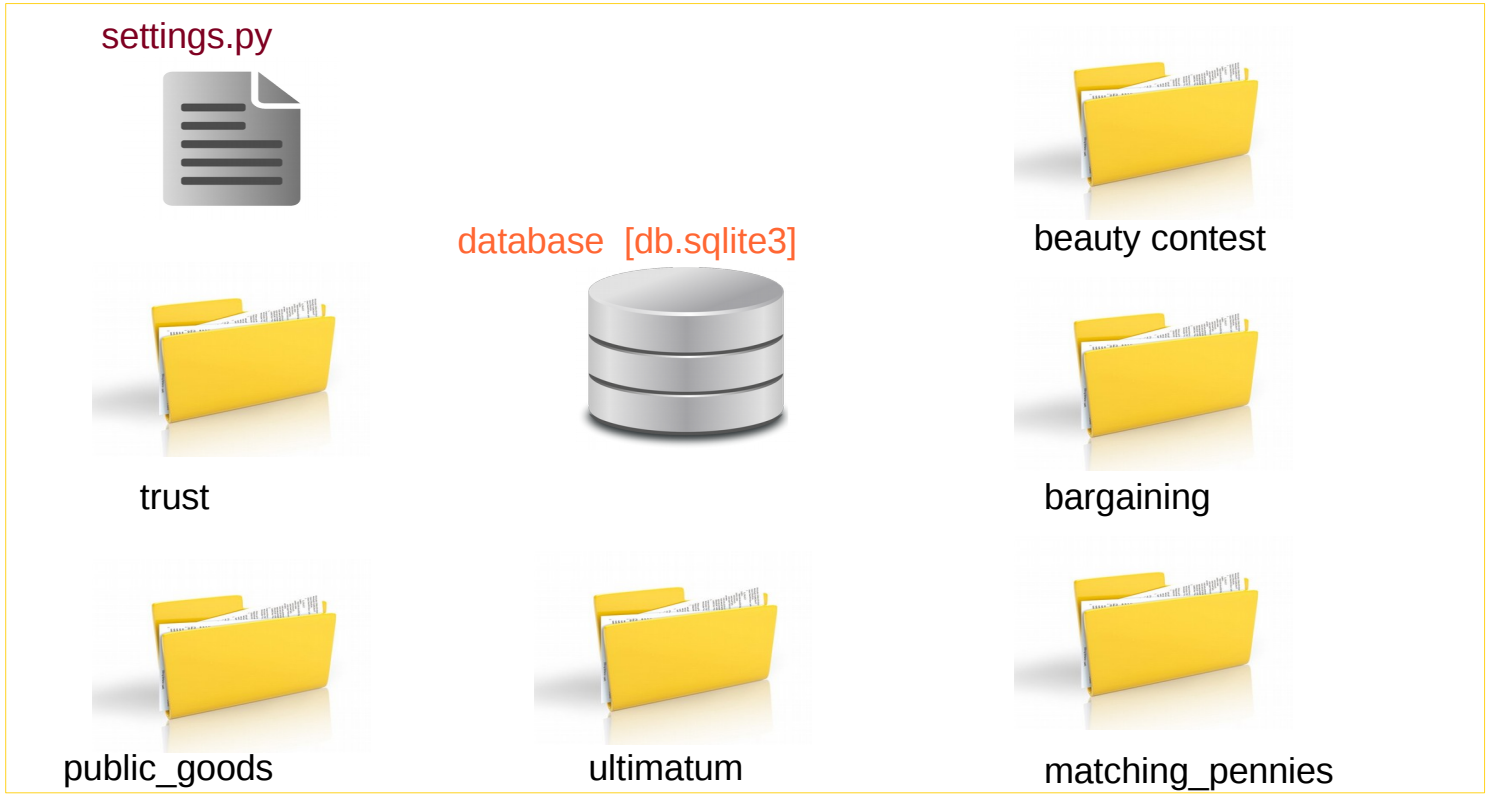
Otree Project Folder

Created with: `otree startproject project_name`



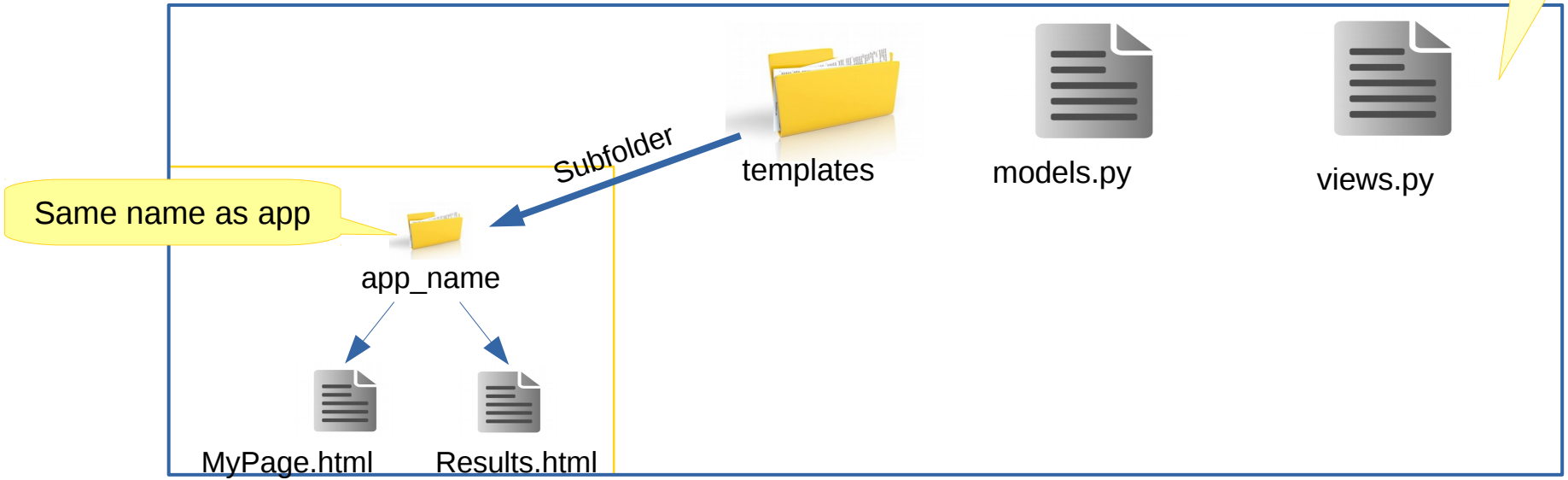
App Folder (e.g. trust)

- Contains a single game or survey
- New app is created when you run

```
otree startapp app_name
```

  - Creates a folder: app_name with the below structure

More files and folders are created but at this point not important to you.



settings.py

- Controls the whole project (language settings, Currency settings, ...)
- Contains a list of all available apps (games, surveys) in this otree project
- If an otree app is <u>not registered in the settings.py it will not show up</u> in your admin menu!

Registering an app / experiment

- In the settings.py find: `SESSION_CONFIG = [` and insert:

```
{
  'name': 'matching_pennies',          name of experiment
  'display_name': "Matching Pennies",    name to show in admin menu
  'num_demo_participants': 2,          number of demo participants
  'app_sequence': ['matching_pennies','survey'],  list of apps for experiment
},
```

