Python flask.jsonify() Examples

The following are code examples for showing how to use *flask.jsonify()*. They are from open source Python projects. You can vote up the examples you like or vote down the ones you don't like.

Example 1

Project: *BASS* Author: *Cisco-Talos* File: *server.py* GNU General Public License v2.0 7 vo

```
def whitelist_add():
    log.info("whitelist_add called")
    try:
        file_ = request.files["file"]
        handle, filename = tempfile.mkstemp()
        os.close(handle)
        file_.save(filename)
        data = request.get_json()
        if data and "functions" in data:
            functions = data["functions"]
        else:
            functions = None
        bass.whitelist_add(filename, functions)
        os.unlink(filename)
    except KeyError:
        log.exception("")
        return make_response(jsonify(message = "Sample file 'file' missing in POS

    return jsonify(message = "OK")
```

Example 2

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *basic.py* Apache License 2.0 7 vo

```
def test_make_response_with_response_instance(self):
        app = flask.Flask(__name__)
        with app.test_request_context():
            rv = flask.make_response(
                flask.jsonify({'msg': 'W00t'}), 400)
            self.assertEqual(rv.status_code, 400)
            self.assertEqual(rv.data, b'{\n  "msg": "W00t"\n}')
            self.assertEqual(rv.mimetype, 'application/json')

            rv = flask.make_response(
                flask.Response(''), 400)
            self.assertEqual(rv.status_code, 400)
            self.assertEqual(rv.data, b'')
            self.assertEqual(rv.mimetype, 'text/html')

            rv = flask.make_response(
                flask.Response('', headers={'Content-Type': 'text/html'}),
                400, [('X-Foo', 'bar')])
            self.assertEqual(rv.status_code, 400)
            self.assertEqual(rv.headers['Content-Type'], 'text/html')
            self.assertEqual(rv.headers['X-Foo'], 'bar')
```

Example 3

6 vo

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *basic.py* Apache License 2.0

```
def test_make_response_with_response_instance(self):
        app = flask.Flask(__name__)
        with app.test_request_context():
            rv = flask.make_response(
                flask.jsonify({'msg': 'W00t'}), 400)
            self.assertEqual(rv.status_code, 400)
            self.assertEqual(rv.data, b'{\n  "msg": "W00t"\n}')
            self.assertEqual(rv.mimetype, 'application/json')

            rv = flask.make_response(
                flask.Response(''), 400)
            self.assertEqual(rv.status_code, 400)
            self.assertEqual(rv.data, b'')
            self.assertEqual(rv.mimetype, 'text/html')

            rv = flask.make_response(
                flask.Response('', headers={'Content-Type': 'text/html'}),
                400, [('X-Foo', 'bar')])
            self.assertEqual(rv.status_code, 400)
            self.assertEqual(rv.headers['Content-Type'], 'text/html')
            self.assertEqual(rv.headers['X-Foo'], 'bar')
```

Example 4

Project: *hydrus* Author: *HTTP-APIs* File: *resources.py* MIT License 5 vo

```
def get(self) -> Response:
        """Return main entrypoint for the api."""
        return set_response_headers(jsonify(get_doc().entrypoint.get()))
```

Example 5

Project: *hydrus* Author: *HTTP-APIs* File: *resources.py* MIT License 5 vo

```
def get(self) -> Response:
        """Return the main hydra vocab."""
        return set_response_headers(jsonify(get_doc().generate()))
```

Example 6

Project: *hydrus* Author: *HTTP-APIs* File: *resources.py* MIT License 5 vo

```
def get(self) -> Response:
        """Return application main Entrypoint."""
        response = {"@context": get_doc().entrypoint.context.generate()}
        return set_response_headers(jsonify(response))
```

