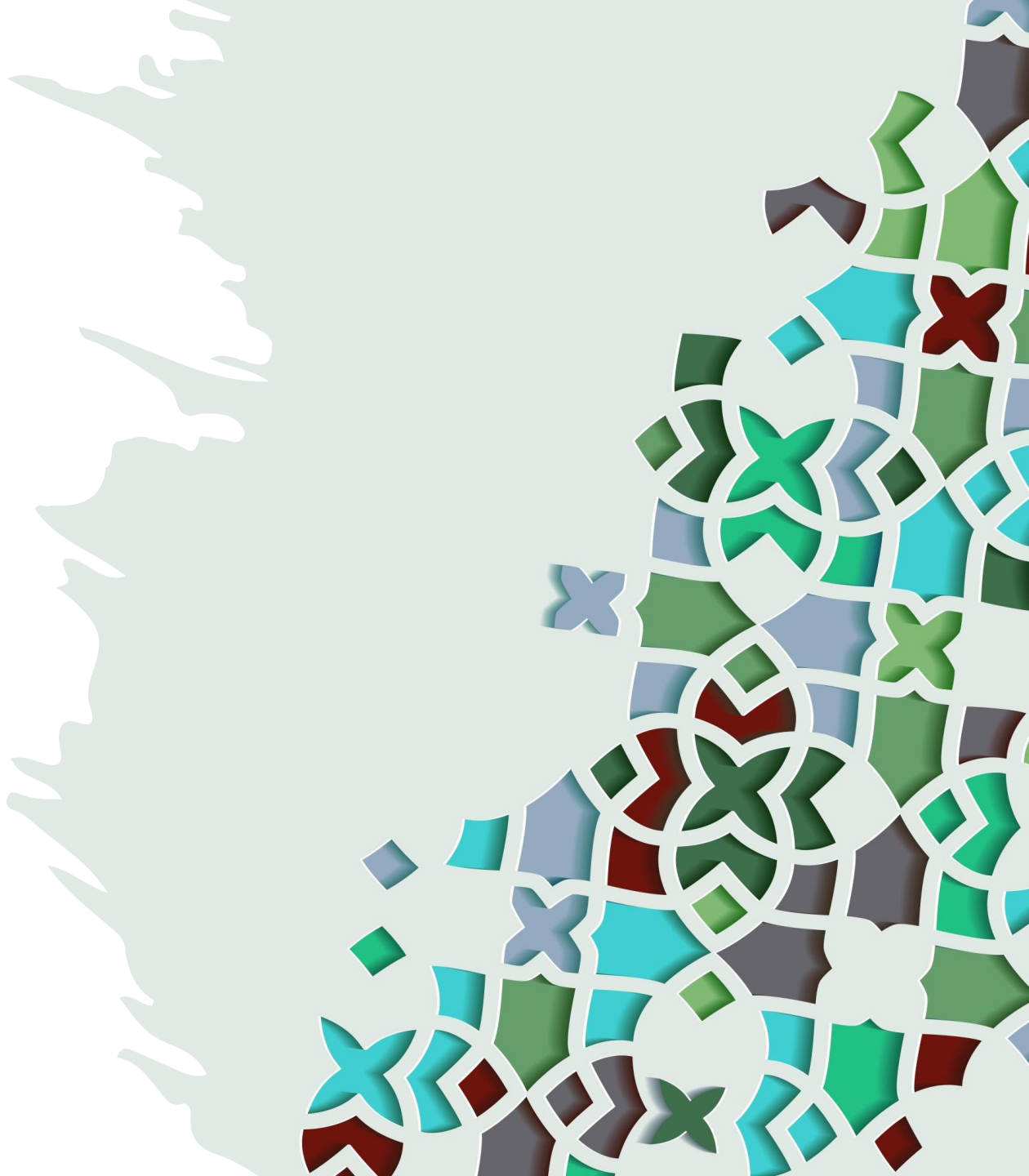
create, delete,
update, view

objects

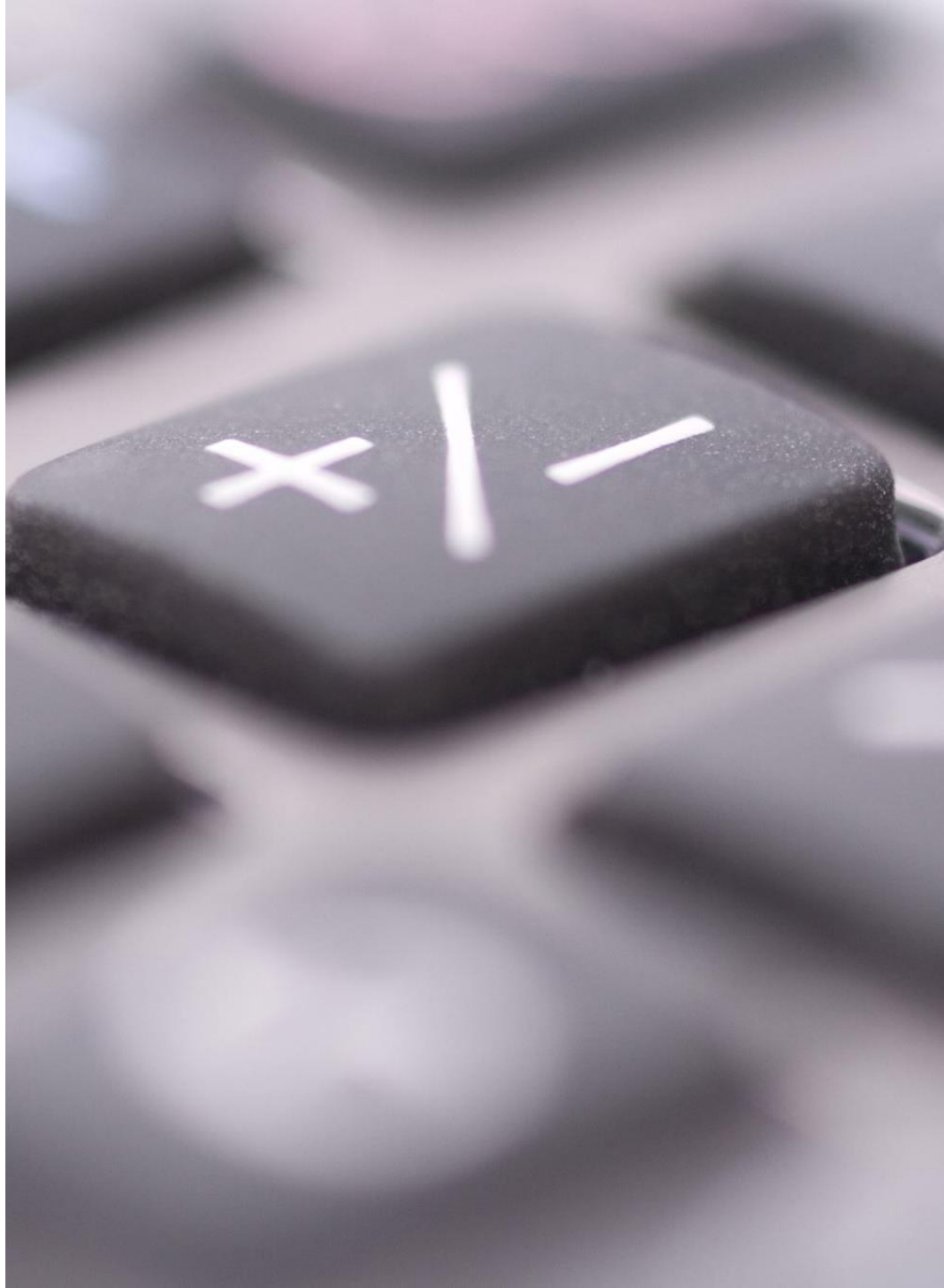
python manage.py shell



Model Company

```
class Company(models.Model):


  COMPANY_TYPES = (
      ("tech", "Tech Business"),
      ("food", "Food Industry"),
  )
  name = models.CharField(max_length=100)
  description = models.TextField('Beschreibung der Firma')
  number_of_employees = models.IntegerField()
  company_type = models.CharField(
      max_length=10,
      choices=COMPANY_TYPES
  )
  sub_title = models.CharField(max_length=30, null=True, blank=True)
```



create a new object

To create a new object in the database (in the Django shell or in a view), we simply create a new object of the desired class.

```
from appname.models import Company
```

```
company_1 = Company(name='Superduper AG‘,
description='IT Company from Hamburg',
number_of_employees=5, company_type='tech')
```

with the save() method of the model we can save the object persistently in the database

```
company_1.save()
```

inspect object

The newly created object is an object of the Company class. We can access the attributes and methods of this object.

company_1.name

Superduper Ag

Get object from database

We can load an object via the ID (=primary key). For this we use the get method of the object manager of the model. The get-Method cannot return more than one single object!

```
id=1
c = Company.objects.get(pk=id)
c
<Company: Suberduper AG>
```

instead of pk, we could use id as well

```
c = Company.objects.get(id=id)
```

dict

we can view the attributes with __dict__

company_1.__dict__

{

'_state': <django.db.models.base.ModelState object at 0x7fd1207f9160>,

'id': 1,

'name': 'Superduper AG',

'description': 'eine IT Firma aus Hamburg',

'number_of_employees': 5,