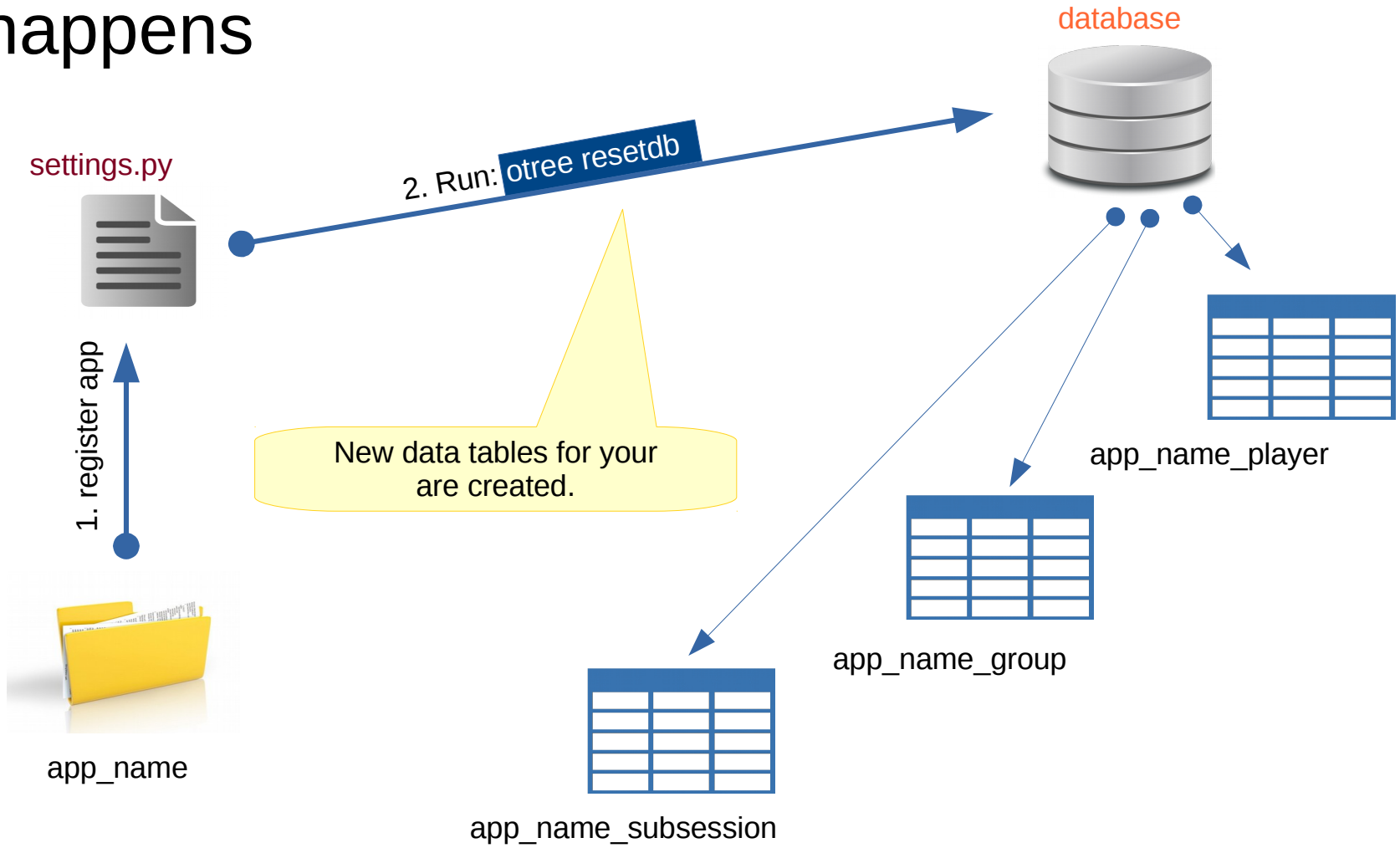
(example, “matching pennies”)

database [db.sqlite3/PostgreSQL]

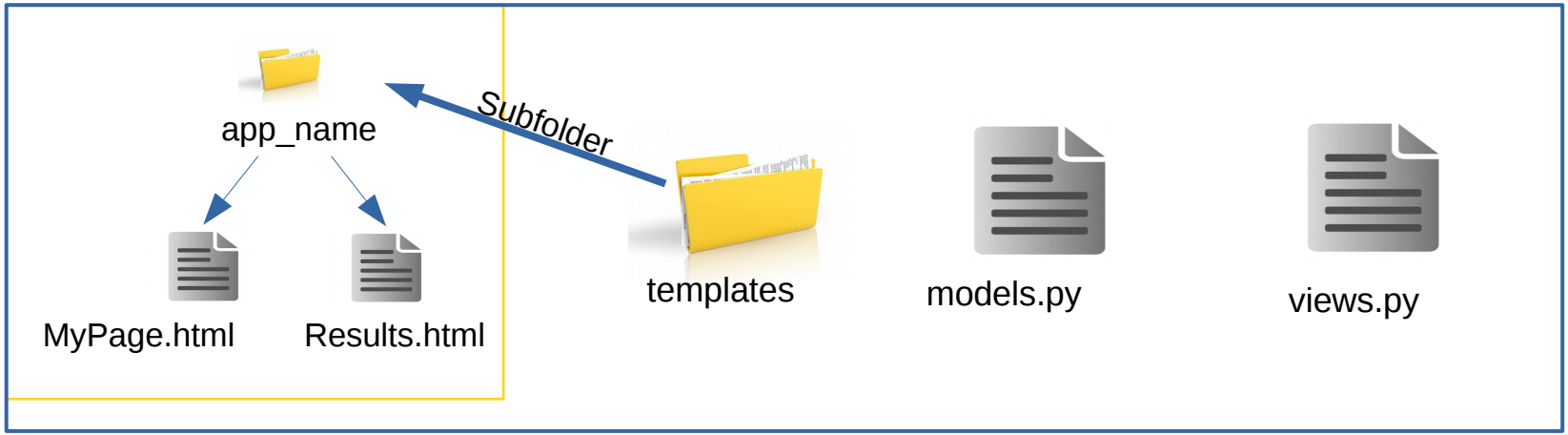
(background knowledge)

