```python
#!/usr/bin/env python
"""
type will select all objects of that type
selector will select a specific group within an object type
unique will get at a specific object in a selector
"""
import time
from pages import formfield

class user(object):
  def __init__(self, _UID=None, _name=None, _email=None, _default_foods=[], _provide=[]):
    self.type = 'user'
    self.UID =  str(_UID) # unique
    self.name = _name
    self.selector = self.name
    self.ts = None

    # formfield constructor is var_name, human_name, help text (optional), type (defaults to normal), role
    self.form_fields = [

      # creation = only appears when creating master object for the first time
      # editmaster = only appears when editing a master
      # edit = only appears when editing a unique object
      # admin = only appears when admin viewing

      formfield('UID','Unique ID','Don\'t touch this unless you\'re Lucas','normal','admin'),
      formfield('name','Username','Don\'t be a faggot and choose something stupid','normal','creation'),
      formfield('email','Email Address','Privacy policy? how bout I spam the shit out of you','normal','creation, editmaster, edit'),
      formfield('default_foods','Foods that by default <b>only you</b> consume','e.g. for Lucas, he would put \'grape juice, roast beef\', etc.','large', 'creation, editmaster, edit'),
      formfield('food_owned','Food Owned','List of food you have directly purchased. (Comma seperated)','large','admin'),
      formfield('food_consumed','Food Consumed','List of food you have consumed. (Comma seperated)','large','admin'),
      formfield('meals_owned','Meals Cooked','List of meals you have cooked. (Comma seperated)','large','admin'),
      formfield('meals_consumed','Meals Eaten','List of meals you have eaten. (Comma seperated)','large','admin'),
      formfield('receipts_payed','Receipts Owned','List of receipts you have payed for. (Comma seperated)','large','admin')
    ]

    self.email = _email
    # pw (maybe later?)

    # what they eat (list of food selectors)
    self.default_foods = _default_foods

    self.food_owned = []
    self.food_consumed = []

    self.meals_owned = []
    self.meals_consumed = []

    self.receipts_payed = []

  def sync(self):
    self.selector = self.name

  def key(self):
    self.selector = self.name
    return str(self.UID) + '.' + str(self.selector) + '.' + str(self.type)

  def print_default_food(self):
    return ', '.join(self.consume)

  def set_default_vals(self):
    self.form_fields[0].default_val = self.UID
    self.form_fields[1].default_val = self.name
    self.form_fields[2].default_val = self.email
    self.form_fields[3].default_val = self.default_foods
    self.form_fields[4].default_val = self.food_owned
    self.form_fields[5].default_val = self.food_consumed
    self.form_fields[6].default_val = self.meals_owned
    self.form_fields[7].default_val = self.receipts_payed
    self.form_fields[8].default_val = self.meals_consumed


class food(object):
  def __init__(self, _UID=None, _name=None, _price=None):
    self.type = 'food'
    self.UID  = str(_UID) # unique
    self.name = _name # selector
    self.selector = self.name
    self.ts = None

    # formfield constructor is var_name, human_name, help text (optional), type (defaults to normal)
    self.form_fields = [
      formfield('UID','Unique ID','Don\'t touch this unless you\'re Lucas','normal','admin'),
      formfield('name','Name of food','e.g. \'apple\' or \'rocket grenade\'','normal', 'creation'),
      formfield('category','Categories','e.g. For milk, this could be something like \'dairy, drink\'','large','creation, editmaster, edit'),
      formfield('price','Price','Price for one of this type of food, e.g. \'$20.00\'','normal', 'creation, editmaster, edit'),
      formfield('owner','Owner','Who paid for this?','normal', 'edit'),
      formfield('eaten','Eaten / consumed','Has been eaten or consumed, Has <b>not</b> been eaten or consumed, ','boolean', 'edit')
    ]

    self.price = None
    self.owner = None # who paid, user
    self.eaten = False # has this been used?
