Python flask.request.authorization() Examples

The following are code examples for showing how to use *flask.request.authorization()*. 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: *flask_boilerplate* Author: *guptakvgaurav* File: *auth.py* MIT License 7 vo

```
def authenticator(strategy):
    strategy_fn = None

    def basic_authenticator(f):
        @wraps(f)
        def authenticate(*args, **kwargs):
            app.logger.info('In wrapped function')
            username = request.authorization['username']
            password = request.authorization['password']
            is_valid = True if username == password else False
            if not is_valid:
                app.logger.error('[Authentication] [User-{}] tried to access '
                                 '[path-{}] with [password-{}]'
                                 .format(username, request.path, password))
                return jsonify({
                    'message': 'Username and password must be same.'
                })
            return f(*args, **kwargs)
        return authenticate

    if strategy.lower() == 'basic':
        strategy_fn = basic_authenticator

    return strategy_fn
```

Example 2

Project: *ambassador-auth-httpbasic* Author: *datawire* File: *app.py* Apache License 2.0 7 vo

```
def requires_auth(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        # Favicon is the little icon associated with a domain in a web browser. Br
        # resource /favicon.ico alongside any other HTTP request which pollutes th
        # because usually the favicon cannot be resolved. This tells the browser t
        if request.path == "/favicon.ico":
            return Response(status=403)

        auth = request.authorization
        if not auth or not check_auth(auth.username, auth.password):
            return unauthorized()

        return f(*args, **kwargs)

    return decorated
```

Example 3

6 vo

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

```
def check_authentication_response() -> Union[Response, None]:
    """
    Return the response as per the authentication requirements.
    """
    if get_authentication():
        if get_token():
            token = check_token(request, get_session())
            if not token:
                if request.authorization is None:
                    return failed_authentication(False)
                else:
                    return verify_user()
        elif request.authorization is None:
            return failed_authentication(False)
        else:
            return verify_user()
    else:
        return None
```

Example 4

Project: *everyclass-server* Author: *everyclass* File: *views.py* Mozilla Public License 2.0 6 vo

```
def exit_maintenance():
    config = get_config()
    auth = request.authorization
    if auth \
            and auth.username in config.MAINTENANCE_CREDENTIALS \
            and config.MAINTENANCE_CREDENTIALS[auth.username] == auth.password:
        try:
            os.remove(config.MAINTENANCE_FILE)  # remove maintenance file
        except OSError:
            return 'Not in maintenance mode. Ignore command.'
        open(os.path.join(os.getcwd(), 'reload'), "w+").close()  # uwsgi reload
        return 'success'
    else:
        return Response(
                'Could not verify your access level for that URL.\n'
                'You have to login with proper credentials', 401,
                {'WWW-Authenticate': 'Basic realm="Login Required"'})
```

Example 5

Project: *flasky* Author: *RoseOu* File: *flask_httpauth.py* MIT License 6 vo

```
def login_required(self, f):
        @wraps(f)
        def decorated(*args, **kwargs):
            auth = request.authorization
            # We need to ignore authentication headers for OPTIONS to avoid
            # unwanted interactions with CORS.
            # Chrome and Firefox issue a preflight OPTIONS request to check
            # Access-Control-* headers, and will fail if it returns 401.
            if request.method != 'OPTIONS':
                if auth:
                    password = self.get_password_callback(auth.username)
                else:
                    password = None
                if not self.authenticate(auth, password):
                    return self.auth_error_callback()
            return f(*args, **kwargs)
        return decorated
```

Example 6

Project: *extrapypi* Author: *karec* File: *login.py* MIT License 6 vo

```
def load_user_from_request(request):
    """Used to identify a request from pip or twine
    when downloading / uploading packages and releases
    """
    if request.authorization is None:
        return None
    username = request.authorization.get('username')
    password = request.authorization.get('password')

    user = User.query.filter_by(username=username).first()
    if not user or not custom_app_context.verify(password, user.password_hash):
        return None

    identity_changed.send(
        app._get_current_object(),
        identity=Identity(user.id)
    )

    return user
```