- Heart of an otree project
- Gets created after you run `otree resetdb`
- Contains parameters to control otree
- Stores data of app (experiments) in data tables
- Only registered apps are connected to the database

What happens



Let’s look in the app



models.py

- Controls variables in the app
- Where functions are defined
- Contains 4 main classes
  - Constants – constant value (endowments, multipliers, etc.)
  - Subsession – functions run before session starts
  - Group – group level functions, group variables stored in database
  - Player – player level functions, player variables stored in database

Create a variable for database

```
Class Player(BasePlayer):

    sent_amount = models.CurrencyField()
```

Creates column in data table app_name_player of name sent_amount.
For the creation to work the app has to be registered and you have to reset the database with `otree resetdb`

Available field types:

- CurrencyField – float values with currency extension (defined in settings.py).
- IntegerField – whole number field
- FloatField – decimal numbers
- CharField – text variables
- BooleanField – “Yes / No” (True/False) fields

Options when creating database variables

- `choices = [1,2,3,4,…]` - limits choices for subject; default creates drop down menu in app
- `blank = True` – allows for empty answers by subjects
- `initial = False` – show no starting value
- `verbose_name = “How much ...”` - label shown next to form field to subject
- `widget = widgets.RadioSelectHorizontal()` - horizontal line of radio buttons instead of drop menu for choices (other widgets available)

views.py

- Structures the app and makes form fields and variables available to the template files (html) that the user sees

- Contains one class per html-template with the identical name of type Page
  (e.g. SendResult.html : `class SendResult(Page):`)

- Contains one class per waiting page of type WaitPage
  (e.g. `class Wait_for_A(WaitPage):`)

- Contains list `page_sequence=[ SendResult, Wait, …]` that controls the order of shown templates

Predifined functions in views.py

Outside of template

```
def vars_for_all_templates(self):
    return{
            “endowment”:Constants.endowment,
            “rounds”:Contstants.num_rounds,
            … }
```

Returns dictionary of variables available in ALL templates. E.g., number of rounds is available in all templates via `{{rounds}}`.

Inside template

```
def vars_for_template(self):
    return{
            “prior”:player.prior,
            “random”:player.random(),
            … }
```

Returns dictionary of variables available in THIS template. E.g., random number is available in this template via `{{random}}` for each player separately.

Predifined functions in views.py II

Inside template

```
def is_displayed(self):
    return self.player.id_in_group == 1
```

Shows page to player **IF THE PLAYER'S ID IS 1**.

```
def before_next_page(self):
    self.player.check_minimum()
```

Do something before next page loads. Here: the player's minimum offer before next page.

```
def after_all_players_arrive(self):
    self.group.calculate_all_payoff()
```

Do something after all players have reached this stage. Here: set payoff for all. Mainly used on Waitpages, as these advance automatically if result is TRUE.

Create form fields

In views.py

```
Class Send(Page):
    form_model = models.Group
```
– tells template to which class in models.py the form fields in the list correspond to (here Group; alternatively Player)

```
form_fields = ['sent_amount', …]
```
- list of form fields used in this template

In Send.html (templates/app_name)

```
{% formfield group.sent_amount with label = "How much ..."%}
```

Creates a form field in the html website.
If you set `verbose_name ="..."` in the models.py for this field the part

template page (e.g., Send.html)

- Html file with django extensions

- `{% formfield models_class.field_name with label="Text next to field" %}` - creating form fields; part with label … is unnecessary if verbose_name is defined in models.py
- `{{var_name}}` — access variables form views

