builtins.TypeError

```
TypeError: Object of type 'int64' is not JSON serializable
```

Traceback (most recent call last)

File "/Users/gta/anaconda3/lib/python3.6/site-packages/flask/app.py", line *1997*, in __call__

```
return self.wsgi_app(environ, start_response)
```

File "/Users/gta/anaconda3/lib/python3.6/site-packages/flask/app.py", line *1985*, in wsgi_app

```
response = self.handle_exception(e)
```

File "/Users/gta/anaconda3/lib/python3.6/site-packages/flask/app.py", line *1540*, in handle_exception

```
reraise(exc_type, exc_value, tb)
```

File "/Users/gta/anaconda3/lib/python3.6/site-packages/flask/_compat.py", line *33*, in reraise

```
raise value
```

File "/Users/gta/anaconda3/lib/python3.6/site-packages/flask/app.py", line *1982*, in wsgi_app

```
response = self.full_dispatch_request()
```

File "/Users/gta/anaconda3/lib/python3.6/site-packages/flask/app.py", line *1614*, in full_dispatch_request

```
rv = self.handle_user_exception(e)
```

File "/Users/gta/anaconda3/lib/python3.6/site-packages/flask/app.py", line *1517*, in handle_user_exception

```
reraise(exc_type, exc_value, tb)
```

File "/Users/gta/anaconda3/lib/python3.6/site-packages/flask/_compat.py", line *33*, in reraise

```
raise value
```

File "/Users/gta/anaconda3/lib/python3.6/site-packages/flask/app.py", line *1612*, in full_dispatch_request

```
rv = self.dispatch_request()
```

File "/Users/gta/anaconda3/lib/python3.6/site-packages/flask/app.py", line *1598*, in dispatch_request

```
        # request came with the OPTIONS method, reply automatically
        if getattr(rule, 'provide_automatic_options', False) \
           and req.method == 'OPTIONS':
            return self.make_default_options_response()
        # otherwise dispatch to the handler for that endpoint
        return self.view_functions[rule.endpoint](**req.view_args)
```

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

File "/Users/gta/Desktop/uscdatabootcamp/hw-9/app_temperature.py", line *27*, in tobs

```
    temp_query = session.query(Hawaii.tobs).filter(Hawaii.date_format >= (dt.date.today() - dt.timedelta(days=365))).all()

    flat_temp = list(np.ravel(temp_query))

    return jsonify(flat_temp)

    #   Produces an error that reads 'TypeError: key Timestamp('2016-08-23 00:00:00') is not a string'
    #   The error may be due to to the fact that the timestamp is a datetime object and not a string.
    # https://github.com/grantaguinaldo/uscdatabootcamp/blob/master/Misc_Files/app_temp_error.pdf

```

File "/Users/gta/anaconda3/lib/python3.6/site-packages/flask/json.py", line *263*, in jsonify

```
(dumps(data, indent=indent, separators=separators), '\n'),
```

File "/Users/gta/anaconda3/lib/python3.6/site-packages/flask/json.py", line *123*, in dumps

```
rv = _json.dumps(obj, **kwargs)
```

File "/Users/gta/anaconda3/lib/python3.6/json/__init__.py", line *238*, in dumps

```
**kw).encode(obj)
```

File "/Users/gta/anaconda3/lib/python3.6/json/encoder.py", line *201*, in encode

```
chunks = list(chunks)
```

File "/Users/gta/anaconda3/lib/python3.6/json/encoder.py", line *428*, in _iterencode

```
yield from _iterencode_list(o, _current_indent_level)
```

File "/Users/gta/anaconda3/lib/python3.6/json/encoder.py", line *325*, in _iterencode_list

```
yield from chunks
```

File "/Users/gta/anaconda3/lib/python3.6/json/encoder.py", line *437*, in _iterencode

```
o = _default(o)
```

File "/Users/gta/anaconda3/lib/python3.6/site-packages/flask/json.py", line *80*, in default

```
return _json.JSONEncoder.default(self, o)
```

File "/Users/gta/anaconda3/lib/python3.6/json/encoder.py", line *180*, in default

```
o.__class__.__name__)
```

TypeError: Object of type 'int64' is not JSON serializable

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.