Example 7

Project: *hydrus* Author: *HTTP-APIs* File: *resources.py* MIT License 5 vo

```
def get(self, id_: str, path: str) -> Response:
        """
        GET object with id = id_ from the database.
        :param id_ : Item ID
        :param path : Path for Item ( Specified in APIDoc @id)
        """
        id_ = str(id_)
        auth_response = check_authentication_response()
        if isinstance(auth_response, Response):
            return auth_response

        class_type = get_doc().collections[path]["collection"].class_.title
        # Get path of the collection-class
        class_path = get_doc().collections[path]["collection"].class_.path

        if checkClassOp(class_path, "GET"):
            # Check if class_type supports GET operation
            try:
                # Try getting the Item based on ID and Class type
                response = crud.get(
                    id_,
                    class_type,
                    api_name=get_api_name(),
                    session=get_session())

                response = finalize_response(class_path, response)
                return set_response_headers(
                    jsonify(hydrafy(response, path=path)))

            except (ClassNotFound, InstanceNotFound) as e:
                error = e.get_HTTP()
                return set_response_headers(jsonify(error.generate()), status_cod
        abort(405)
```

Example 8

Project: *hydrus* Author: *HTTP-APIs* File: *resources.py* MIT License 5 vo

```
def delete(self, id_: str, path: str) -> Response:
        """Delete object with id=id_ from database."""
        id_ = str(id_)
        auth_response = check_authentication_response()
        if isinstance(auth_response, Response):
            return auth_response

        class_type = get_doc().collections[path]["collection"].class_.title
        # Get path of the collection-class
        class_path = get_doc().collections[path]["collection"].class_.path

        if checkClassOp(class_path, "DELETE"):
            # Check if class_type supports PUT operation
            try:
                # Delete the Item with ID == id_
                crud.delete(id_, class_type, session=get_session())
                method = "DELETE"
                resource_url = "{}{}/{}/{}".format(
                    get_hydrus_server_url(), get_api_name(), path, id_)
                last_job_id = crud.get_last_modification_job_id(session=get_sessio
                new_job_id = crud.insert_modification_record(method, resource_url,
                                                             session=get_session()
                send_sync_update(socketio=socketio, new_job_id=new_job_id,
                                 last_job_id=last_job_id, method=method,
                                 resource_url=resource_url)
                status_description = "Object with ID {} successfully deleted".forn
                status = HydraStatus(code=200, title="Object successfully deleted.
                                     desc=status_description)
                return set_response_headers(jsonify(status.generate()))
            except (ClassNotFound, InstanceNotFound) as e:
                error = e.get_HTTP()
                return set_response_headers(jsonify(error.generate()), status_cod

        abort(405)
```

Example 9

Project: *hydrus* Author: *HTTP-APIs* File: *resources.py* MIT License 5 vo

```
def delete(self, path: str) -> Response:
        """
        Method executed for DELETE requests.
        Used to delete a non-collection class.
        :param path - Path for Item ( Specified in APIDoc @id)
        """
        auth_response = check_authentication_response()
        if isinstance(auth_response, Response):
            return auth_response

        endpoint_ = checkEndpoint("DELETE", path)
        if not endpoint_['method']:
            abort(endpoint_['status'])
        elif path in get_doc().parsed_classes and "{}Collection".format(
                path) not in get_doc().collections:
            # No Delete Operation for collections
            try:
                class_type = get_doc().parsed_classes[path]['class'].title
                crud.delete_single(class_type, session=get_session())
                method = "DELETE"
                resource_url = "{}{}/{}".format(
                    get_hydrus_server_url(), get_api_name(), path)
                last_job_id = crud.get_last_modification_job_id(session=get_sessic
                new_job_id = crud.insert_modification_record(method, resource_url,
                                                             session=get_session()
                send_sync_update(socketio=socketio, new_job_id=new_job_id,
                                 last_job_id=last_job_id, method=method,
                                 resource_url=resource_url)
                status = HydraStatus(code=200, title="Object successfully added")
                return set_response_headers(jsonify(status.generate()))
            except (ClassNotFound, InstanceNotFound) as e:
                error = e.get_HTTP()
                return set_response_headers(
                    jsonify(error.generate()), status_code=error.code)
```