- `{% block title %} … {% endblock %}` - title area of page shown to subject
- `{% block content %} … {% endlock %}` - content area
- `{% block scripts %} … {% endblock %}` - area for JavaScripts and CSS
- `{% next_button %}` - create standardized button leading to next page

views.py ~/Documents/Lehre/16_WiSe/Project Module/otre...

```
class SendBack(Page):
    """This page is only for P2
    P2 sends back some amount (of the tripled amount received)
to P1"""
    form_model = models.Group
    form_fields = ['sent_back_amount']

    def is_displayed(self):
        return self.player.id_in_group == 2

    def vars_for_template(self):
        tripled_amount = self.group.sent_amount *
Constants.multiplication_factor

        return {
                'tripled_amount': tripled_amount,
                'prompt':
                    'Please enter a number from 0 to %s:' %
tripled_amount}
```

Make available to template

SendBack.html ~/Documents/Lehre/16_WiSe/Project Module/ot.....

```
{% extends "global/Base.html" %}
{% load staticfiles otree_tags %}

{% block title %}
    Your Choice
{% endblock %}

{% block content %}
<p>
You are Participant B. Participant A sent you
{{group.sent_amount}} and you received {{tripled_amount}}. Now
you have {{tripled_amount}}. How many points will you send to
participant A?
</p>

    {% formfield group.sent_back_amount with label=prompt %}
    <p>{% next_button %}</p>
    {% include Constants.instructions_template %}

{% endblock %}
```

Creates available form field

Demo | Trust Game: se... | Please wait | Your Choice

127.0.0.1:8000/p/tu5oqwfj/trust/Se

Your Choice

Calls form field

You are Participant B. Participant A sent you 50 points and you received 150 points. Now you have 150 points. How many points will you send to participant A?

Please enter a number from 0 to 150 points:

points

Next

Instructions

You have been randomly and anonymously paired with another participant. One of you will be selected at random to be participant A; the other will be participant B. You will learn whether you are participant A or B prior to making any decision.

To start, participant A receives 100 points; participant B receives nothing. Participant A can send some or all of his 100 points to participant B. Before B receives these points they will be multiplied by 3. Once B receives the tripled points he can decide to send some or all of his points to A.

Debug info

Basic info	
ID in group	2
Group	1
Round number	1
Participant	P2
Participant label	

*models.py ~/Documents/Lehre/16_WiSe/Project Module/otre...

```
class Constants(BaseConstants):
    name_in_url = 'trust'
    players_per_group = 2
    num_rounds = 1

    instructions_template = 'trust/Instructions.html'

    # Initial amount a
    amount_allocated =
    multiplication_fa


class Subsession(Base
    pass


class Group(BaseGroup):
    sent_amount = models.Cu  cyField(
        min=0, max=Constant  mount_allocated,
        doc="""Amount sent  P1""",
    )

    sent_back_amount = models.CurrencyField(
        doc="""Amount sent back by P2""",
        min=c(0),
    )

    def set_payoffs(self):
        p1 = self.get_player_by_id(1)
        p2 = self.get_player_by_id(2)
        p1.payoff = Constants.amount_allocated -
self.sent_amount + self.sent_back_amount
        p2.payoff = self.sent_amount *
Constants.multiplication_factor - self.sent_back_amount


class Player(BasePlayer):

    def role(self):
        return {1: 'A', 2: 'B'}[self.id_in_group]
```

Creates data field for database, makes form field available to app

views.py
~/Documents/Lehre/16_WiSe/Project Module/otre...

```python
class SendBack(Page):
    """This page is only for P2
    P2 sends back some amount (of the tripled amount received)
to P1"""

    form_model = models.Group
    form_fields = ['sent_back_amount']

    def is_displayed(self):
        return self.player.id_in_group == 2

    def vars_for_template(self):
        tripled_amount = self.group.sent_amount *
Constants.multiplication_factor

        return {
                'tripled_amount': tripled_amount,
                'prompt':
                    'Please enter a number from 0 to %s:' %
tripled_amount}
```

Python | Tab Width: 8 | Ln 1, Col 1 | INS

SendBack.html
~/Documents/Lehre/16_WiSe/Project Module/ot.....

```html
{% extends "global/Base.html" %}
{% load staticfiles otree_tags %}

{% block title %}
    Your Choice
{% endblock %}

{% block content %}
<p>
You are Participant B. Participant A sent you
{{group.sent_amount}} and you received {{tripled_amount}}. Now
you have {{tripled_amount}}. How many points will you send to
participant A?
</p>

    {% formfield group.sent_back_amount with label=prompt %}
    <p>{% next_button %}</p>
    {% include Constants.instructions_template %}

{% endblock %}
```

HTML | Tab Width: 8 | Ln 9, Col 4 | INS

Demo | Trust Game: se... | Please wait | Your Choice

127.0.0.1:8000/p/tu5oqwfj/trust/Se

Your Choice

You are Participant B. Participant A sent you 50 points and you received 150 points. Now you have 150 points. How many points will you send to participant A?

