Topic 1:

Easy to reuse

Topic 2:

class Student:

def \_\_init\_\_(self, name, age, GPA):

self.name = name

self.age = age

self.GPA = GPA

def get\_GPA(self):

return self.GPA

def set\_GPA(self, GPA):

self.GPA = GPA

def main():

bob = Student("Bob", 15, 3.0)

print(bob.get\_GPA())

bob.set\_GPA(4.0)

print(bob.get\_GPA())

if \_\_name\_\_ == '\_\_main\_\_':

main()

The self keyword represents the object itself when the method is called.

Topic 3：

A）: the price is, 11

B）: x[index] =

del()

x+

x+=

x=

len()

str()

repr()

contains()

iter()

Topic 4:

class Tree:

def \_\_init\_\_(self, name, age):

self.\_name = name

self.\_age = age

def get\_name(self):

return self.\_name

class Palm(Tree):

def \_\_init\_\_(self, name, age, color):

super().\_\_init\_\_(name, age)

self.\_color = color

def get\_color(self):

return self.\_color

def main():

palm1 = Palm("Lucky", 30, "Green")

print(palm1.get\_name())

print(palm1.get\_color())

if \_\_name\_\_ == '\_\_main\_\_':

main()

print(1): Lucky

print(2): green

print(3): Funny

print(4):20

Topic 5:

A): the code that is only being executed when the program is used by itself instead of being imported.

B): \_\_slots\_\_ allows you to explicitly state which instance attributes you expect your object instances to have, with the expected results