# Dynamic Routing

In this lesson, we will learn how to create dynamic URL routes in our application.

WE'LL COVER THE FOLLOWING



## Introduction

In the last lesson, we studied *views* and *routes*. Also, we learned about the different parameters of the `route` decorator. In static routing, the `rule` parameter of the `route` decorator was a simple string. However, in dynamic routing, the `rule` parameter is not a constant string. Instead, a **variable** rule is passed to the `route()`. Let's figure out how this is done!

## Variable rules

In `Flask` we can add **variable rules** inside the URL route by using the following syntax: `<variable_name>`. The variable called `variable_name` will then be passed to the *view* function to be used.

### An example using variable rules

```
"""An example application to demonstrate Variable Rules in Routing"""
from flask import Flask
app = Flask(__name__)
```

```
@app.route("/")
def home():
    """View for the Home page of the website."""
    return "Welcome to the HomePage!"


@app.route("/<my_name>")
def greatings(my_name):
    """View function to greet the user by name."""
    return "Welcome "+ my_name +"!"


if __name__ == "__main__":
    app.run(debug=True, host="0.0.0.0", port=3000)
```

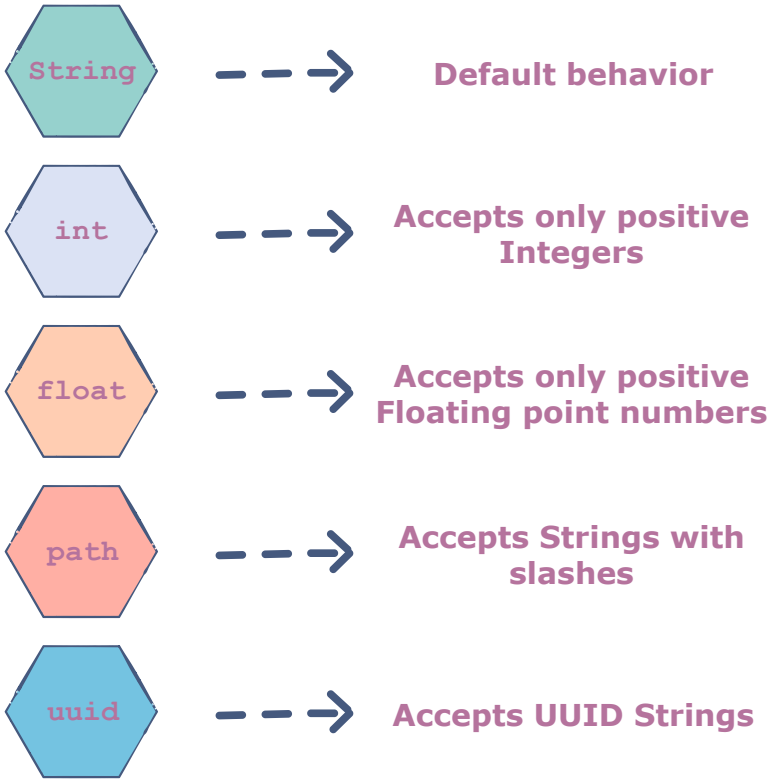
> **Try this:** press **"RUN"** and then open the mini-application given above in a separate tab. Append "/" + **your name** to the URL and observe the output!

### Explanation #

In the mini-application given above, we can observe that in the second **view function** called `greetings()`, we have used the variable rule: `/<my_name>` in **line #12**. In this URL, `my_name` is the name of a variable. This variable is then passed as a parameter to the `greetings()` function in **line #13**. Finally, the variable `my_name` is then used in **line #15** to return a greeting to the user.

## Converter #

In the previous example, the variable `my_name` was extracted from the URL. Then this variable was *converted* into a string and passed to the function `greetings()` to be used. This is the default behavior of the **converter**. Converters can convert the following data types:




Types of Converters

> **Note:** If you are unfamiliar with **UUID objects** in Python, please refer to the official documentation.

## An Example using dynamic routing #

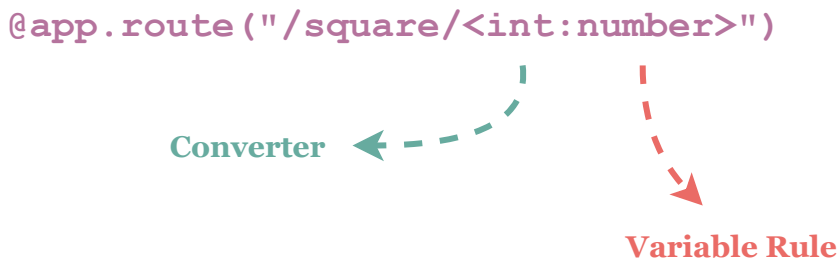
```
"""An example application to demonstrate Dynamic Routing"""
from flask import Flask
app = Flask(__name__)

@app.route("/")
def home():
    """View for the Home page of the Website"""
    return "Welcome to the HomePage!"


@app.route('/square/<int:number>')
def show_square(number):
    """View that shows the square of the number passed by URL"""
    return "Square of "+ str(number) +" is: "+ str(number * number)

if __name__ == "__main__":
    app.run(debug=True, host="0.0.0.0", port=3000)
```

> **Try this:** open the mini-application given above in a separate tab.

> Append **“/square/121”** to the URL and observe the output. ***Try with different numbers as well!***

## Explanation #

In the example above, there is a view function called `show_square()`. This view function uses a **variable rule** as well as a **converter** for this rule.




- **variable rule:** `number`
- **converter:** `int`

---

In the next lesson, we will start working on our course project! Get ready for some hands-on experience.