Python flask.request.accept_mimetypes() Examples

The following are code examples for showing how to use *flask.request.accept_mimetypes()*. 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: *libresign* Author: *this-is-ari* File: *accepts.py* MIT License 6 vo

```
def accepts(*args):
    mimetypes = []
    for arg in args:
        if isinstance(arg, list):
            mimetypes += arg
        else:
            mimetypes.append(arg)

    def is_acceptable():
        return not 'Accept' in request.headers or \
            any(x in request.accept_mimetypes for x in mimetypes)

    def decorator(fun):
        @wraps(fun)
        def wrapper(*args, **kwargs):
            if not is_acceptable():
                return jsonify(
                    msg="Unable to comply with Accept header, " +
                        "see mimetypes field for acceptable mimetypes",
                    mimetypes=mimetypes
                ), 406

            return fun(*args, **kwargs)
        return wrapper
    return decorator
```

Example 2

Project: *RNASEqTool* Author: *armell* File: *gene_expression_resources.py* MIT License 5 vo

```
def post(self):
        try:
            args = chart_parser.parse_args()
            selected_genes = args["gene_set"]
            selected_dataset = args["dataset_identifier"]
            selected_charttype = args["chart_type"]
            print args["chart_type"]

            request_to_handle = request.accept_mimetypes.best_match(['application

            if request_to_handle == 'application/pdf':
                accept = "pdf"
            else:
                accept = "html"

            chart_resource = nc.sample_distribution_for_genes(selected_dataset, se
                                                              title=selected_datas
                                                              chart_type=selected_
                                                              chart_rendering=acce
```

```
            # returns the write rendering based on generator [bokeh with bokehjs c
            if accept == "html":
                if chart_resource["generator"] == "bokeh":
                    return cr.BokehResource(chart_resource["chart"]), 201
                if chart_resource["generator"] == "matplot":
                    return cr.MatPlotLibResource(chart_resource["chart"]), 201

            if accept == "pdf":
                filename = ig.get_generated_guid_as_string() + ".pdf"
                file_path = APP_CONFIG["domain"] + APP_CONFIG["static_documents"]
                chart_resource["chart"].savefig("." + APP_CONFIG["static_documents
                return cr.StringApiResource(file_path + filename), 201
        except:
            return cr.StringApiResource("Chart creation failed"), 400
```

Example 3

Project: *RNASEqTool* Author: *armell* File: *gene_expression_resources.py* MIT License 5 vo

```
def get(self, datasetId, elements="file"):
        print("get dataset")
        print(datasetId)
        path = APP_CONFIG["application_files_location"] + APP_CONFIG["application_
        if elements == "file":
            print("creating frame")
            print("accepting ")
            print(request.accept_mimetypes)
            # convert public to intern...
            return cr.DataFrameApiResource(dr.get_data_frame_from_hdf(datasetId, p
        if elements == "genes":
            print("retrieving genes")
            dataset_with_ensembl = dr.get_dataset_genes_with_ensembl_info(dataset1

            return cr.JsonResource(dataset_with_ensembl), 201
        if elements == "desc":
            dataset = dr.get_data_frame_from_hdf(datasetId, path)
            filtered = dataset.sum(1) == 0
            d_filtered = dataset[-filtered]
            filtered_data_set = d_filtered  # pd.DataFrame(scale(d_filtered), inde
            json_message = {"meta":{"samples":len(filtered_data_set.columns), "ger
                "samples": [{"id": s} for idx, s in enumerate(filtered_data_set.co
                {"identifier": item[0], "mean": item[1].mean(), "sd": item[1].std(
                for item in filtered_data_set.iterrows()
                ]}
            return cr.JsonResource(json_message), 201
```

Example 4

Project: *har-sanitizer* Author: *google* File: *decorators.py* Apache License 2.0 5 vo

```
def accept(mimetype):
    def decorator(func):
        """
        Decorator which returns a 406 Not Acceptable if the client won't accept
        a certain mimetype
        """
        @wraps(func)
        def wrapper(*args, **kwargs):
            if mimetype in request.accept_mimetypes:
                return func(*args, **kwargs)
```

```
            message = "Request must accept {} data".format(mimetype)
            data = json.dumps({"message": message})
            return Response(data, 406, mimetype="application/json")
        return wrapper
    return decorator
```

Example 5

Project: *AUCR* Author: *AUCR* File: *handlers.py* GNU General Public License v3.0 5 vo

```
def wants_json_response():
    """Data type json response request."""
    return request.accept_mimetypes['application/json'] >= request.accept_mimetyp
```