Please enter a number from 0 to 150 points:

points

Next

Instructions

You have been randomly and anonymously paired with another participant. One of you will be selected at random to be participant A; the other will be participant B. You will learn whether you are participant A or B prior to making any decision.

To start, participant A receives 100 points; participant B receives nothing. Participant A can send some or all of his 100 points to participant B. Before B receives these points they will be multiplied by 3. Once B receives the tripled points he can decide to send some or all of his points to A.

Debug info

Basic info	
id_in_group	
Group	
Round number	1
Participant	P2
Participant label	

*models.py
~/Documents/Lehre/16_WiSe/Project Module/otre...

```python
class Constants(BaseConstants):
    name_in_url = 'trust'
    players_per_group = 2
    num_rounds = 1

    instructions_template = 'trust/Instructions.html'

    # Initial amount allocated to each player
    amount_allocated = c(100)
    multiplication_factor = 3


class Subsession(BaseSubsession):
    pass


class Group(BaseGroup):
    sent_amount = models.CurrencyField(
        min=0, max=Constants.amount_allocated,
        doc="""Amount sent by P1""",
    )

    sent_back_amount = models.CurrencyField(
        doc="""Amount sent back by P2""",
        min=c(0),
    )

    def set_payoffs(self):
        p1 = self.get_player_by_id(1)
        p2 = self.get_player_by_id(2)
        p1.payoff = Constants.amount_allocated -
self.sent_amount + self.sent_back_amount
        p2.payoff = self.sent_amount *
Constants.multiplication_factor - self.sent_back_amount


class Player(BasePlayer):

    def role(self):
        return {1: 'A', 2: 'B'}[self.id_in_group]
```

Python | Tab Width: 8 | Ln 63, Col 1 | INS

Calls upon models.py to get database connection
Writes value stored in database to subject screen

views.py ~/Documents/Lehre/16_WiSe/Project Module/otre...

```python
class SendBack(Page):
    """This page is only for P2
    P2 sends back some amount (of the tripled amount received)
to P1"""

    form_model = models.Group
    form_fields = ['sent_back_amount']

    def is_displayed(self):
        return self.player.id_in_group == 2

    def vars_for_template(self):
        tripled_amount = self.group.sent_amount *
Constants.multiplication_factor

        return {
            'tripled_amount': tripled_amount,
            'prompt':
                'Please enter a number from 0 to %s:' %
tripled_amount}
```

Python | Tab Width: 8 | Ln 1, Col 1 | INS

Makes available to template

SendBack.html ~/Documents/Lehre/16_WiSe/Project Module/ot.....

```{% extends "global/Base.html" %}
{% load staticfiles otree_tags %}

{% block title %}
    Your Choice
{% endblock %}

{% block content %}
<p>
You are Participant B. Participant A sent you
{{group.sent_amount}} and you received {{tripled_amount}}. Now
you have {{tripled_amount}}. How many points will you send to
participant A?
</p>

    {% formfield group.sent_back_amount with label=prompt %}
    <p>{% next_button %}</p>
    {% include Constants.instructions_template %}

{% endblock %}
```

HTML | Tab Width: 8 | Ln 9, Col 4 | INS

Demo | Trust Game: se... | Please wait | Your Choice

127.0.0.1:8000/p/tu5oqwfj/trust/Se

Grabs sent_amount value from database and multiplier from Contansts class.
Multiplies it and stores in var tripled_amount.

Your Choice

You are Participant B. Participant A sent you 50 points and you received 150 points. Now you have 150 points. How many points will you send to participant A?

Please enter a number from 0 to 150 points:

points

Next

Instructions

You have been randomly and anonymously paired with another participant. One of you will be selected at random to be participant A; the other will be participant B. You will learn whether you are participant A or B prior to making any decision.

To start, participant A receives 100 points; participant B receives nothing. Participant A can send some or all of his 100 points to participant B. Before B receives these points they will be multiplied by 3. Once B receives the tripled points he can decide to send some or all of his points to A.

