ValueError

```
ValueError: invalid literal for int() with base 10: ''
```

Traceback (most recent call last)

File "C:\Users\Juan Pinto\AppData\Local\Programs\Python\Python313\Lib\site-packages\flask\app.py", line *1498*, in `__call__`

```
    ) -> cabc.Iterable[bytes]:
        """The WSGI server calls the Flask application object as the
        WSGI application. This calls :meth:`wsgi_app`, which can be
        wrapped to apply middleware.
        """
        return self.wsgi_app(environ, start_response)
               ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

File "C:\Users\Juan Pinto\AppData\Local\Programs\Python\Python313\Lib\site-packages\flask\app.py", line *1476*, in `wsgi_app`

```
            try:
                ctx.push()
                response = self.full_dispatch_request()
            except Exception as e:
                error = e
                response = self.handle_exception(e)
                           ^^^^^^^^^^^^^^^^^^^^^^^^
            except:  # noqa: B001
                error = sys.exc_info()[1]
                raise
            return response(environ, start_response)
        finally:
```

File "C:\Users\Juan Pinto\AppData\Local\Programs\Python\Python313\Lib\site-packages\flask\app.py", line *1473*, in `wsgi_app`

```
        ctx = self.request_context(environ)
        error: BaseException | None = None
        try:
            try:
                ctx.push()
                response = self.full_dispatch_request()
                           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
            except Exception as e:
                error = e
                response = self.handle_exception(e)
            except:  # noqa: B001
                error = sys.exc_info()[1]
```

File "C:\Users\Juan Pinto\AppData\Local\Programs\Python\Python313\Lib\site-packages\flask\app.py", line *882*, in `full_dispatch_request`

```
            request_started.send(self, _async_wrapper=self.ensure_sync)
            rv = self.preprocess_request()
            if rv is None:
                rv = self.dispatch_request()
        except Exception as e:
            rv = self.handle_user_exception(e)
                 ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
        return self.finalize_request(rv)

    def finalize_request(
        self,
        rv: ft.ResponseReturnValue | HTTPException,
```

File "C:\Users\Juan Pinto\AppData\Local\Programs\Python\Python313\Lib\site-packages\flask\app.py", line *880*, in `full_dispatch_request`

```

        try:
            request_started.send(self, _async_wrapper=self.ensure_sync)
            rv = self.preprocess_request()
            if rv is None:
                rv = self.dispatch_request()
                     ^^^^^^^^^^^^^^^^^^^^^^^
        except Exception as e:
            rv = self.handle_user_exception(e)
        return self.finalize_request(rv)

    def finalize_request(
```

File "C:\Users\Juan Pinto\AppData\Local\Programs\Python\Python313\Lib\site-packages\flask\app.py", line *865*, in `dispatch_request`

```
            and req.method == "OPTIONS"
        ):
            return self.make_default_options_response()
        # otherwise dispatch to the handler for that endpoint
        view_args: dict[str, t.Any] = req.view_args  # type: ignore[assignment]
        return self.ensure_sync(self.view_functions[rule.endpoint])(**view_args)
  # type: ignore[no-any-return]
               ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^


    def full_dispatch_request(self) -> Response:
        """Dispatches the request and on top of that performs request
        pre and postprocessing as well as HTTP exception catching and
        error handling.
```

File "C:\xampp\htdocs\estudiantes_inc\app.py", line *41*, in `registro_estudiantes`

```
        segundo_nombre = request.form.get('segundo_nombre', '')
        primer_apellido = request.form['primer_apellido']
        segundo_apellido = request.form.get('segundo_apellido', '')
        correo = request.form['correo']
        fecha_inscripcion = request.form['fecha_inscripcion']
        carrera = int(request.form['carrera'])  # Convertir a entero
                  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
        foto = request.files['foto']
        documento = request.files['documento_estudiante']  # Cambiado para manejar
  el archivo PDF


        # Guardar la foto en la carpeta 'uploads'
        if foto:
```

ValueError: invalid literal for int() with base 10: ''

The debugger caught an exception in your WSGI application. You can now look at the traceback which led to the error.

To switch between the interactive traceback and the plaintext one, you can click on the "Traceback" headline. From the text traceback you can also create a paste of it. For code execution mouse-over the frame you want to debug and click on the console icon on the right side.

You can execute arbitrary Python code in the stack frames and there are some extra helpers available for introspection:

- `dump()` shows all variables in the frame
- `dump(obj)` dumps all that's known about the object

Brought to you by **DON'T PANIC**, your friendly Werkzeug powered traceback interpreter.