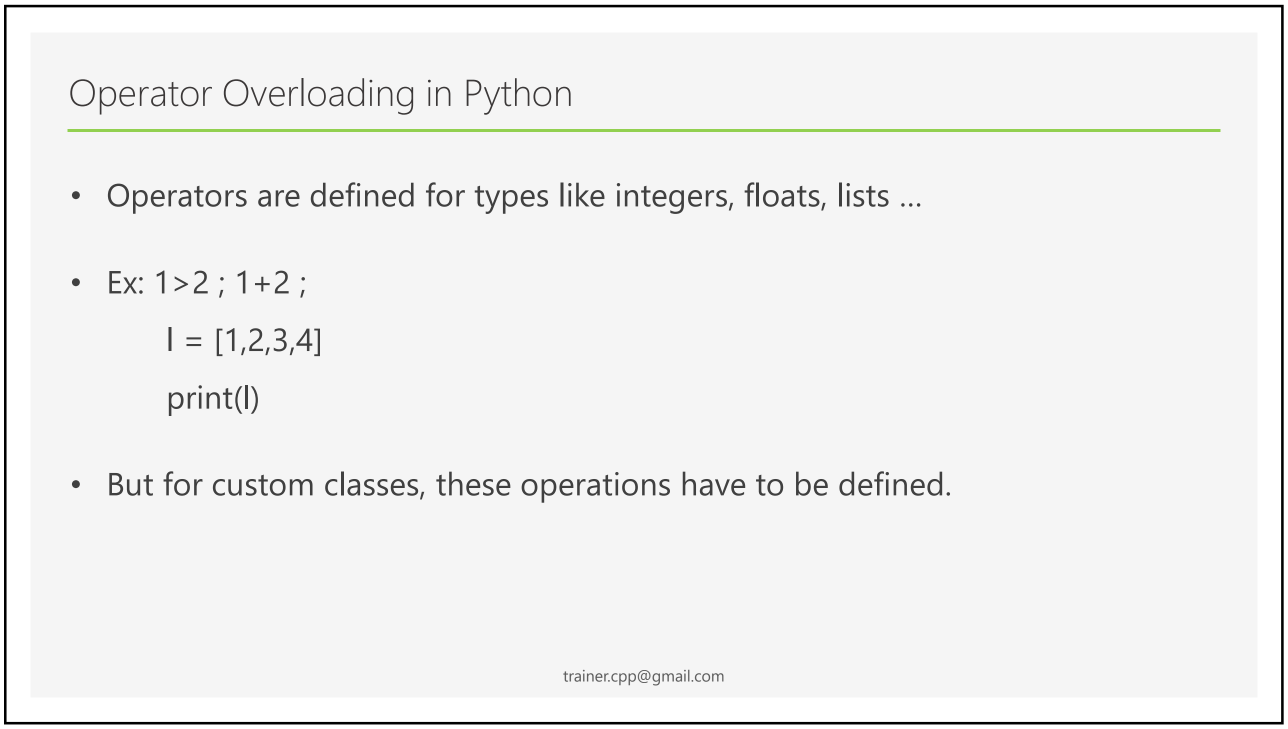
Some Management for NoteBook

- Try to create a class **NoteBookManager** that displays a menu to edit, add, display and search notes.
- Update the **NoteBookManager** class to handle multiple notes
- Additionally try to persist notes in separate files, so that they can be accessed later [use file handling]

  Save all NoteBooks in a separate folder **file**

  Create one file for each NoteBook.