Example 7

Project: *python-aws-ecr-deployer* Author: *filc* File: *controllers.py* Apache License 2.0 6 vo

```
def _template_rendering(template):
    def decorator(fn):
        @wraps(fn)
        def inner_fn(*args, **kwargs):
            data = fn(*args, **kwargs)

            auth = request.authorization
            basic_auth = '' if not auth else base64.b64encode(bytes(':'.join([auth

            data.update({
                'basic_auth': basic_auth,
                'base_url': g.cn.g_('app_config').get('base_url'),
                'ecs_clusters': g.cn.f_('aws.get_ecs_clusters', region=g.cn.g_('ap
                'selected_ecs_cluster': g.cn.g_('session').get('selected_ecs_clust
            })

            return render_template(template, **data)
        return inner_fn
    return decorator
```

Example 8

Project: *heroku-python-boilerplate* Author: *chavli* File: *decorators.py* GNU General Public License v3.0 6 vo

```
def require_token(func):
    """ verifies the uuid/token combo of the given account. account type can be:
        customer, fox, merchant """
    @wraps(func)
    def decorator(*args, **kwargs):
        if request.authorization:
            uuid = request.authorization.username
            token = request.authorization.password
            try:
                manager = SessionManager()
                valid = manager.verify(uuid, token)
                if not valid:
                    return UnauthorizedResponseJson().make_response()
            except Exception as e:
                traceback.print_exc()
                return ExceptionResponseJson("unable to validate credentials", e).
        else:
            return UnauthorizedResponseJson().make_response()
        return func(*args, **kwargs)
    return decorator
```

Example 9

Project: *heroku-python-boilerplate* Author: *chavli* File: *decorators.py* GNU General Public License v3.0 6 vo

```
def require_password(func):
    """ verifies the given username/password combo """
    @wraps(func)
    def decorator(*args, **kwargs):
        if request.authorization:
            username = request.authorization.username
            password = request.authorization.password
            try:
                manager = AccountManager()
                valid = manager.verify_account(username, password)
                if not valid:
                    return UnauthorizedResponseJson().make_response()
            except Exception as e:
                traceback.print_exc()
                return ExceptionResponseJson("unable to validate credentials", e).
        else:
            return UnauthorizedResponseJson().make_response()
        return func(*args, **kwargs)
    return decorator
```

Example 10

Project: *bbotte.github.io* Author: *bbotte* File: *auth_ldap3.py* Apache License 2.0 6 vo

```
def ldap_protected(f):
  @wraps(f)
  def decorated(*args, **kwargs):
    # this can be configured and taken from elsewhere
    # path, method, groups_allowed (users in Allowed Users group will be allowed t
    authorization_config = {
      "/": {
         "GET": ["Allowed Users"]
      }
    }

    auth_endpoint_rule = authorization_config.get(request.url_rule.rule)
    if auth_endpoint_rule is not None:
      groups_allowed = auth_endpoint_rule.get(request.method) or True
    else:
      groups_allowed = True

    auth = request.authorization
    if not ('username' in session):
      if not auth or not ldap_authenticate(request,auth.username, auth.password, g
        return auth_401()
    else:
      auth_logger.debug("%s calling %s endpoint"%(session['username'],f.__name__))
    return f(*args, **kwargs)
  return decorated
```

Example 11

Project: *big-query-log-drain* Author: *jlgoldman* File: *app.py* MIT License 6 vo

```
def log():
    diagnostics.request_count += 1
    auth = request.authorization
    if not auth or auth.username != settings.LOG_DRAIN_USERNAME or auth.password !
        diagnostics.unauthorized_count += 1
        return '', 403
    diagnostics.authorized_count += 1

    log_records = []
    for log_line in _parse_log_lines(request.data):
        if log_line.startswith(settings.LOG_RECORD_PREFIX):
            json_string = log_line.replace(settings.LOG_RECORD_PREFIX, '', 1).stri
            log_record = json.loads(json_string)
            log_records.append(log_record)
            diagnostics.log_lines_processed += 1

    logplex_frame_id = request.headers.get('Logplex-Frame-Id')

    if log_records:
        _post_to_bigquery(log_records, logplex_frame_id)

    return ''
```