Debug info

Shows tripled amount to subject.

!!! tripled amount is NOT stored in database here!!!

Basic info

ID in group	2
Group	1
Round number	1
Participant	P2
Participant label	

*models.py ~/Documents/Lehre/16_WiSe/Project Module/otre...

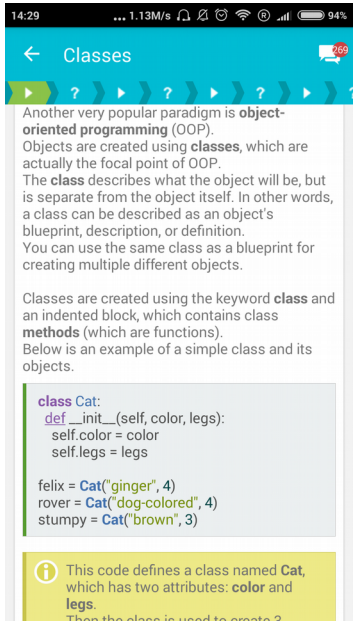
```python
class Constants(BaseConstants):
    name_in_url = 'trust'
    players_per_group = 2
    num_rounds = 1

    instructions_template = 'trust/Instructions.html'

    # Initial amount allocated to each player
    amount_allocated = c(100)
    multiplication_factor = 3


class Subsession(BaseSubsession):
    pass


class Group(BaseGroup):
    sent_amount = models.CurrencyField(
        min=0, max=Constants.amount_allocated,
        doc="""Amount sent by P1""",
    )

    sent_back_amount = models.CurrencyField(
        doc="""Amount sent back by P2""",
        min=c(0),
    )

    def set_payoffs(self):
        p1 = self.get_player_by_id(1)
        p2 = self.get_player_by_id(2)
        p1.payoff = Constants.amount_allocated -
self.sent_amount + self.sent_back_amount
        p2.payoff = self.sent_amount *
Constants.multiplication_factor - self.sent_back_amount


class Player(BasePlayer):

    def role(self):
        return {1: 'A', 2: 'B'}[self.id_in_group]
```

Python | Tab Width: 8 | Ln 63, Col 1 | INS

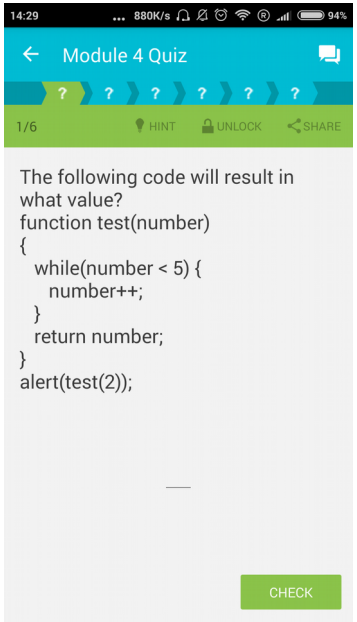
Hints: Learn to program

SoloLearn.com

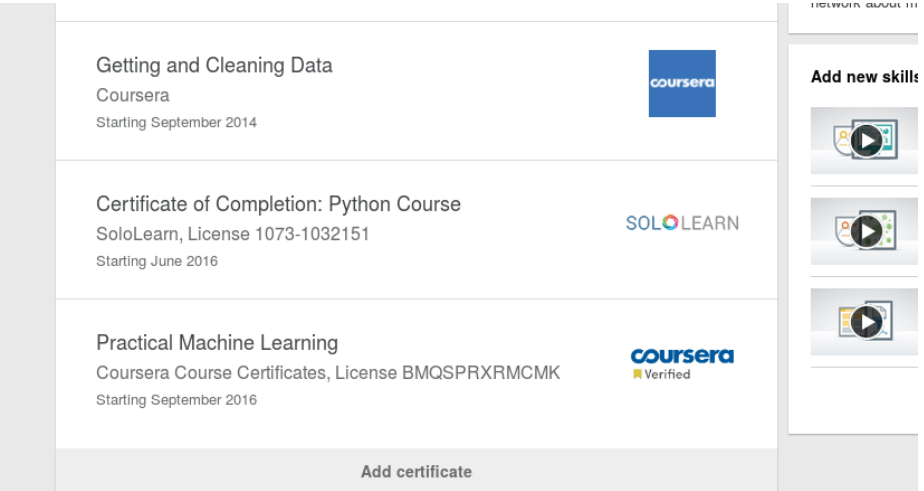
Python



JavaScript



- Free mobile app
- Finished courses – certificates for LinkedIn



Programming Editors

Closed Source IDEs

- Sublime text , PyCharm , Eclipse , Visual Studio
  - Pro: Well maintained, optimized, key-ready
  - Con: paid – usually with evaluation duration

