 # Task 1

What is the main purpose behind using the virtual environment?

If you are working on different projects, you may face the problem with projects that depend on different versions of libraries.

To prevent such clutter, developers often create a virtual environment for a project to keep the development environment and system environment clean.

How to create a virtual environment?

We can use Virtualenv .

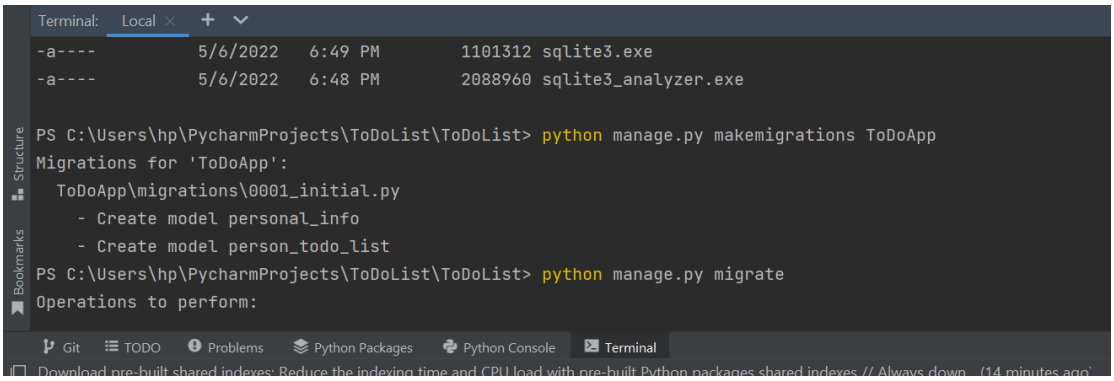
Virtualenv can create several different virtual environments that do not overlap each other in the system.

Creation of virtual environments is done by executing the command venv:

```
python -m venv .venv
```

 # Task 2:

In terminal:




```
from ToDoApp.models import personal_info

person = personal_info(f_name='wafa', l_name='ali')

person.save()

person = personal_info.objects.get(f_name='wafa')

personal_info.objects.all()
```

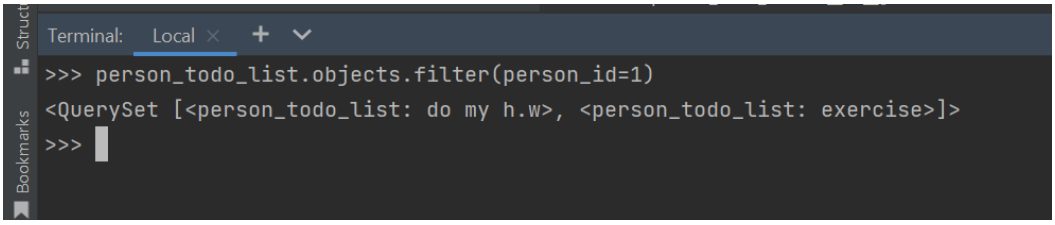
```
>>> person = personal_info.objects.get(f_name=wafa')
  File "<console>", line 1
    person = personal_info.objects.get(f_name=wafa')
                                                   ^
SyntaxError: unterminated string literal (detected at line 1)
>>> person = personal_info.objects.get(f_name='wafa')
>>> personal_info.objects.all()
<QuerySet [<personal_info: wafa ali>]>
>>> person.l_name
'ali'
>>> person.f_name
'wafa'
>>>
```

```
from ToDoApp.models import person_todo_list

person_todo_list.objects.create(item ="do my h.w", person=person)
```

```
>>> from ToDoApp.models import person_todo_list
>>> person_todo_list.objects.create(item ="do my h.w", person=person)
<person_todo_list: do my h.w>
>>> person_todo_list.objects.create(item ="exercise", person=person)
<person_todo_list: exercise>
>>>
```

person_todo_list.objects.filter(f_name='wafa')



personal_info table

Table: ToDoApp_personal_info

	id	f_name	l_name
	Filter	Filter	Filter
1	1	wafa	ali

person_todo_list table

Table: ToDoApp_person_todo_list

	id	item	person_id
	Filter	Filter	Filter
1	1	do my h.w	1
2	2	exercise	1

models.py

views.py | models.py | urls.py

```
# Create your models here.

class personal_info(models.Model):
    f_name = models.CharField(max_length=50, help_text="Your first name .")
    l_name = models.CharField(max_length=50, help_text="Your last name")

    def __str__(self):
        return self.f_name + ' ' + self.l_name


class person_todo_list(models.Model):
    item = models.CharField(max_length=100, help_text='add your to do list item')
    person = models.ForeignKey(personal_info, on_delete=models.CASCADE)

    def __str__(self):
        return self.item
```