Example 12

Project: *flump* Author: *rolepoint* File: *sqlalchemy-auth.py* MIT License 6 vo

```
def check_auth(*args, **kwargs):
    # Check they have included auth
    auth = request.authorization
    if not auth:
        # Make use of flump error handling, this will return a nicely formatted
        # 401 response
        raise Unauthorized

    # Get the user with the passed in email address
    user = User.query.filter_by(email=auth.username).first()

    # If no User exists, or the password is incorrect, raise Unauthorized.
    if not (user and user.verify_password(auth.password)):
        raise Unauthorized


# Register the FlumpBlueprint on our app.
```

Example 13

Project: *karp-docker* Author: *spraakbanken* File: *wsauth.py* MIT License 6 vo

```
def resources():
    lexlist = {}
    lexiconfig = json.load(open('lexiconf.json'))
    request.get_data()
    data = request.form
    if data and data is not None:
        user = data.get('username', '')
    elif request.authorization is not None:
        user = request.authorization.username
    else:
        user = 'dummyuser'
    for name, val in lexiconfig.items():
        lexlist[name] = {"read": True, "write": True, "admin": True}
    return jsonify({"permitted_resources": {"lexica": lexlist},
                    "username": user,
                    "authenticated": True})
```

Example 14

Project: *scylla* Author: *acaceres2176* File: *scylla.py* Apache License 2.0 6 vo

```
def requires_auth(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        auth = request.authorization
        if not auth or not check_auth(auth.username, auth.password):
            return authenticate()
        return f(*args, **kwargs)
    return decorated

#@app.route('/regex', methods = ["POST"])
#def submit_name_regex():
#
    #request.getdata()
#    request_json = json.loads(request.data.decode('utf-8'))
#    regex = request_json["regex"]
#    print(request_json)
#    subprocess.Popen("/usr/bin/nohup find /home/ubuntu/all_unzipped -type f -exec
#    return "h'ok"

#@app.route("/status")
```

Example 15

Project: *CHN-Server* Author: *CommunityHoneyNetwork* File: *decorators.py* GNU Lesser General Public License v2.1 6 vo

```
def sensor_auth(view):
    """
    Authenticates the view, allowing access if user
    is authenticated or if requesting from a sensor.
    """
    @wraps(view)
    def wrapped_view(*args, **kwargs):
        if current_user and current_user.is_authenticated:
            return view(*args, **kwargs)
        elif request.authorization:
            auth = request.authorization
            if auth and auth.get('username') == auth.get('password') and\
               Sensor.query.filter_by(uuid=auth.get('username')).count() == 1:
                return view(*args, **kwargs)
        return error_response(errors.API_NOT_AUTHORIZED, 401)
    return wrapped_view
```

Example 16

Project: *roger-api* Author: *rogertalk* File: *auth.py* MIT License 6 vo

```
def get_session():
    try:
        access_token = request.args.get('access_token')
        if not access_token:
            authorization = request.headers['Authorization']
            token_type, access_token = authorization.split(' ')
            assert token_type == 'Bearer'
        session = Session.from_access_token(access_token)
    except:
        return None
    # Allow admins to override the session account id.
    # TODO: This needs to be checked on the token, so that a token for an
    #       admin granted to a third-party app can't also do this.
    on_behalf_of = request.args.get('on_behalf_of')
    if on_behalf_of:
        if session.account.admin:
            session = Session(int(on_behalf_of))
        else:
            raise errors.ForbiddenAction('Forbidden use of on_behalf_of')
    return session
```