Example 6

Project: *noobotkit* Author: *nazroll* File: *__init__.py* MIT License 5 vo

```
def make_response(self, data, *args, **kwargs):
        """Looks up the representation transformer for the requested media
        type, invoking the transformer to create a response object. This
        defaults to default_mediatype if no transformer is found for the
        requested mediatype. If default_mediatype is None, a 406 Not
        Acceptable response will be sent as per RFC 2616 section 14.1

        :param data: Python object containing response data to be transformed
        """
        default_mediatype = kwargs.pop('fallback_mediatype', None) or self.default
        mediatype = request.accept_mimetypes.best_match(
            self.representations,
            default=default_mediatype,
        )
        if mediatype is None:
            raise NotAcceptable()
        if mediatype in self.representations:
            resp = self.representations[mediatype](data, *args, **kwargs)
            resp.headers['Content-Type'] = mediatype
            return resp
        elif mediatype == 'text/plain':
            resp = original_flask_make_response(str(data), *args, **kwargs)
            resp.headers['Content-Type'] = 'text/plain'
            return resp
        else:
            raise InternalServerError()
```

Example 7

Project: *noobotkit* Author: *nazroll* File: *__init__.py* MIT License 5 vo

```
def mediatypes(self):
        """Returns a list of requested mediatypes sent in the Accept header"""
        return [h for h, q in sorted(request.accept_mimetypes,
                                     key=operator.itemgetter(1), reverse=True)]
```

Example 8

Project: *noobotkit* Author: *nazroll* File: *__init__.py* MIT License 5 vo

```
def dispatch_request(self, *args, **kwargs):
```

```
        # Taken from flask
        #noinspection PyUnresolvedReferences
        meth = getattr(self, request.method.lower(), None)
        if meth is None and request.method == 'HEAD':
            meth = getattr(self, 'get', None)
        assert meth is not None, 'Unimplemented method %r' % request.method

        for decorator in self.method_decorators:
            meth = decorator(meth)

        resp = meth(*args, **kwargs)

        if isinstance(resp, ResponseBase):  # There may be a better way to test
            return resp

        representations = self.representations or OrderedDict()

        #noinspection PyUnresolvedReferences
        mediatype = request.accept_mimetypes.best_match(representations, default=
        if mediatype in representations:
            data, code, headers = unpack(resp)
            resp = representations[mediatype](data, code, headers)
            resp.headers['Content-Type'] = mediatype
            return resp

        return resp
```

Example 9

Project: *libresign* Author: *this-is-ari* File: *signature.py* MIT License 5 vo

```
def request_wants_pdf() -> bool:
    best = request.accept_mimetypes \
        .best_match(['application/pdf', 'image/png'], 'application/pdf')

    return best == 'application/pdf'
```

Example 10

Project: *elogy* Author: *johanfforsberg* File: *utils.py* GNU General Public License v3.0 5 vo

```
def request_wants_json():
    "Check whether we should send a JSON reply"
    best = request.accept_mimetypes \
        .best_match(['application/json', 'text/html'])
    print(best)
    print(request.accept_mimetypes[best],
          request.accept_mimetypes['text/html'])

    return best == 'application/json' and \
        request.accept_mimetypes[best] >= \
        request.accept_mimetypes['text/html']
```

Example 11

Project: *lear* Author: *bcgov* File: *business_filings.py* Apache License 2.0 5 vo

```
def get(identifier, filing_id=None):
        """Return a JSON object with meta information about the Service."""
        business = Business.find_by_identifier(identifier)
```

```
        if not business:
            return jsonify({'message': f'{identifier} not found'}), HTTPStatus.NOT

        if filing_id:
            rv = db.session.query(Business, Filing). \
                filter(Business.id == Filing.business_id).\
                filter(Business.identifier == identifier).\
                filter(Filing.id == filing_id).\
                one_or_none()
            if not rv:
                return jsonify({'message': f'{identifier} no filings found'}), HTT

            if str(request.accept_mimetypes) == 'application/pdf':
                return legal_api.reports.get_pdf(rv[1])

            return jsonify(rv[1].json)

        # Does it make sense to get a PDF of all filings?
        if str(request.accept_mimetypes) == 'application/pdf':
            return jsonify({'message': _('Cannot return a single PDF of multiple f
                HTTPStatus.NOT_ACCEPTABLE

        rv = []
        filings = Filing.get_filings_by_status(business.id, [Filing.Status.COMPLET
        for filing in filings:
            rv.append(filing.json)

        return jsonify(filings=rv)
```