Example 10

Project: *hydrus* Author: *HTTP-APIs* File: *resources.py* MIT License 5 vo

```
def delete(self, path, int_list):
        """
        To delete multiple objects
        :param path: endpoints
        :param int_list: Optional String containing ',' separated ID's
        :return:
        """
        auth_response = check_authentication_response()
        if isinstance(auth_response, Response):
            return auth_response
        class_type = get_doc().collections[path]["collection"].class_.title
        if checkClassOp(class_type, "DELETE"):
            # Check if class_type supports PUT operation
            try:
                # Delete the Item with ID == id_
                crud.delete_multiple(int_list, class_type, session=get_session())
                method = "DELETE"
                path_url = "{}{}/{}".format(
                    get_hydrus_server_url(), get_api_name(), path)
                last_job_id = crud.get_last_modification_job_id(session=get_sessic
                id_list = int_list.split(',')
                for item in id_list:
                    resource_url = path_url + item
                    new_job_id = crud.insert_modification_record(method, resource_
                                                                 session=get_sessi
                    send_sync_update(socketio=socketio, new_job_id=new_job_id,
                                     last_job_id=last_job_id, method=method,
                                     resource_url=resource_url)
                    last_job_id = new_job_id
                status_description = "Objects with ID {} successfully deleted".for
                    id_list)
                status = HydraStatus(code=200, title="Objects successfully deletec
                                     desc=status_description)
                return set_response_headers(jsonify(status.generate()))

            except (ClassNotFound, InstanceNotFound) as e:
                error = e.get_HTTP()
                return set_response_headers(jsonify(error.generate()), status_cod

        abort(405)
```

Example 11

Project: *hydrus* Author: *HTTP-APIs* File: *auth.py* MIT License 5 vo

```
def token_response(token: str) -> Response:
    """
    Return succesful token generation object
    """
    message = {200: "User token generated"}
    response = set_response_headers(jsonify(message), status_code=200,
                                    headers=[{'X-Authorization': token}])
    return response
```

Example 12

Project: *hydrus* Author: *HTTP-APIs* File: *auth.py* MIT License 5 vo

```
def failed_authentication(incorrect: bool) -> Response:
    """
    Return failed authentication object.
    """
    if not incorrect:
        message = {401: "Need credentials to authenticate"}
        realm = 'Basic realm="Login required"'
    else:
        message = {401: "Incorrect credentials"}
        realm = 'Basic realm="Incorrect credentials"'
    nonce = create_nonce(get_session())
    response = set_response_headers(jsonify(message), status_code=401,
                                    headers=[{'WWW-Authenticate': realm},
                                                      {'X-Authentication': nonce}])
    return response
```

Example 13

Project: *hydrus* Author: *HTTP-APIs* File: *auth.py* MIT License 5 vo

```
def verify_user() -> Union[Response, None]:
    """
    Verify the credentials of the user and assign token.
    """
    try:
        auth = check_authorization(request, get_session())
        if auth is False:
            return failed_authentication(True)
        elif get_token():
            token = add_token(request, get_session())
            return token_response(token)
    except Exception as e:
        error = e.get_HTTP()  # type: HydraError
        return set_response_headers(jsonify(error.generate()), status_code=error.
    return None
```

Example 14

Project: *Mastering-Python-Networking-Second-Edition* Author: *PacktPublishing*
File: *chapter9_6.py* MIT License 5 vo

```
def get_devices():
    return jsonify({'device': [device.get_url()
                               for device in Device.query.all()]})
```

Example 15

Project: *Mastering-Python-Networking-Second-Edition* Author: *PacktPublishing*
File: *chapter9_6.py* MIT License 5 vo

```
def get_device(id):
    return jsonify(Device.query.get_or_404(id).export_data())
```

Example 16

Project: *Mastering-Python-Networking-Second-Edition* Author: *PacktPublishing*
File: *chapter9_6.py* MIT License 5 vo

```
def new_device():
    device = Device()
    device.import_data(request.json)
    db.session.add(device)
    db.session.commit()
    return jsonify({}), 201, {'Location': device.get_url()}
```

Example 17

Project: *Mastering-Python-Networking-Second-Edition* Author: *PacktPublishing*
File: *chapter9_6.py* MIT License 5 vo

```
def edit_device(id):
    device = Device.query.get_or_404(id)
    device.import_data(request.json)
    db.session.add(device)
    db.session.commit()
    return jsonify({})
```

Example 18

Project: *Mastering-Python-Networking-Second-Edition* Author: *PacktPublishing*
File: *chapter9_7.py* MIT License
5 vc

```
def get_devices():
    return jsonify({'device': [device.get_url()
                               for device in Device.query.all()]})
```