Example 17

Project: *pryv* Author: *leopoldhoudin* File: *auth.py* MIT License 6 vo

```
def requires_auth(func):
    """
    Decorates the given function as requiring the inbound request to be authentica
    Basic Auth to authenticate the user.

    :param func: The function to decorate.
    :type func: ``function``

    :return: A decorated function.
    :rtype: ``function``
    """

    @wraps(func)
    def decorated(*args, **kwargs):
        """
        Verifies that the authorization headers of the request.
        """
        auth = request.authorization
        if not auth or not authenticate(auth.username, auth.password):
            return make_response('', 401, {'WWW-Authenticate': 'Basic realm="Logir
        return func(*args, **kwargs)

    return decorated
```

Example 18

Project: *GitDigger* Author: *lc-soft* File: *auth.py* GNU Affero General Public License v3.0 6 vo

```
def auth_required(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        auth = request.authorization
        if not auth:
            return abort(401)
        login = auth.username
        if login[1:-1].find('@') >= 0:
            user = User.query.filter_by(email=login).first()
            login_type = 'email'
        else:
            user = User.query.filter_by(username=login).first()
            login_type = 'username'
        if user is None:
            return abort(401, message='Unknown %s' % login_type)
        if not check_password_hash(user.password, auth.password):
            return abort(401, message='Invalid password')
        return f(*args, **kwargs)
    return decorated
```

Example 19

Project: *Simple-Freeradius-Admin* Author: *Karlheinzniebuhr* File: *routes.py* Mozilla Public License 2.0 6 vo

```
def api_auth():
    auth = request.authorization
    try:
        if not auth or not auth.username or not auth.password:
            return make_response('Could not verify', 401, {'WWW-Authenticate':'Bas

        user = db.session.query(User).filter_by(name=auth.username).first()

        if not user:
            return make_response('Could not verify', 401, {'WWW-Authenticate':'Bas

        if check_password_hash(user.password, auth.password):
                token = jwt.encode({'public_id' : user.public_id, 'exp' : datetime

                return jsonify({'token' : token.decode('UTF-8')})

        return make_response('Could not verify', 401, {'WWW-Authenticate' : 'Basic

    except Exception as e:
        err_message = "Api encountered an error: " + str(e)
        print(err_message)
        return make_response('Could not verify', 401, {'WWW-Authenticate' : 'Basic
```

Example 20

Project: *everyclass-server* Author: *everyclass* File: *views.py* Mozilla Public License 2.0 5 vo

```
def enter_maintenance():
    config = get_config()
    auth = request.authorization
    if auth \
            and auth.username in config.MAINTENANCE_CREDENTIALS \
            and config.MAINTENANCE_CREDENTIALS[auth.username] == auth.password:
        open(config.MAINTENANCE_FILE, "w+").close()  # maintenance file
        open(os.path.join(os.getcwd(), 'reload'), "w+").close()  # uwsgi reload
        return 'success'
    else:
        return Response(
                'Could not verify your access level for that URL.\n'
                'You have to login with proper credentials', 401,
                {'WWW-Authenticate': 'Basic realm="Login Required"'})
```

Example 21

Project: *eve-auth-jwt*   Author: *rs*   File: *auth.py*   MIT License   5 vo

```
def authorized(self, allowed_roles, resource, method):
        authorized = False

        if request.authorization:
            auth = request.authorization
            authorized = self.check_auth(auth.username, auth.password,
                                         allowed_roles, resource, method)
        else:
            try:
                access_token = request.args['access_token']
            except KeyError:
                access_token = request.headers.get('Authorization', '').partition(
            authorized = self.check_token(access_token, allowed_roles, resource, m

        return authorized
```

Example 22

Project: *bitmm*   Author: *thmp*   File: *web.py*   MIT License   5 vo

```
def requires_auth(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        auth = request.authorization
        if not auth or not check_auth(auth.username, auth.password):
            return authenticate()
        return f(*args, **kwargs)
    return decorated
```