    self.category = [] # list of strings, e.g. 'milk' would be under 'dairy'

  def sync(self):
    self.selector = self.name

  def key(self):
    self.selector = self.name
    return str(self.UID) + '.' + str(self.selector) + '.' + str(self.type)

  def set_default_vals(self):
    self.form_fields[0].default_val = self.UID
    self.form_fields[1].default_val = self.name
    self.form_fields[2].default_val = self.category
    self.form_fields[3].default_val = self.price
    self.form_fields[4].default_val = self.owner
    self.form_fields[5].default_val = self.eaten

class meal(object):
  def __init__(self, _UID=None, _name=''):
    self.type = 'meal'
    self.UID = _UID # unique
    self.name = _name # selector
    self.selector = self.name
    self.ts = None

    # formfield constructor is var_name, human_name, help text (optional), type (defaults to normal)
    self.form_fields = [
      formfield('UID','Unique ID','Don\'t touch this unless you\'re Lucas','normal','admin'),
      formfield('name','Name of meal','e.g. \'chicken parm\ or \'genital herpes\'','normal', 'creation'),
      formfield('ingrediants','Ingrediants','Comma seperated list of foods in this dish. <b>IMPORTANT:</b> For now, if the dish uses more than one quantity of some food, enter it in that many times. e.g., if \'chicken parm\' has two boxes of spaghetti, do \'spaghetti, spaghetti\'','large','creation, editmaster, edit'),
      formfield('participants','Who is eating at the meal','Comma seperated list of users who will eat the meal.','normal', 'creation, editmaster, edit'),
      formfield('cooks','Chefs','Comma seperated list of people responsible for cooking this meal.','normal', 'editmaster, edit, creation'),
      formfield('date','Date','What day will this meal be eaten <b>MUST BE IN THIS FORM:</b> 9/9/2011','normal', 'edit'),
      formfield('eaten','Has this meal been eaten?','Yes, No','boolean', 'edit')
    ]

    self.participants = [] # list of user
    self.price = None
    self.eaten = None
    self.cooks = [] # list user
    self.date = None # date it is gonna be cooked / eaten
    self.ingrediants = [] # list of food

  def sync(self):
    self.selector = self.name

  def key(self):
    self.selector = self.name
    return str(self.UID) + '.' + str(self.selector) + '.' + str(self.type)

  def set_default_vals(self):
    self.form_fields[0].default_val = self.UID
    self.form_fields[1].default_val = self.name
    self.form_fields[2].default_val = self.ingrediants
    self.form_fields[3].default_val = self.participants
    self.form_fields[4].default_val = self.cooks
    self.form_fields[5].default_val = self.date
    self.form_fields[6].default_val = self.eaten

class menu(object):
  def __init__(self, _UID=None, start_date=''):
    self.type = 'menu'
    self.UID = _UID
    self.start_date = start_date
    self.selector = self.start_date
    self.ts = None

    self.form_fields = [
      formfield('UID','Unique ID','Don\'t touch this unless you\'re Lucas','normal','admin'),
      formfield('start_date','Start Date','What date will this menu be starting? <b>MUST BE IN THIS FORM:</b> 9/9/2011', 'normal', 'creation, edit'),
      formfield('meals','Meals','What meals are on the menu?','large', 'creation, editmaster, edit'),
      formfield('done','Done<br>Is the menu over with?','Yes, No','boolean', 'edit')
    ]

    #self.people
    self.meals = [] # meals for the week, maybe change this to dict
    self.done = None

  def sync(self):
    self.selector = self.UID

  def key(self):
    self.selector = self.name
    return str(self.UID) + '.' + str(self.selector) + '.' + str(self.type)

  def set_default_vals(self):
    self.form_fields[0].default_val = self.UID
    self.form_fields[1].default_val = self.start_date
    self.form_fields[2].default_val = self.meals
    self.form_fields[3].default_val = self.done


class receipt(object):
  def __init__(self, _UID=None, purchase_date=''):
    self.type = 'receipt'
    self.UID = _UID
    self.purchase_date = purchase_date
    self.selector = self.purchase_date
    self.ts = None

    self.form_fields = [
      formfield('UID','Unique ID','Don\'t touch this unless you\'re Lucas','normal','admin'),
      formfield('purchase_date','Purchase Date','When did you buy this? <b>MUST BE IN THIS FORM:</b> 9/9/2011', 'normal', 'creation, editmaster, edit'),
      formfield('owner','Who paid?','User who paid for the receipt','normal',
creation, editmaster, edit'),
      formfield('contents','Contents','What foods are on the receipt','large',
 'creation, editmaster, edit')
    ]

    self.owner = None # user who paid
    self.contents = [] # list of food

  def sync(self):
    self.selector = self.start_date

  def key(self):
    self.selector = self.start_date
    return str(self.UID) + '.' + str(self.selector) + '.' + str(self.type)

  def set_default_vals(self):
    self.form_fields[0].default_val = self.UID
    self.form_fields[1].default_val = self.purchase_date
    self.form_fields[2].default_val = self.owner
    self.form_fields[3].default_val = self.contents

```