Example 12

Project: *xuemc* Author: *skycucumber* File: *__init__.py* GNU General Public License v2.0 5 vo

```
def make_response(self, data, *args, **kwargs):
        """Looks up the representation transformer for the requested media
        type, invoking the transformer to create a response object. This
        defaults to default_mediatype if no transformer is found for the
        requested mediatype. If default_mediatype is None, a 406 Not
        Acceptable response will be sent as per RFC 2616 section 14.1

        :param data: Python object containing response data to be transformed
        """
        default_mediatype = kwargs.pop('fallback_mediatype', None) or self.default
        mediatype = request.accept_mimetypes.best_match(
            self.representations,
            default=default_mediatype,
        )
        if mediatype is None:
            raise NotAcceptable()
        if mediatype in self.representations:
            resp = self.representations[mediatype](data, *args, **kwargs)
            resp.headers['Content-Type'] = mediatype
            return resp
        elif mediatype == 'text/plain':
            resp = original_flask_make_response(str(data), *args, **kwargs)
            resp.headers['Content-Type'] = 'text/plain'
            return resp
        else:
            raise InternalServerError()
```

Example 13

Project: *xuemc* Author: *skycucumber* File: *__init__.py* GNU General Public License v2.0 5 vo

```
def mediatypes(self):
        """Returns a list of requested mediatypes sent in the Accept header"""
        return [h for h, q in sorted(request.accept_mimetypes,
                                     key=operator.itemgetter(1), reverse=True)]
```

Example 14

Project: *xuemc* Author: *skycucumber* File: *__init__.py* GNU General Public License v2.0 5 vo

```
def dispatch_request(self, *args, **kwargs):

        # Taken from flask
        #noinspection PyUnresolvedReferences
        meth = getattr(self, request.method.lower(), None)
        if meth is None and request.method == 'HEAD':
            meth = getattr(self, 'get', None)
        assert meth is not None, 'Unimplemented method %r' % request.method

        for decorator in self.method_decorators:
            meth = decorator(meth)

        resp = meth(*args, **kwargs)

        if isinstance(resp, ResponseBase):  # There may be a better way to test
            return resp

        representations = self.representations or {}

        #noinspection PyUnresolvedReferences
        mediatype = request.accept_mimetypes.best_match(representations, default=
        if mediatype in representations:
            data, code, headers = unpack(resp)
            resp = representations[mediatype](data, code, headers)
            resp.headers['Content-Type'] = mediatype
            return resp

        return resp
```

Example 15

Project: *url_shortener* Author: *martydill* File: *__init__.py* MIT License 5 vo

```
def make_response(self, data, *args, **kwargs):
        """Looks up the representation transformer for the requested media
        type, invoking the transformer to create a response object. This
        defaults to default_mediatype if no transformer is found for the
        requested mediatype. If default_mediatype is None, a 406 Not
        Acceptable response will be sent as per RFC 2616 section 14.1

        :param data: Python object containing response data to be transformed
        """
        default_mediatype = kwargs.pop('fallback_mediatype', None) or self.default
        mediatype = request.accept_mimetypes.best_match(
            self.representations,
            default=default_mediatype,
        )
        if mediatype is None:
            raise NotAcceptable()
        if mediatype in self.representations:
```

```
            resp = self.representations[mediatype](data, *args, **kwargs)
            resp.headers['Content-Type'] = mediatype
            return resp
        elif mediatype == 'text/plain':
            resp = original_flask_make_response(str(data), *args, **kwargs)
            resp.headers['Content-Type'] = 'text/plain'
            return resp
        else:
            raise InternalServerError()
```

Example 16

Project: *url_shortener* Author: *martydill* File: *__init__.py* MIT License 5 vo

```
def mediatypes(self):
        """Returns a list of requested mediatypes sent in the Accept header"""
        return [h for h, q in sorted(request.accept_mimetypes,
                                     key=operator.itemgetter(1), reverse=True)]
```

Example 17

Project: *url_shortener* Author: *martydill* File: *__init__.py* MIT License 5 vo

```
def dispatch_request(self, *args, **kwargs):

        # Taken from flask
        #noinspection PyUnresolvedReferences
        meth = getattr(self, request.method.lower(), None)
        if meth is None and request.method == 'HEAD':
            meth = getattr(self, 'get', None)
        assert meth is not None, 'Unimplemented method %r' % request.method

        for decorator in self.method_decorators:
            meth = decorator(meth)

        resp = meth(*args, **kwargs)

        if isinstance(resp, ResponseBase):  # There may be a better way to test
            return resp

        representations = self.representations or {}

        #noinspection PyUnresolvedReferences
        mediatype = request.accept_mimetypes.best_match(representations, default=
        if mediatype in representations:
            data, code, headers = unpack(resp)
            resp = representations[mediatype](data, code, headers)
            resp.headers['Content-Type'] = mediatype
            return resp

        return resp
```