'company_type': 'tech',

'sub_title': None

}

id is an autoincrement and was set automatically. _state is a class for holding instance data

Model Manager

A manager is the interface through which database operations are provided for a model. For each model there is at least one manager (default). The name of the default manager is objects.

```
>>Company.objects
<django.db.models.manager.Manager object at 0x7f8f09cb71c0>
```

Manager get Methods

We have already seen the get method. We can pass more keyword arguments to it.

```
c = Company.objects.get(name='Superduper AG', number_of_employees=5)
c
<Company: Suberduper AG>
```

If no object is found based on the keyword arguments, an exception is thrown.

Delete object

To delete an object from the DB, we must first load the object instance via get(). Then we execute the delete() method on the instance.

```
o = Company.objects.get(pk=1)
deleted_object = o.delete()
```

The object has now been deleted from the database.In deleted_object there is a tuple with the ID of the deleted object and the amount of objects, which were delete.

update object

To update an object (in the database), we must first load the object instance via get(). Then we assign new values to it and again use the save() method of the instance itself.

```
o = Company.objects.get(pk=1)
o.number_of_employees = 3
o.save()
```

The object has now been saved in the database.

Delete and update more than one object

We can of course delete and update more than one object. We will see how this works later.

delete all objects, which is_active Attribute is false

```
objects = Company.objects.filter(is_active=false).delete()
```

Update all objects sub_title, where current sub_title="x"

```
objects = Company.objects.filter(sub_title="x").update(sub_title="")
```