

# Python Podstawy – Intel

infoShare Academy

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Nagłówek</h1>
    <p>Paragraf tekstu.</p>
  </body>
</html>
```



```
<ul>
  <li>Czerwony</li>
  <li>Zielony</li>
  <li>Niebieski</li>
</ul>
```

- Czerwony
- Zielony
- Niebieski

```
<ol>
  <li>Otworzyć opakowanie</li>
  <li>Wsypać do wody i rozmieszać</li>
  <li>Podgrzewać aż do zagotowania</li>
</ol>
```

1. Otworzyć opakowanie
2. Wsypać do wody i rozmieszać
3. Podgrzewać aż do zagotowania

```
<dl>
  <dt>Warzywa</dt>
  <dd>Rośliny zielne (jednoroczne, dwuletnie lub
wieloletnie), które mogą służyć człowiekowi za pokarm.
Mogą być spożywane w całości lub tylko częściowo
(nasiona, kwiatostany, owoce, pędy, liście).</dd>
  <dt>Owoce</dt>
  <dd>Jadalne części drzew lub krzewów będące z punktu
widzenia morfologii roślin owocami lub owocostanami.
Rośliny uprawne, z których się je uzyskuje nazywane są
drzewami i krzewami owocowymi.</dd>
  <dt>Nabiał</dt>
  <dd>Produkty spożywcze pochodzenia mlecznego: sery,
twarogi, jogurty, kefiry, śmietany, serki homogenizowane
itp., a także jajka</dd>
</dl>
```

#### Warzywa

Rośliny zielne (jednoroczne, dwuletnie lub wieloletnie), które mogą służyć człowiekowi za pokarm. Mogą być spożywane w całości lub tylko częściowo (nasiona, kwiatostany, owoce, pędy, liście, korzenie).

#### Owoce

Jadalne części drzew lub krzewów będące z punktu widzenia morfologii roślin owocami lub owocostanami. Rośliny uprawne, z których się je uzyskuje nazywane są drzewami i krzewami owocowymi.

#### Nabiał

Produkty spożywcze pochodzenia mlecznego: sery, twarogi, jogurty, kefiry, śmietany, serki homogenizowane itp., a także jajka

```
<ul>
  <li>Instalacja
    <ol>
      <li>Elementy pakietu</li>
      <li>Podłączenie okablowania</li>
      <li>Ustawienie anteny</li>
    </ol>
  </li>
  <li>Regulacja
    <ol>
      <li>Dostrajanie kanałów</li>
      <li>Poprawa jakości obrazu</li>
      <li>Poprawa jakości dźwięku</li>
    </ol>
  </li>
</ul>
```

- Instalacja
  1. Elementy pakietu
  2. Podłączenie okablowania
  3. Ustawienie anteny
- Regulacja
  1. Dostrajanie kanałów
  2. Poprawa jakości obrazu
  3. Poprawa jakości dźwięku

```

```



```
<table>
  <caption>Twój koszyk</caption>
  <tr>
    <th>Produkt</th>
    <th>Ilość</th>
    <th>Cena za sztukę</th>
    <th>Razem</th>
  </tr>
  <tr>
    <td>Skórzana torebka</td>
    <td>1</td>
    <td>100.00PLN</td>
    <td>100.00PLN</td>
  </tr>
  <tr>
    <td>Portfel</td>
    <td>2</td>
    <td>25.00PLN</td>
    <td>50.00PLN</td>
  </tr>
</table>
```

| Twój koszyk      |       |           |           |
|------------------|-------|-----------|-----------|
| Produkt          | Ilość | Cena      | Razem     |
| Skórzana torebka | 1     | 100.00PLN | 100.00PLN |
| Portfel          | 2     | 25.00PLN  | 50.00PLN  |

```
<h1>Główny tytuł</h1>
  <h2>Nagłówek drugiego rzędu</h2>
  <p>Treść paragrafu</p>
  <h3>Nagłówek trzeciego rzędu</h3>
  <p>Treść paragrafu</p>
  <h3>Kolejny nagłówek trzeciego rzędu</h3>
  <p>Treść paragrafu</p>
```

# Główny tytuł

## Nagłówek drugiego rzędu

Treść paragrafu

### Nagłówek trzeciego rzędu

Treść paragrafu

### Kolejny nagłówek trzeciego rzędu

Treść paragrafu

selektor

własność

wartość

**p { color: pink; }**

deklaracja

