1#VM image setup (konto: appuser:appuser)
```
cd apps
mkdir Koolitus
```
cd Koolitus
pyvenv --system-site-packages venv
source venv/bin/activate
git clone 'https://github.com/margus-parnsalu/Course.git'
cd Course
python setup.py develop
initialize_minu_db development.ini
pserve development.ini

2#Koolituse setup ja raamistikud:
Postgre - psycopg2 - SqlAlchemy - Pyramid - WTForms - Jinja2
————————————————————————————————————
Pyramid - http://docs.pylonsproject.org/en/latest/docs/pyramid.html
SqlAlchemy - http://docs.sqlalchemy.org/en/rel_1_0/
Jinja2 - http://jinja.pocoo.org
WTForms - https://wtforms.readthedocs.org/en/latest/
Bootstrap - http://getbootstrap.com

3#Pyramid kui M - Model, V - View, T - Template raamistik

URL Dispatch - /minu.__init__.py
->
View logic - /minu.views.py
->
Model logic - /minu.models.py
Form logic - /minu.forms.py
->
View logic  - /minu.views.py
->
Template - /minu/templates/*.jinja2

Abimoodulid Demo projektis:
Turvamoodul naidisloogikaga - security.py
Aruannetes custom sortimise loogika - sorts.py
Testid - tests.py

4#Pyramid rakenduse käivitamine ja helper skriptid
Folderid ja sisaldus: Setup.py; development.ini ja production.ini

```
- cd <directory containing this file>

- $VENV/bin/python setup.py develop #Projekti eeldusseoste
paigaldamine
```

```
- $VENV/bin/initialize_minu_db development.ini #Andmebaasi
tabelite loomine mudeli alusel

- $VENV/bin/pserve development.ini #Rakenduse kaivitamine
lokaalses masinas
```

5#PCREATE scaffold projekti genereerimine:
Õppimiseks hea vaadata, kuidas näidisprojektides struktuuri ja komponente(N: JInaja2, SqlAlchemy) kasutatakse

```
pcreate -s alchemy minu_projekt
```

6#Rakenduse tutvustamine Department naitel
URL Dispatch routes
Views
Templates - base ja vaadete seos
Forms

pserve development.ini —reload

7#Pyramid Debug toolbar
Eeldus HTML </body>

8#Jooksva rakendusega naide renderer jinja pealt json peale ja tagasi.

9#Tutvu Employee mudeliga

Ava - models.py
Vaata class Employee mudelit ja tuvasta valjade tyybid ning seos teiste class'ide / tabelitega
Mis eesmarki kannab vali 'department'?

Veendu, et andmebaasis on olemas hr_employees tabel ja kasuta pshell'i et testida Employee
klassi toimivust.
Projekti kaustas: $VENV/pshell development.ini

```
>>> from minu.models import DBSession, Employee
>>> q=DBSession.query(Employee).all()
```

Mis on q vaartus ja q andmetüüp?

10#Kirjelda URI suunamised vaadetele

Tegevus:
Vota copy/paste Departments routidest. ‘employee_add’ juba eksisteerib
Rename department -> employee.
N: `config.add_route('employee_view', '/employees')`

Tulem:

```
#Employees
config.add_route('employee_view', '/employees')
config.add_route('employee_view:page', '/employees/page/{page:\d+}')
config.add_route('employee_add', '/employees/add')
config.add_route('employee_edit', '/employees/{emp_id:\d+}/edit')
```

11#Loo Employee aruanne rakendusse

View:
Kasuta department_view definitsiooni copy/paste.
Rename department -> employee

route_name=‘employee_view’

Muuda paring selliseks, mis tagastab Employee koos Department objektiga. Kasuta outerjoin meetodit ehk Employee tagastatakse ka juhul, kui Department pole tabelites seotud.

Lisa paringu tingimus, et Employee.end_date oleks tyhi.

Tulem:

```
employees = (DBSession.query(Employee, Department)
                .outerjoin(Department, Employee.department_id==Department.department_id)
                .filter(Employee.end_date==None)
                .order_by(text(sort_value))
            )
```

