Урок 7

Отладка

`python -m pdb manage.py runserver 8000`

import pdb; pdb.set_trace()

l(ist) [<first> [,<last>]] - печать исходного кода.
a(rgs) - выводит аргументы функции;

whatis <arg> - выведет тип объекта;

! - префикс для работы напрямую с интерпретатором
w(here) - выводит информаию о позиции в которой сейчас находитесь;

s(tep) - "step into", перейти во внутрь вызова объекта, если это возможно, иначе перейти к следующей строке

n(ext) - "step over" (перешагнуть), перейти к следующей строке кода;

unt(il) - перейти к следующей строке кода, но гарантировано чтобы номер строки был больше чем текущий.

r(eturn) - завершить ("выйти из") текущую функцию;

u(p) - подняться на один стек-фрейм вверх;

d(own) - опуститься на один стек-фрейм вниз;

j(ump) <lineno> - перепрыгнуть на указанную строку кода не выполняя код находящийся между

Процессы и потоки

```
import subprocess
```

call - просто

```
print subprocess.call('python D:/axe/python/easygui/easygui.pyw', shell=True)
```

popen - правильно

```
from subprocess import Popen, PIPE
proc = Popen(
    "python D:/axe/python/easygui/easygui.pyw",
    shell=True,
    stdout=PIPE, stderr=PIPE
)
proc.wait()    # дождаться выполнения
res = proc.communicate()  # получить tuple('stdout res', 'stderr res')
if proc.returncode:
    print res[1]
print 'result:', res[0]
```

Процессы и потоки

```
django@convert:~$ pdfinfo /home/1.pdf
Creator:        Adobe InDesign CS5.5 (7.5.1)
Producer:       Adobe PDF Library 9.9
CreationDate:   Mon Jan 12 10:35:36 2015
Tagged:         yes
Pages:          24
Encrypted:      no
Page size:      841.89 x 1190.55 pts
File size:      36445800 bytes
```

```
def get_pages_count(paper):
        call = ['pdfinfo', '/home/1.pdf']
        pages_count = 0
        for string in subprocess.check_output(call).split('\n'):
                if 'Pages:' in string:
                pages_count = int(string.split(':')[1])
                break
        return pages_count
```

Процессы и потоки

PIPE - это технология, которая позволяет выстраивать запуск процессов в цепочки и передавать информацию между ними. Если stdin=PIPE, то доступен файловый дескриптор proc.stdin далее можно делать: proc.stdin.write('какой-то текст') proc.close()

```
from threading import Thread
from time import sleep
class MyThread(Thread):
    def __init__(self):
        Thread.__init__(self)
    def run(self):
        for i in range(20):
            print self
            sleep(0.222)
c = MyThread()
b = MyThread()
c.start()
b.start()
print "finish"
```

Управление процессами в Linux

Команда ps

-a отобразить все процессы, связанных с терминалом (отображаются процессы всех пользователей)
-e отобразить все процессы
-t список терминалов отобразить процессы, связанные с терминалами
-u идентификаторы пользователей отобразить процессы, связанные с данными идентификаторыми
-g идентификаторы групп отобразить процессы, связанные с данными идентификаторыми групп
-x отобразить все процессы, не связанные с терминалом

ps -ax | grep httpd

kill 5431

Программа top

free mem - оперативка

df - диск

Заворачиваем bash команды питоном.

Пакет registration

pip install django-registration

INSTALL_APPS (

..

'registration',

..

)

python manage.py syncdb

```
    url(r'^accounts/', include('registration.urls')),
    url(r'^logout/$', 'django.contrib.auth.views.logout',{'next_page': '/'}, name='logout'),
    url(r'^login/$', 'django.contrib.auth.views.login', name='login'),
```

```
<a href="{% url 'registration_register' %}">Registration</a>
```

Шаблон формы

```
{% extends "base.html" %}
{% block 'content' %}
<h1>Регистрация</h1>
<form method="post" action="">{% csrf_token %}
<dl class="register">
{% for field in form %}
    <dt>{{ field.label_tag }}</dt>
    <dd class="clearfix">{{ field }}
    {% if field.help_text %}<div class="clearfix">{{ field.help_text }}</div>{% endif %}
    {% if field.errors %}<div class="myerrors clearfix">{{ field.errors }}</div>{% endif %}
    </dd>
{% endfor %}
</dl>
<input type="submit" class="btn btn-success" value="Зарегистрироваться" / class="clearfix">
</form>
{% endblock %}
```

Авторизация

```
{% if user.is_authenticated %}
  <a class="btn btn-danger pull-right" href="{% url 'logout' %}">{% trans 'Logout' %}</a>
{% else %}
          {% for error in form.non_field_errors %}
              {{ error }}
          {% endfor %}
       <form method="post" action="{% url 'django.contrib.auth.views.login' %}" class="form_class">
            {% csrf_token %}
                    <input name="username" type="text" class="form-control" placeholder="{% trans 'Login' %}">
                    <input type="password" name="password"  class="form-control" placeholder="{% trans "Password" %}">
                     <button type="submit" class="btn btn-success pull-right">{% trans "Login" %}</button>
        </form>
      {% endif %}
```

Шаблоны пакета registration

activate.html

activation_compleate.html

activation_email.txt

activation_email_subject.txt

login.html

logout.html

registration_compleate.html

registration_form.html