Example 23

Project: *flasky*   Author: *RoseOu*   File: *flask_httpauth.py*   MIT License   5 vo

```
def username(self):
        if not request.authorization:
            return ""
        return request.authorization.username
```

Example 24

Project: *PythonMicroservicesDevelopment_Code*   Author: *mtianyan*   File: *flask_globals.py*   Apache License 2.0   5 vo

```
def authenticate():
        if request.authorization:
                g.user = request.authorization['username']
        else:
                g.user = 'Anonymous'
```

Example 25

Project: *PythonMicroservicesDevelopment_Code* Author: *mtianyan* File: *flask_auth.py* Apache License 2.0 5 vo

```
def auth():
        print("The raw Authorization header")
        print(request.environ["HTTP_AUTHORIZATION"])
        print("Flask's Authorization header")
        print(request.authorization)
        return ""
```

Example 26

Project: *greenhouse-interview-analytics* Author: *NoRedInk* File: *decorators.py* BSD 3-Clause "New" or "Revised" License 5 vo

```
def requires_auth(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        auth = request.authorization
        if not auth or not check_auth(auth.username, auth.password):
            return authenticate()
        return f(*args, **kwargs)
    return decorated
```

Example 27

Project: *pongr* Author: *wseaton* File: *admin.py* MIT License 5 vo

```
def is_accessible(self):
        auth = request.authorization or request.environ.get('REMOTE_USER')  # wor
        if not auth or (auth.username, auth.password) != ('admin', 'password123'):
            raise HTTPException('', Response(
                "Please log in.", 401,
                {'WWW-Authenticate': 'Basic realm="Login Required"'}
            ))
        return True
```

Example 28

Project: *gransk* Author: *pcbje* File: *ui.py* Apache License 2.0 5 vo

```
def requires_auth(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        auth = request.authorization
        if not auth or not check_auth(auth.username, auth.password):
          if _globals.get('test'):
            return f(*args, **kwargs)
          return authenticate()
        return f(*args, **kwargs)
    return decorated
```

Example 29

Project: *social-relay* Author: *jaywink* File: *auth.py* GNU Affero General Public License v3.0 5 vo

```
def basic_auth():
    """Ensure basic authorization."""
    auth = request.authorization
    if not auth or not check_auth(auth.username, auth.password):
        return authenticate()
```

Example 30

Project: *slot* Author: *nhshd-slot* File: *basic_auth.py* MIT License 5 vo

```
def requires_auth(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        auth = request.authorization
        if not auth or not check_auth(auth.username, auth.password):
            return authenticate()
        return f(*args, **kwargs)
    return decorated
```

Example 31

Project: *weblablib* Author: *weblabdeusto* File: *views.py* GNU Affero General Public License v3.0 5 vo

```
def _require_http_credentials():
    """
    All methods coming from WebLab-Deusto must be authenticated (except for /api).
    WEBLAB_USERNAME and WEBLAB_PASSWORD configuration variables, which are used by
    Take into account that this username and password authenticate the WebLab-Deus
    For example, a WebLab-Deusto in institution A might have 'institutionA' as WEB
    randomly generated password as WEBLAB_PASSWORD.
    """
    # Don't require credentials in /api
    if request.url.endswith('/api'):
        return None

    auth = request.authorization
    if auth:
        provided_username = auth.username
        provided_password = auth.password
    else:
        provided_username = provided_password = None

    expected_username = current_app.config[ConfigurationKeys.WEBLAB_USERNAME]
    expected_password = current_app.config[ConfigurationKeys.WEBLAB_PASSWORD]
    if provided_username != expected_username or provided_password != expected_pas
        if request.url.endswith('/test'):
            error_message = "Invalid credentials: no username provided"
            if provided_username:
                error_message = "Invalid credentials: wrong username provided. Che
            return Response(json.dumps(dict(valid=False, error_messages=[error_mes

        if expected_username:
            current_app.logger.warning("Invalid credentials provided to access {}.

        return Response(response=("You don't seem to be a WebLab-Instance"), statu

    return None
```