12#Template
Loo department_r.jinja2 alusel
Aruanne peaks naitama valja all olevad valjad ja voimaldama tootaja nime pealt liikumist muutmise vormile:
Tulem:

```
    <table class="table table-hover">
            <thead>
                <tr>
                    <th>{{ sorter('employee_view', 'Name', '+employee') }}</th>
                    <th>{{ sorter('employee_view', 'Department', '+department') }}</th>
                    <th>{{ sorter('employee_view', 'Salary', '+salary') }}</th>
```

```
                    <th>{{ sorter('employee_view', 'Hire
Date', '+hired') }}</th>
                    <th>{{ sorter('employee_view', 'End Date',
'+hireend') }}</th>
                </tr>
            <tbody>
                {% for employee, department in employees %}
                <tr>
                    <td><a
href="{{ request.route_url('employee_edit',
emp_id=employee.employee_id) }}">
                            {{ employee.first_name + ' ' +
employee.last_name }} </a></td>
                    <td> {{ department.department_name }} </
td>
                    <td> {{ employee.salary }} </td>
                    <td> {{ employee.hire_date }} </td>
                    <td> {{ employee.end_date or '' }} </td>
                </tr>
                {% endfor %}
            </tbody>
    </table>
    <div class="text-center">
    <ul class="pagination">
        <li>{{ employees.pager(format='$link_first
$link_previous~2~$link_next$link_last',
                curpage_attr={ 'class':'active' },
                dotdot_attr={ 'class':'disabled' })|safe }}</
li>
    </ul>
    </div>
```

Taienda menyy punkti aruande viitega base.jinja2 faiiis:

```
    <li><a
href="{{ request.route_url('employee_view') }}">Employees</
a></li>
```

Katseta auande toimivust rakenduses

13#Loo vormi loogika tootajate muutmiseks

```
from .models import DBSession, Department, Employee

#LOV ehk Query_factory Departments jaoks
def Departments():
    return DBSession.query(Department).all()

from wtforms import validators, StringField, IntegerField,
```

```
DateField, QuerySelectField
```

Vormi klassi loomine

```
class EmployeeForm(BaseForm):
    first_name = StringField(u'First Name',
[validators.Length(min=4, max=64),
validators.InputRequired(message=(u'Input First Name'))])
    last_name = StringField(u'Last Name',
[validators.Length(min=4, max=64),
validators.InputRequired(message=(u'Input Last Name'))])
    email = StringField(u'E-mail', [validators.Email(),
validators.InputRequired(message=(u'Input E-mail'))])
    phone_number = StringField(u'Phone Number',
[validators.Length(min=4, max=20),
validators.InputRequired(message=(u'Input Phone Number'))])
    salary = IntegerField(u'Salary',
[validators.InputRequired(message=(u'Input Salary'))])
    hire_date = DateField(u'Hire Date',
[validators.InputRequired(message=(u'Select Hire Date'))],
format='%d-%m-%Y')
    end_date = DateField(u'End Date', [validators.Optional()],
format='%d-%m-%Y')
    department = QuerySelectField('Department',
[validators.DataRequired()], query_factory=Departments,
allow_blank=True)
```

QuerySelectField osakonna valimiseks ja LOV tyypi query factory loomine:
LOV kontekst

```
Query_factory seos naide:
    department = QuerySelectField('Department',
[validators.DataRequired()], query_factory=Departments,
allow_blank=True)
```

14#Loo muutmise vaade ehk views.py: employee_edit

```
from .forms import (DepartmentForm, EmployeeForm)

@view_config(route_name='employee_edit',
renderer='employee_f.jinja2', request_method=['GET','POST'])
def employee_edit(request):
    try:
        employee =
DBSession.query(Employee).filter(Employee.employee_id==request
.matchdict['emp_id']).first()
    except DBAPIError:
        return Response(conn_err_msg, content_type='text/
plain', status_int=500)
    if employee is None:
```

```
        return HTTPNotFound('Employee not found!')
    form = EmployeeForm(request.POST, employee,
csrf_context=request.session)
    if request.method == 'POST' and form.validate():
        #Update Employee
        employee.first_name = form.first_name.data
        employee.last_name = form.last_name.data
        employee.email = form.email.data
        employee.phone_number = form.phone_number.data
        employee.salary = form.salary.data
        employee.hire_date = form.hire_date.data
        employee.department = form.department.data
        employee.end_date = form.end_date.data
        DBSession.add(employee)
        request.session.flash('Employee Updated!')
        return
HTTPFound(location=request.route_url('employee_view'))
    return {'form': form}
```

15#Tutvu rakenduses Employee menyyga ja sisesta andmeid

FINISH

#BOONUS LINGID
#Pyramid SqlAlchemy tutorial:
http://docs.pylonsproject.org/projects/pyramid/en/latest/tutorials/wiki2/index.html
#Python style guide:
https://google-styleguide.googlecode.com/svn/trunk/pyguide.html?showone=Naming#Imports
#HR Demo laiem näidis:
https://github.com/margus-parnsalu/HRDemo
#Python 10 myths by PayPal
https://www.paypal-engineering.com/2014/12/10/10-myths-of-enterprise-python/