```
body {  
    background-color: #d0e4fe;  
}  
  
h1 {  
    color: orange;  
    text-align: center;  
}  
  
p {  
    font-family: "Times New Roman";  
    font-size: 20px;  
}
```

html Alt H


```
<b>
  Tekst na czerwono
</b>
<p>
  Wyśrodkowany paragraf
</p>

```

Result

**Tekst na czerwono**

Wyśrodkowany paragraf



CSS Alt C

```
b {
  color: red;
}
p {
  text-align: center;
  text-decoration: underline;
}
img {
  border: 2px green dotted;
  padding: 5px;
  margin: 0 auto 0 auto;
  display: block;
}
```

> **pip install Flask**

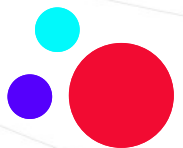
(...)

> **flask run** (domyślnie port 5000, app.py)

> **flask run --host=0.0.0.0 --port=8080**

> **export FLASK\_ENV=development; flask run**

> **export FLASK\_APP=aplikacja.py; flask run**



# Flask – aplikacja

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"
```



# Flask – routing

```
@app.route("/")  
def hello():  
    return 'Hello World!'
```

```
@app.route("/witaj")  
def hello():  
    return 'Witaj!'
```

```
@app.route("/witaj/<imie>")  
def hello(imie):  
    return 'Witaj {}'.format(imie)
```



# Flask – konwersja typów

```
@app.route("/kwadrat/<int:liczba>")  
def kwadrat(num):  
    return f'Kwadrat liczby {num} to {num ** 2}'
```

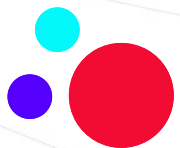
Wszystkie parametry są stringiem, stąd może zająć konieczność ich konwersji.

|       |                                |
|-------|--------------------------------|
| int   | l. całkowite                   |
| float | l. zmiennoprzecinkowe          |
| path  | string ale akceptuje slashe \/ |
| any   | próbuję dobrać typ             |
| uuid  | stringi UUID                   |



# Flask – metody HTTP

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        #ZALOGUJ MNIE
        pass
    else:
        #POKAŻ FORMULARZ LOGOWANIA
        pass
```



# Flask – generowanie URL

```
from flask import url_for

@app.route('/')
def index(): pass

@app.route('/login')
def login(): pass

@app.route('/user/<username>')
def profile(username): pass

print url_for('index')
/

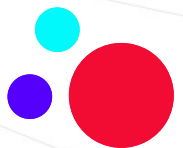
print url_for('login')
/login

print url_for('login', next='/')
/login?next=/

print url_for('profile', username='John Doe')
/user/John%20Doe
```

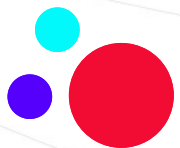
Diagram illustrating the mapping between Flask routes and generated URLs using `url_for`:

- `@app.route('/') → print url_for('index')` results in `/`
- `@app.route('/login') → print url_for('login')` results in `/login`
- `@app.route('/login') → print url_for('login', next='/')` results in `/login?next=/'`
- `@app.route('/user/<username>') → print url_for('profile', username='John Doe')` results in `/user/John%20Doe`



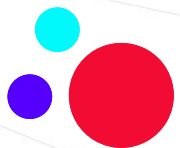
# Flask – szablony

```
@app.route('/')  
@app.route('/list')  
def list():  
    blogi = Blog.get_all_blogs()  
    return render_template('list.html', blogi=blogi)
```



# Jinja – system szablonów

```
<!doctype html>
<html>
  <head>
    <title>Hello from Flask</title>
  </head>
  <body>
    {% if name %}
      <h1>Hello {{ name }}!</h1>
    {% else %}
      <h1>Hello, World!</h1>
    {% endif %}
  </body>
</html>
```



# Jinja – rozszerzanie szablonów

## index.html

```
<!doctype html>
<html>
  <head>
    <title>Hello Flask</title>
  </head>
  <body>
    {% block content %}
      Domyślna wartość
    {% endblock %}
  </body>
</html>
```

## list.html

```
{% extends „index.html” %}
```

```
{% block content %}
```

Zmieniona wartość

```
{% endblock %}
```

<https://www.sqlalchemy.org/>

DZIEKUJĘ NA DZIŚ  
Python Podstawy – Intel