Example 19

Project: *Mastering-Python-Networking-Second-Edition* Author: *PacktPublishing*
File: *chapter9_7.py* MIT License
5 vc

```
def get_device_version(id):
    device = Device.query.get_or_404(id)
    hostname = device.hostname
    ip = device.mgmt_ip
    prompt = hostname+"#"
    result = show_version(hostname, prompt, ip, 'cisco', 'cisco')
    return jsonify({"version": str(result)})
```

Example 20

Project: *Mastering-Python-Networking-Second-Edition* Author: *PacktPublishing*
File: *chapter9_7.py* MIT License
5 vc

```
def get_role_version(device_role):
    device_id_list = [device.id for device in Device.query.all() if device.role ==
    result = {}
    for id in device_id_list:
        device = Device.query.get_or_404(id)
        hostname = device.hostname
        ip = device.mgmt_ip
        prompt = hostname + "#"
        device_result = show_version(hostname, prompt, ip, 'cisco', 'cisco')
        result[hostname] = str(device_result)
    return jsonify(result)
```

Example 21

Project: *Mastering-Python-Networking-Second-Edition* Author: *PacktPublishing*
File: *chapter9_7.py* MIT License
5 vc

```
def edit_device(id):
    device = Device.query.get_or_404(id)
    device.import_data(request.json)
    db.session.add(device)
    db.session.commit()
    return jsonify({})
```

Example 22

Project: *Mastering-Python-Networking-Second-Edition* Author: *PacktPublishing*
File: *chapter9_5.py* MIT License

5 vo

```
def interface(hostname, interface_number):
    return jsonify(name=hostname, interface=interface_number)
```

Example 23

Project: *BASS* Author: *Cisco-Talos* File: *server.py* GNU General Public License v2.0 5 vo

```
def job_create():
    try:
        job = bass.create_job()
        return jsonify(message = "ok", job = job.json())
    except Exception as ex:
        return make_response(jsonify(message = str(ex), trace = traceback.format_
```

Example 24

Project: *BASS* Author: *Cisco-Talos* File: *server.py* GNU General Public License v2.0 5 vo

```
def jobs_list():
    return jsonify(message = "ok", jobs = [j.json() for j in bass.list_jobs()])
```

Example 25

Project: *BASS* Author: *Cisco-Talos* File: *server.py* GNU General Public License v2.0 5 vo

```
def job_get_status(job_id):
    try:
        return jsonify(message = "ok", job = bass.get_job(job_id).json())
    except KeyError:
        return make_response(jsonify(message = "Invalid job id"), 400)
    except Exception as ex:
        return make_response(jsonify(message = str(ex), trace = traceback.format_
```

Example 26

Project: *BASS* Author: *Cisco-Talos* File: *server.py* GNU General Public License v2.0 5 vo

```
def job_add_sample(job_id):
    try:
        samples = []
        for name, file_ in request.files.items():
            handle, filename = tempfile.mkstemp()
            os.close(handle)
            file_.save(filename)
            samples.append(bass.get_job(job_id).add_sample(filename, name))
        return jsonify(message = "ok", samples = [s.json() for s in samples])
    except KeyError:
        log.exception("Invalid job id")
        return make_response(jsonify(message = "Invalid job id"), 400)
```

Example 27

Project: *BASS* Author: *Cisco-Talos* File: *server.py* GNU General Public License v2.0 5 vo

```
def job_submit(job_id):
    try:
        bass.submit_job(job_id)
        return jsonify(message = "ok")
    except KeyError:
        return make_response(jsonify(message = "Invalid job id"), 400)
```

Example 28

Project: *BASS* Author: *Cisco-Talos* File: *server.py* GNU General Public License v2.0 5 vo

```
def function_get(fid):
    global Session
    session = Session()
    try:
        function = session.query(Function).filter(Function.id == fid).one()
        return make_response(jsonify(**json.loads(function.data)), 200)
    except NoResultFound:
        return make_response(jsonify(message = "Function not found"), 404)
```