Example 32

5 vo

Project: *minicps* Author: *scy-phy* File: *run.py* MIT License

```
def requires_auth(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        auth = request.authorization
        if not auth:
            return authenticate()

        elif not check_auth(auth.username, auth.password):
            return authenticate()
        return f(*args, **kwargs)

    return decorated
```

Example 33

Project: *calibre-web* Author: *janeczku* File: *opds.py* GNU General Public License v3.0 5 vo

```
def requires_basic_auth_if_no_ano(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        auth = request.authorization
        if config.config_anonbrowse != 1:
            if not auth or auth.type != 'basic' or not check_auth(auth.username, a
                return authenticate()
        return f(*args, **kwargs)
    if config.config_login_type == constants.LOGIN_LDAP and services.ldap:
        return services.ldap.basic_auth_required(f)
    return decorated
```

Example 34

Project: *karp-backend* Author: *spraakbanken* File: *helpers.py* MIT License 5 vo

```
def get_user():
    auth = request.authorization
    if auth is None:
        return "UnkownUser"
    return auth.username
```

Example 35

Project: *jbox* Author: *jpush* File: *flask_httpauth.py* MIT License 5 vo

```
def login_required(self, f):
        @wraps(f)
        def decorated(*args, **kwargs):
            auth = request.authorization
            if auth is None and 'Authorization' in request.headers:
                # Flask/Werkzeug do not recognize any authentication types
                # other than Basic or Digest, so here we parse the header by
                # hand
                try:
                    auth_type, token = request.headers['Authorization'].split(
                        None, 1)
                    auth = Authorization(auth_type, {'token': token})
                except ValueError:
                    # The Authorization header is either empty or has no token
                    pass
            # if the auth type does not match, we act as if there is no auth
            # this is better than failing directly, as it allows the callback
            # to handle special cases, like supporting multiple auth types
            if auth is not None and auth.type.lower() != self.scheme.lower():
                auth = None

            # Flask normally handles OPTIONS requests on its own, but in the
            # case it is configured to forward those to the application, we
            # need to ignore authentication headers and let the request through
            # to avoid unwanted interactions with CORS.
            if request.method != 'OPTIONS':  # pragma: no cover
                if auth and auth.username:
                    password = self.get_password_callback(auth.username)
                else:
                    password = None
                if not self.authenticate(auth, password):
                    return self.auth_error_callback()
            return f(*args, **kwargs)
        return decorated
```

Example 36

Project: *jbox* Author: *jpush* File: *flask_httpauth.py* MIT License 5 vo

```
def username(self):
        if not request.authorization:
            return ""
        return request.authorization.username
```

Example 37

Project: *python-flask-restful-api* Author: *akashtalole* File: *auth.py* MIT License 5 vo

```
def requires_basic_auth(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        auth = request.authorization
        if not auth or not AuthManager.check_auth_admin(auth.username, auth.passwo
            return make_response('Could not verify your access level for that URL.
                                 'You have to login with proper credentials', 401,
                                 {'WWW-Authenticate': 'Basic realm="Login Requirec
        return f(*args, **kwargs)
    return decorated
```

Example 38

Project: *mlsb-platform* Author: *fras2560* File: *authentication.py* Apache License 2.0 5 vo

```
def requires_admin(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        auth = request.authorization
        if not auth or not check_auth(auth.username, auth.password):
            return authenticate()
        elif 'admin' in session and 'password' in session:
            # check if user signed in already
            logged = check_auth(session['admin'], session['password'])
            if not logged:
                return authenticate()
        return f(*args, **kwargs)
    return decorated
```

Example 39

Project: *stackstorm-ghost2logger* Author: *StackStorm-Exchange* File: *ghost2loggersensor.py* Apache License 2.0 5 vo

```
def requires_auth(f):
    """Wrapper function."""
    @wraps(f)
    def decorated(*args, **kwargs):
        auth = request.authorization
        if not auth or not check_auth(auth.username, auth.password):
            return authenticate()
        return f(*args, **kwargs)
    return decorated
```