Open Source IDEs

- Bluefish , Ninja IDE , [***Vim , Emacs***]
  - Pro: Free, do what you need, highly configurable
  - Con: Probably not as powerful as the above one

  [***Special case***]: extremely powerful with very large user community, BUT very steep learning curve. Need a lot of practice and personal configuration before you are effective.

Git

Track your changes and collaborate

Commits on Oct 27, 2016

item list update, cheatsheet
cataclysmic committed 4 days ago

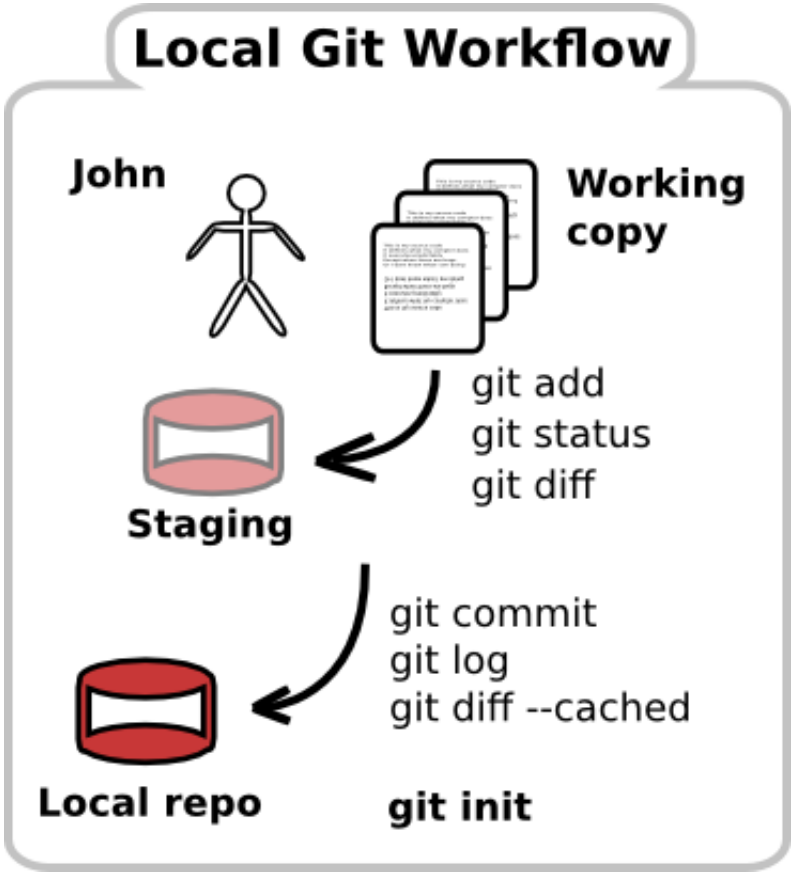
753103a

Commits on Oct 26, 2016

slides final
cataclysmic committed 5 days ago

slides and cheatsheet
cataclysmic committed 5 days ago

```
 otree startapp trust_c
@@ -88,7 +94,7 @@ otree startapp trust_c

 1. Truster receives endowment
 2. Truster sends nothing/part/all of endowment to trustee
-3. Experimenter increases money by some multiplier
+3. Experimenter (software) increases money by some multiplier
 4. Trustee (hopefully) sends money back to truster
 5. Both receive respective payoff

@@ -108,7 +114,7 @@ otree startapp trust_c
     :reveal_background: #ffffff
     :END:

-#+attr_html: :width 400px
+#+attr_html: :width 600px
 [[./join.png]]
```



- Used for collaborative work
- Works for text file based formats (html, py, css, js, tex, txt, …)
- Tracks your changes
- Tracks who changes what
- Enables parallel working, while ensuring that nothing get deleted.
- Local and online databases ensure data security and sharing

- Youtube channel with great videos jckelley2

Expert level

Command Cheatsheet

Git & Github Workflow

blog.jr0cket.co.uk