Example 29

Project: *BASS* Author: *Cisco-Talos* File: *server.py* GNU General Public License v2.0 5 vo

```
def function_raw_hash_get():
    global Session
    session = Session()
    filename, file_ = request.files.items()[0]
    db = Database(pickle.load(file_))

    arch_name = db.architecture_name
    if arch_name == "metapc":
        arch_name = "x86"
    try:
        arch = session.query(Architecture).filter(Architecture.name == arch_name a
                Architecture.bits == db.architecture_bits and \
                Architecture.little_endian == db.architecture_endianness == "littl
    except NoResultFound:
        return make_response(jsonify(message = "Architecture not found"), 404)

    try:
        func = next(db.functions)
    except StopIteration:
        return make_response(jsonify(message = "No function found in database"),

    raw_hash = _function_calculate_raw_sha256(func)
    size = _function_get_size(func)

    try:
        function = session.query(Function).filter(Function.raw_sha256 == raw_hash
                Function.size == size and \
                Function.arch == arch.id).one()
        return make_response(jsonify(**json.loads(function.data)), 200)
    except NoResultFound:
        return make_response(jsonify(message = "Function not found"), 404)
```

Example 30

Project: *BASS* Author: *Cisco-Talos* File: *server.py* GNU General Public License v2.0 5 vo

```
def function_mnem_hash_get():
    global Session
    session = Session()
    filename, file_ = request.files.items()[0]
    db = Database(pickle.load(file_))

    arch_name = db.architecture_name
    if arch_name == "metapc":
        arch_name = "x86"
    try:
        arch = session.query(Architecture).filter(Architecture.name == arch_name a
                Architecture.bits == db.architecture_bits and \
                Architecture.little_endian == db.architecture_endianness == "litt]
    except NoResultFound:
        return make_response(jsonify(message = "Architecture not found"), 404)

    try:
        func = next(db.functions)
    except StopIteration:
        return make_response(jsonify(message = "No function found in database"),

    mnem_hash = _function_calculate_mnem_sha256(func)

    try:
        function = session.query(Function).filter(Function.mnem_sha256 == mnem_has
                Function.arch == arch.id).one()
        return make_response(jsonify(**json.loads(function.data)), 200)
    except NoResultFound:
        return make_response(jsonify(message = "Function not found"), 404)
```

Example 31

Project: *BASS* Author: *Cisco-Talos* File: *ida_service.py* GNU General Public License v2.0 5 vc

```
def bindiff_export():
    """
    Run the IDA Pro autoanalysis on the input file and export a BinExport database
    :param input: The input file
    :return: Status code 200 and a JSON object containing the output database
        name in key 'output', or status code 422 on invalid parameters, 408 on
        timeout or 500 on other errors.
    """
    logger.info("bindiff_export called")

    directory = None
    try:
        directory = tempfile.mkdtemp()
        if len(request.files) != 1:
            return make_response(jsonify(error = "Missing file parameter"), 422)

        filename, file_ = request.files.items()[0]
        input_ = os.path.join(directory, sanitize_filename(filename))
        file_.save(input_)

        output = os.path.join(directory, "output.BinExport")

        timeout = request.form.get('timeout', None)
        is_64_bit = request.form.get('is_64_bit', True)
        try:
            run_ida(input_, is_64_bit, timeout, os.path.join(PREFIX, "export_binex
            logger.info("Command completed successfully")
            return send_file(open(output, "rb"), as_attachment = True, attachment_
        except TimeoutError:
            return jsonify(error = "Program execution timed out"), 408
        except OSError as err:
            return jsonify(error = "Program execution failed with error %d" % err

    finally:
        if directory is not None:
            shutil.rmtree(directory)
```