Example 40

Project: *pyhackeriet* Author: *hackeriet* File: *__init__.py* Apache License 2.0 5 vo

```
def requires_auth(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        auth = request.authorization
        if not auth or not check_auth(auth.username, auth.password):
            return authenticate()
        return f(*args, **kwargs)
    return decorated
```

Example 41

Project: *pyhackeriet* Author: *hackeriet* File: *__init__.py* Apache License 2.0 5 vo

```
def requires_admin(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        auth = request.authorization
        if not auth or not check_admin(auth.username, auth.password):
            return authenticate()
        return f(*args, **kwargs)
    return decorated
```

Example 42

Project: *pyhackeriet* Author: *hackeriet* File: *__init__.py* Apache License 2.0 5 vo

```
def account():
    user=request.authorization.username
    return render_template('account.html', username=user,
                           history=brusdb.transaction_history(user),
                           balance=brusdb.balance(user),
                           key=stripe_keys['publishable_key'])
```

Example 43

Project: *pyhackeriet* Author: *hackeriet* File: *__init__.py* Apache License 2.0 5 vo

```
def manual_subtract():
    user=request.authorization.username
    if brusdb.subtract_funds(user, int(request.form['value']),
                            request.form['desc'], True):
        return redirect(url_for('account'))
    else:
        return "Insufficient funds"
```

Example 44

Project: *pyhackeriet* Author: *hackeriet* File: *__init__.py* Apache License 2.0 5 vo

```
def admin():
    user=request.authorization.username
    return render_template('admin.html', username=user,
                           users=members.list_users())
```

Example 45

Project: *tensorflow-object-detection-example* Author: *GoogleCloudPlatform* File: *decorator.py* Apache License 2.0 5 vo

```
def requires_auth(f):
  @wraps(f)
  def decorated(*args, **kwargs):
    auth = request.authorization
    if not auth or not check_auth(auth.username, auth.password):
      return authenticate()
    return f(*args, **kwargs)
  return decorated
```

Example 46

Project: *tensorflow-object-detection-example* Author: *GoogleCloudPlatform* File: *decorator.py* Apache License 2.0 5 vo

```
def requires_auth(f):
  @wraps(f)
  def decorated(*args, **kwargs):
    auth = request.authorization
    if not auth or not check_auth(auth.username, auth.password):
      return authenticate()
    return f(*args, **kwargs)
  return decorated
```

Example 47

Project: *celery-dashboard* Author: *mehdigmira* File: *auth.py* MIT License 5 vo

```
def requires_auth(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        auth = request.authorization
        conf = current_app.celery_app.conf
        if conf.dashboard_username and conf.dashboard_password:
            if not auth or not check_auth(auth.username, auth.password):
                return authenticate()
        return f(*args, **kwargs)
    return decorated
```

Example 48

Project: *vocaltrimmer* Author: *vincentpalma* File: *admin.py* MIT License 5 vo

```
def is_accessible(self):
        auth = request.authorization or request.environ.get('REMOTE_USER')  # wor
        if not auth or (auth.username, auth.password) != app.config['ADMIN_CREDENT
            raise HTTPException('', Response('You have to an administrator.', 401,
                {'WWW-Authenticate': 'Basic realm="Login Required"'}
            ))
        return True

# Users
```

Example 49

Project: *course_plus_server* Author: *luckymore0520* File: *user.py* MIT License 5 vo

```
def getCurrentUser(request):
    auth = request.authorization
    if not auth:
        return None
    token = auth.username
    return User.verify_auth_token(token)
```

Example 50

Project: *upb-son-editor-backend* Author: *CN-UPB* File: *configapi.py* Apache License 2.0 5 vo

```
def requires_auth(f):
    def decorated(*args, **kwargs):
        auth = request.authorization
        if not auth or not check_auth(auth.username, auth.password):
            return authenticate()
        return f(*args, **kwargs)

    return decorated
```