Zmodyfikowałem część kodu dla walidacji elementów formularza w pliku views.py (imie, email):

```
from django.shortcuts import render
import re

def index(request):
    return render(request, 'appmonday/index.html')

def form(request):
    return render(request, 'appmonday/form.html')

def show(request):
    if request.POST:
        if 'imie' in request.POST and 'email' in request.POST:
            imie = request.POST.get("imie", "").strip()
            email = request.POST.get("email", "").strip()

    errors = {}
    if not imie:
        errors['imie'] = "Imię jest wymagane"
    elif len(imie) < 2:
        errors['imie'] = "Imię jest za krótkie"
    elif not re.match(r"^[A-Z]{1}[a-złąćśźżęóń]{1,}$", imie):
        errors['imie'] = "Pierwsza duża litera, reszta małe"

    if not email:
        errors['email'] = "Email jest wymagany"
    elif not re.match(r"^[^@\s]+@[^@\s]+\.[^@\s]+$", email):
        errors['email'] = "np. jan@wp.pl"


    if errors:
        return render(request, 'appmonday/form.html', {
            "errors": errors,
            "imie": imie,
            "email": email,
            })
    else:
        return render(request, 'appmonday/show.html', {
                "imie": imie,
                "email": email,
                })
```

Dokonaj zmian w kodzie, sprawdź poprawność wykonanych zmian.

Wyjaśnienie dla wyrażenia regularnego użytego w walidacji email:

^ – **początek** całego tekstu (kotwica).
[^@\s]+ – **co najmniej jeden** znak, który **nie jest** @ i **nie jest** spacją/białym znakiem.
[...] – klasa znaków (zestaw dozwolonych).
^ **wewnątrz nawiasów** – negacja (czyli „wszystko **oprócz** …”).
@ – dosłowna małpa.
\s – biały znak (spacja, tab, nowa linia itd.).
+ – jeden lub więcej powtórzeń.

@ – dosłowna małpa oddzielająca lokalną część od domeny.
[^@\s]+ – znów „jeden lub więcej znaków, które nie są @ ani białe”.
\. – dosłowna kropka (ucieczka \ sprawia, że to kropka, a nie „dowolny znak”).
[^@\s]+ – końcówka domeny (np. pl, com, info), znów „bez @ i bez białych znaków”.
$ – **koniec** całego tekstu (kotwica).

**Zadanie:**

Przygotuj formularz dla klienta, który będzie musiał wpisać takie dane jak:

imię, nazwisko, email, numer telefonu, miasto, kod pocztowy, ulica, numer domu, pesel

Zadbaj o poprawną walidację wszystkich pól z poziomu języka Python.

**Wskazówki:**

Normalizuj dane (obcinaj spacje, usuwaj myślniki z telefonu) zanim przystąpisz do walidacji.

Regex ma być prosty i czytelny — ma łapać typowe błędy, nie wszystkie możliwe przypadki.

Nowy materiał - obiekt typu radio (z dodaną właściwością checked)

Do form.html dodajemy:

```
<label><input type="radio" name="wyksztalcenie" value="Podstawowe" checked>Podstawowe</label>
<label><input type="radio" name="wyksztalcenie" value="Średnie">Średnie</label>
<label><input type="radio" name="wyksztalcenie" value="Wyższe">Wyższe</label>
```

Do show.html dodajemy:

```
<p><strong>Wykształcenie:</strong> {{ wyksztalcenie }}</p>
```

Do views.py dodajemy:

```
wyksztalcenie = request.POST.get("wyksztalcenie")
return render(request, "podjeden/show1.html", {
    "first_name": first_name,
    "last_name": last_name,
    "email": email,
    "phone": phone,
    "wyksztalcenie": wyksztalcenie,
})
```

Nowy materiał - obiekt typu radio (bez właściwości checked)

Zmiana w form.html:

```
<label><input type="radio" name="wyksztalcenie" value="Podstawowe">Podstawowe</label>
<label><input type="radio" name="wyksztalcenie" value="Średnie">Średnie</label>
<label><input type="radio" name="wyksztalcenie" value="Wyższe">Wyższe</label>
```

Zmiana w views.py:

```
errors = {}
if not first_name: errors["first_name"] = "Imię jest wymagane."
if not last_name:  errors["last_name"]  = "Nazwisko jest wymagane."
if not email:      errors["email"]      = "E-mail jest wymagany."
if not phone:      errors["phone"]      = "Telefon jest wymagany."
if not wyksztalcenie:   errors["wyksztalcenie"] = "Zaznacz wykształcenie."
```

```
if errors:
    return render(request, "podjeden/form1.html", {
        "errors": errors,
        "first_name": first_name,
        "last_name": last_name,
        "email": email,
        "phone": phone,
        "wyksztalcenie": wyksztalcenie,
    })
```

Nowy materiał - obiekt typu checkbox

---

Do form.html dodajemy:

```
<label><input type="checkbox" name="hobby" value="czytanie"> Czytanie</label>
<label><input type="checkbox" name="hobby" value="bieganie"> Bieganie</label>
<label><input type="checkbox" name="hobby" value="gotowanie"> Gotowanie</label>
```

Do show.html dodajemy:

```
<ul>
    {% if hobby %}
    <b>Twoje zainteresowania</b>
    {% endif %}
    {% for h in hobby %}
    <li>{{ h }}</li>
    {% empty %}
    <i>Brak zainteresowań</i>
    {% endfor %}
</ul>
```

Do views.py dodajemy:

```
hobby = request.POST.getlist("hobby")

return render(request, "podjeden/show1.html", {
    "first_name": first_name,
    "last_name": last_name,
    "email": email,
    "phone": phone,
    "wyksztalcenie": wyksztalcenie,
    "hobby": hobby
})
```

Zadania:

1. Wykonaj przedstawione przykłady z użyciem obiektów typu radio oraz checkbox.

2. Wykonaj ankietę muzyczną. Zbuduj formularz z imieniem, e-mailem, radiem „ulubiony gatunek” (3–4 opcje) i grupą checkboxów „ulubieni wykonawcy” (min. 5, wybierz co najmniej 2). Zweryfikuj e-mail i pokaż podsumowanie wyborów w estetycznej formie.