Example 32

Project: *BASS* Author: *Cisco-Talos* File: *ida_service.py* GNU General Public License v2.0 5 vo

```
def pickle_export():
    """
    Run the IDA Pro autoanalysis on the input file and export a BinExport database
    :param input: The input file
    :return: Status code 200 and a JSON object containing the output database
        name in key 'output', or status code 422 on invalid parameters, 408 on
        timeout or 500 on other errors.
    """
    logger.info("bindiff_export called")

    directory = None
    try:
        directory = tempfile.mkdtemp()
        if len(request.files) != 1:
            return make_response(jsonify(error = "Missing file parameter"), 422)

        filename, file_ = request.files.items()[0]
        input_ = os.path.join(directory, sanitize_filename(filename))
        file_.save(input_)

        output = os.path.join(directory, "output.pickle")

        timeout = request.form.get('timeout', None)
        is_64_bit = request.form.get('is_64_bit', False)
        try:
            run_ida(input_, is_64_bit, timeout, os.path.join(PREFIX, "export_binex
            logger.info("Command completed successfully")
            return send_file(open(output, "rb"), as_attachment = True, attachment_
        except TimeoutError:
            return jsonify(error = "Program execution timed out"), 408
        except OSError as err:
            return jsonify(error = "Program execution failed with error %d" % err
    finally:
        if directory is not None:
            shutil.rmtree(directory)
```

Example 33

Project: *BASS* Author: *Cisco-Talos* File: *ida_service.py* GNU General Public License v2.0 5 vo

```
def bindiff_compare():
    logger.info("bindiff_compare called")

    input_dir = tempfile.mkdtemp()
    output_dir = tempfile.mkdtemp()
    try:
        primary = os.path.join(input_dir, "primary")
        secondary = os.path.join(input_dir, "secondary")
        try:
            request.files["primary"].save(primary)
            request.files["secondary"].save(secondary)
        except KeyError:
            return make_response( jsonify(error="Missing parameter 'primary' or 's

        timeout = request.form.get('timeout', None)

        cmd = (BINDIFF_DIFFER, "--primary", primary, "--secondary", secondary, "--
        logger.info("Executing %s", " ".join("'%s'" % x for x in cmd))
        check_call(cmd, cwd = output_dir, timeout = timeout)
        db_path = [os.path.join(output_dir, x) for x in os.listdir(output_dir)]
        if len(db_path) != 1:
            return make_response( jsonify(error = "BinDiff generated 0 or several
        return send_file(open(db_path[0], "rb"), as_attachment = True, attachment_
    except OSError as err:
        if err.errno == -9:
            return make_response( jsonify(error = "Program execution timed out"),
        else:
            return make_response( jsonify(error = "Program execution failed with e
    finally:
        shutil.rmtree(input_dir)
        shutil.rmtree(output_dir)
```

Example 34

Project: *BASS* Author: *Cisco-Talos* File: *ida_service.py* GNU General Public License v2.0 5 vo

```
def bindiff_export():
    """
    Run the IDA Pro autoanalysis on the input file and export a BinExport database
    :param input: The input file
    :return: Status code 200 and a JSON object containing the output database
        name in key 'output', or status code 422 on invalid parameters, 408 on
        timeout or 500 on other errors.
    """
    logger.info("bindiff_export called")

    directory = None
    try:
        directory = tempfile.mkdtemp()
        if len(request.files) != 1:
            return make_response( jsonify(error = "Missing file parameter"), 422)

        filename, file_ = request.files.items()[0]
        input_ = os.path.join(directory, sanitize_filename(filename))
        file_.save(input_)

        output = os.path.join(directory, "output.BinExport")

        timeout = request.form.get('timeout', None)
        is_64_bit = request.form.get('is_64_bit', True)
        try:
            run_ida(input_, is_64_bit, timeout, os.path.join(PREFIX, "export_binex
            logger.info("Command completed successfully")
            return send_file(open(output, "rb"), as_attachment = True, attachment_
        except TimeoutError:
            return  jsonify(error = "Program execution timed out"), 408
        except OSError as err:
            return  jsonify(error = "Program execution failed with error %d" % err

    finally:
        if directory is not None:
            shutil.rmtree(directory)
```