  One file will contain multiple notes



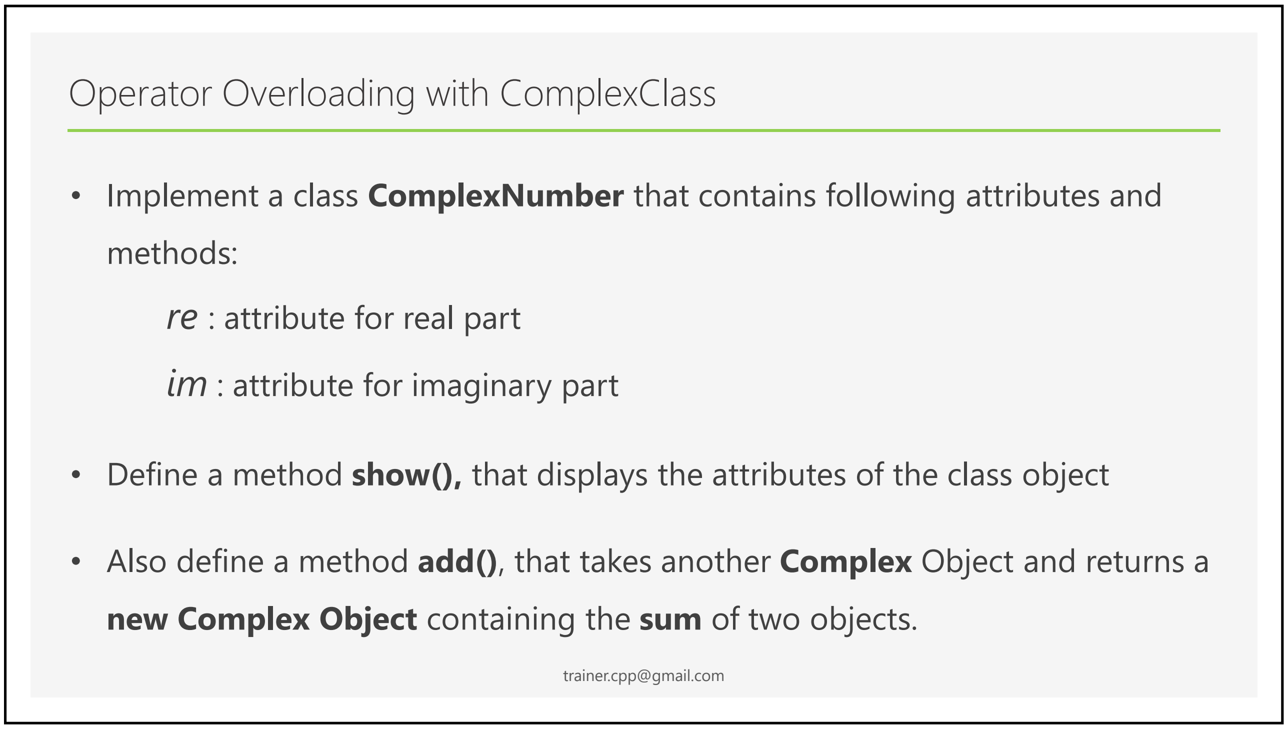
Operator Overloading in Python

- Operators are defined for types like integers, floats, lists …
- Ex: 1>2 ; 1+2 ;

  l = [1,2,3,4]

  print(l)
- But for custom classes, these operations have to be defined.

trainer.cpp@gmail.com



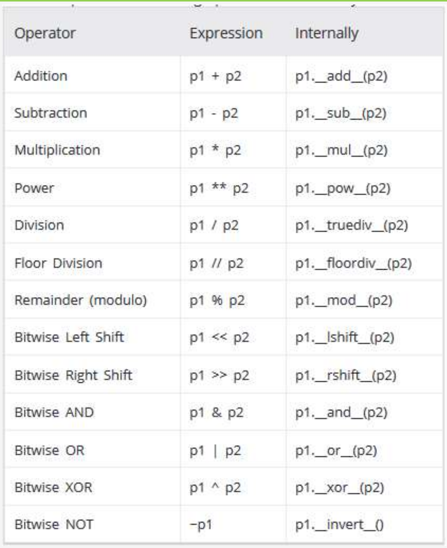
Operator Overloading with ComplexClass

- Implement a class **ComplexNumber** that contains following attributes and methods:

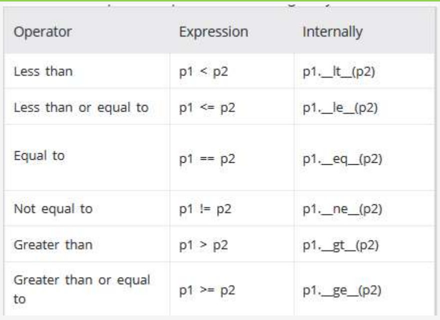
  re : attribute for real part

  im : attribute for imaginary part
- Define a method **show(),** that displays the attributes of the class object
- Also define a method **add()**, that takes another **Complex** Object and returns a **new Complex Object** containing the **sum** of two objects.

trainer.cpp@gmail.com



Operator	Expression	Internally
Addition	p1 + p2	p1.__add__(p2)
Subtraction	p1 - p2	p1.__sub__(p2)
Multiplication	p1 * p2	p1.__mul__(p2)
Power	p1 ** p2	p1.__pow__(p2)
Division	p1 / p2	p1.__truediv__(p2)
Floor Division	p1 // p2	p1.__floordiv__(p2)
Remainder (modulo)	p1 % p2	p1.__mod__(p2)
Bitwise Left Shift	p1 << p2	p1.__lshift__(p2)
Bitwise Right Shift	p1 >> p2	p1.__rshift__(p2)
Bitwise AND	p1 & p2	p1.__and__(p2)
Bitwise OR	p1 | p2	p1.__or__(p2)
Bitwise XOR	p1 ^ p2	p1.__xor__(p2)
Bitwise NOT	~p1	p1.__invert__()



Operator	Expression	Internally
Less than	p1 < p2	p1.__lt__(p2)
Less than or equal to	p1 <= p2	p1.__le__(p2)
Equal to	p1 == p2	p1.__eq__(p2)
Not equal to	p1 != p2	p1.__ne__(p2)
Greater than	p1 > p2	p1.__gt__(p2)
Greater than or equal to	p1 >= p2	p1.__ge__(p2)

trainer.cpp@gmail.com

__str__ and __repr__

- Python provides 2 magic methods that can be overridden in a class
- __str__ is used to generate a string representation, which is meant to be easily readable (used by print method).
- __repr__ is used to generate how things are represented internally by the system
- Ex:
  print(repr(" a,'b' ")), print(str(" a,'b' "))

** add these methods to the NoteBook and Note classes

trainer.cpp@gmail.com