Example 35

Project: *BASS* Author: *Cisco-Talos* File: *ida_service.py* GNU General Public License v2.0 5 vc

```
def bindiff_compare():
    logger.info("bindiff_compare called")

    input_dir = tempfile.mkdtemp()
    output_dir = tempfile.mkdtemp()
    try:
        primary = os.path.join(input_dir, "primary")
        secondary = os.path.join(input_dir, "secondary")
        try:
            request.files["primary"].save(primary)
            request.files["secondary"].save(secondary)
        except KeyError:
            return make_response(jsonify(error="Missing parameter 'primary' or 's

        timeout = request.form.get('timeout', None)

        cmd = (BINDIFF_DIFFER, "--primary", primary, "--secondary", secondary, "--
        logger.info("Executing %s", " ".join("'%s'" % x for x in cmd))
        check_call(cmd, cwd = output_dir, timeout = timeout)
        db_path = [os.path.join(output_dir, x) for x in os.listdir(output_dir)]
        if len(db_path) != 1:
            return make_response(jsonify(error = "BinDiff generated 0 or several
        return send_file(open(db_path[0], "rb"), as_attachment = True, attachment_
    except OSError as err:
        if err.errno == -9:
            return make_response(jsonify(error = "Program execution timed out"),
        else:
            return make_response(jsonify(error = "Program execution failed with e
    finally:
        shutil.rmtree(input_dir)
        shutil.rmtree(output_dir)
```

Example 36

Project: *sanctuary* Author: *bzamecnik* File: *backend.py* MIT License 5 vc

```
def experiments():
    return jsonify({'experiments': sacred_mongo.list_experiments()})
```

Example 37

Project: *sanctuary* Author: *bzamecnik* File: *backend.py* MIT License 5 vc

```
def list_runs():
    return jsonify({'runs': [str(r['_id']) for r in sacred_mongo.list_runs()]})
```

Example 38

Project: *sanctuary* Author: *bzamecnik* File: *backend.py* MIT License 5 vc

```
def list_runs_by_experiment(experiment_id):
    return jsonify({'runs': [str(r['_id']) for r in sacred_mongo.list_runs_by_exp
```

Example 39

Project: *sanctuary* Author: *bzamecnik* File: *backend.py* MIT License 5 vo

```
def run_details(run_id):
    return jsonify(json.loads(bson_dumps(sacred_mongo.get_run(run_id))))
```

Example 40

Project: *sanctuary* Author: *bzamecnik* File: *backend.py* MIT License 5 vo

```
def list_files():
    return jsonify(json.loads(bson_dumps({'files': [f for f in sacred_mongo.list_
```

Example 41

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *main.py* Apache License 2.0 5 vo

```
def addNewUser():
    username = request.form["username"]
    email = request.form["email"]
    password = request.form["password"]

    info = {"userid":1,
            "name":username,
            "email":email,
            "password":password
            }
    return jsonify(status=addUser(info))
```

Example 42

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *main.py* Apache License 2.0 5 vo

```
def login():
    if 'loggedin' in session:
        return jsonify({"status":True})

    name = str(request.form["username"])
    password = str(request.form["password"])
    status=checkLogin(name,password)

    if status==True:
        session["loggedin"]=True
    return jsonify(status=status)
```

Example 43

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *main.py* Apache License 2.0 5 vo

```
def checkUser():
    _name = str(request.form["name"])
    return jsonify(present=checkUserPresence(_name))
```

Example 44

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *json.py* Apache License 2.0 5 vo

```
def jsonify(*args, **kwargs):
    """Creates a :class:`~flask.Response` with the JSON representation of
    the given arguments with an `application/json` mimetype.  The arguments
    to this function are the same as to the :class:`dict` constructor.

    Example usage::

        from flask import jsonify

        @app.route('/_get_current_user')
        def get_current_user():
            return jsonify(username=g.user.username,
                           email=g.user.email,
                           id=g.user.id)

    This will send a JSON response like this to the browser::

        {
            "username": "admin",
            "email": "admin@localhost",
            "id": 42
        }

    For security reasons only objects are supported toplevel.  For more
    information about this, have a look at :ref:`json-security`.

    This function's response will be pretty printed if it was not requested
    with ``X-Requested-With: XMLHttpRequest`` to simplify debugging unless
    the ``JSONIFY_PRETTYPRINT_REGULAR`` config parameter is set to false.

    .. versionadded:: 0.2
    """
    indent = None
    if current_app.config['JSONIFY_PRETTYPRINT_REGULAR'] \
        and not request.is_xhr:
        indent = 2
    return current_app.response_class(dumps(dict(*args, **kwargs),
        indent=indent),
        mimetype='application/json')
```

Example 45

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *helpers.py* Apache License 2.0 5 vo

```
def test_json_bad_requests(self):
        app = flask.Flask(__name__)
        @app.route('/json', methods=['POST'])
        def return_json():
            return flask.jsonify(foo=text_type(flask.request.get_json()))
        c = app.test_client()
        rv = c.post('/json', data='malformed', content_type='application/json')
        self.assert_equal(rv.status_code, 400)
```

Example 46

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *helpers.py* Apache License 2.0 5 vo

```
def test_json_key_sorting(self):
        app = flask.Flask(__name__)
        app.testing = True
        self.assert_equal(app.config['JSON_SORT_KEYS'], True)
        d = dict.fromkeys(range(20), 'foo')

        @app.route('/')
        def index():
            return flask.jsonify(values=d)

        c = app.test_client()
        rv = c.get('/')
        lines = [x.strip() for x in rv.data.strip().decode('utf-8').splitlines()]
        self.assert_equal(lines, [
            '{',
            '"values": {',
            '"0": "foo",',
            '"1": "foo",',
            '"2": "foo",',
            '"3": "foo",',
            '"4": "foo",',
            '"5": "foo",',
            '"6": "foo",',
            '"7": "foo",',
            '"8": "foo",',
            '"9": "foo",',
            '"10": "foo",',
            '"11": "foo",',
            '"12": "foo",',
            '"13": "foo",',
            '"14": "foo",',
            '"15": "foo",',
            '"16": "foo",',
            '"17": "foo",',
            '"18": "foo",',
            '"19": "foo"',
            '}',
            '}'
        ])
```

Example 47

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *json.py* Apache License 2.0 5 vo

```
def jsonify(*args, **kwargs):
    """Creates a :class:`~flask.Response` with the JSON representation of
    the given arguments with an `application/json` mimetype.  The arguments
    to this function are the same as to the :class:`dict` constructor.

    Example usage::

        from flask import jsonify

        @app.route('/_get_current_user')
        def get_current_user():
            return jsonify(username=g.user.username,
                           email=g.user.email,
                           id=g.user.id)

    This will send a JSON response like this to the browser::

        {
            "username": "admin",
            "email": "admin@localhost",
            "id": 42
        }
    For security reasons only objects are supported toplevel.  For more
    information about this, have a look at :ref:`json-security`.

    This function's response will be pretty printed if it was not requested
    with ``X-Requested-With: XMLHttpRequest`` to simplify debugging unless
    the ``JSONIFY_PRETTYPRINT_REGULAR`` config parameter is set to false.

    .. versionadded:: 0.2
    """
    indent = None
    if current_app.config['JSONIFY_PRETTYPRINT_REGULAR'] \
        and not request.is_xhr:
        indent = 2
    return current_app.response_class(dumps(dict(*args, **kwargs),
        indent=indent),
        mimetype='application/json')
```

Example 48

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *helpers.py* Apache License 2.0 5 vo

```
def test_json_bad_requests(self):
        app = flask.Flask(__name__)
        @app.route('/json', methods=['POST'])
        def return_json():
            return flask.jsonify(foo=text_type(flask.request.get_json()))
        c = app.test_client()
        rv = c.post('/json', data='malformed', content_type='application/json')
        self.assert_equal(rv.status_code, 400)
```

Example 49

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *helpers.py* Apache License 2.0 5 vo

```
def test_jsonify(self):
        d = dict(a=23, b=42, c=[1, 2, 3])
        app = flask.Flask(__name__)
        @app.route('/kw')
        def return_kwargs():
            return flask.jsonify(**d)
        @app.route('/dict')
        def return_dict():
            return flask.jsonify(d)
        c = app.test_client()
        for url in '/kw', '/dict':
            rv = c.get(url)
            self.assert_equal(rv.mimetype, 'application/json')
            self.assert_equal(flask.json.loads(rv.data), d)
```

Example 50

Project: *gpu-mux* Author: *google* File: *gpumux.py* Apache License 2.0 5 vo

```
def status():
    return flask.jsonify(job_thread=JOB_THREAD.is_alive(),
                         completed_jobs=[x.json for x in JOBS.completed][::-1],
                         running_jobs=[x.json for x in JOBS.running],
                         pending_jobs='\n'.join(JOBS.pending))
```