Example 18

Project: *lindat-translation* Author: *ufal* File: *views.py* BSD 2-Clause "Simplified" License 5 vo

```
def _request_wants_json():
    best = request.accept_mimetypes \
        .best_match(['application/json', 'text/html'])
    return best == 'application/json' and \
```

```
            request.accept_mimetypes[best] > \
            request.accept_mimetypes['text/html']
```

Example 19

Project: *tripoli* Author: *DDMAL* File: *index.py* MIT License 5 vo

```
def val_with_content_type(value, template):
    """Return either json or text/html with value dict."""
    mimes = request.accept_mimetypes
    json_score = mimes['application/json'] if 'application/json' in mimes else 0
    text_html_score = mimes['text/html'] if 'text/html' in mimes else 0
    if json_score > text_html_score:
        return jsonify(value)
    else:
        return render_template(template, **value)
```

Example 20

Project: *Elogy* Author: *MaxIV-KitsControls* File: *utils.py* GNU General Public License v3.0 5 vo

```
def request_wants_json():
    "Check whether we should send a JSON reply"
    best = request.accept_mimetypes \
        .best_match(['application/json', 'text/html'])
    print(best)
    print(request.accept_mimetypes[best],
          request.accept_mimetypes['text/html'])

    return best == 'application/json' and \
        request.accept_mimetypes[best] >= \
        request.accept_mimetypes['text/html']
```

Example 21

Project: *pipa-pay-server* Author: *davidvon* File: *__init__.py* Apache License 2.0 5 vo

```
def mediatypes(self):
        """Returns a list of requested mediatypes sent in the Accept header"""
        return [h for h, q in request.accept_mimetypes]
```

Example 22

Project: *eq-survey-runner* Author: *ONSdigital* File: *questionnaire.py* MIT License 5 vo

```
def request_wants_json():
    best = request.accept_mimetypes \
        .best_match(['application/json', 'text/html'])
    return best == 'application/json' and \
        request.accept_mimetypes[best] > \
        request.accept_mimetypes['text/html']
```

Example 23

Project: *scarfage* Author: *cmazuc* File: *main.py* GNU General Public License v2.0 5 vo

```
def request_wants_json():
    """
    Check the request to see if the client wants JSON instead of rendered HTML
```

```
    :return: True or False
    """
    best = request.accept_mimetypes \
        .best_match(['application/json', 'text/html'])
    return best == 'application/json' and \
        request.accept_mimetypes[best] > \
        request.accept_mimetypes['text/html']
```

Example 24

Project: *junior_project* Author: *tishq* File: *__init__.py* MIT License 5 vo

```
def make_response(self, data, *args, **kwargs):
        """Looks up the representation transformer for the requested media
        type, invoking the transformer to create a response object. This
        defaults to default_mediatype if no transformer is found for the
        requested mediatype. If default_mediatype is None, a 406 Not
        Acceptable response will be sent as per RFC 2616 section 14.1

        :param data: Python object containing response data to be transformed
        """
        default_mediatype = kwargs.pop('fallback_mediatype', None) or self.default
        mediatype = request.accept_mimetypes.best_match(
            self.representations,
            default=default_mediatype,
        )
        if mediatype is None:
            raise NotAcceptable()
        if mediatype in self.representations:
            resp = self.representations[mediatype](data, *args, **kwargs)
            resp.headers['Content-Type'] = mediatype
            return resp
        elif mediatype == 'text/plain':
            resp = original_flask_make_response(str(data), *args, **kwargs)
            resp.headers['Content-Type'] = 'text/plain'
            return resp
        else:
            raise InternalServerError()
```

Example 25

Project: *junior_project* Author: *tishq* File: *__init__.py* MIT License 5 vo

```
def mediatypes(self):
        """Returns a list of requested mediatypes sent in the Accept header"""
        return [h for h, q in sorted(request.accept_mimetypes,
                                     key=operator.itemgetter(1), reverse=True)]
```

Example 26

Project: *junior_project* Author: *tishq* File: *__init__.py* MIT License 5 vo

```
def dispatch_request(self, *args, **kwargs):

        # Taken from flask
        #noinspection PyUnresolvedReferences
        meth = getattr(self, request.method.lower(), None)
        if meth is None and request.method == 'HEAD':
            meth = getattr(self, 'get', None)
```

```
        assert meth is not None, 'Unimplemented method %r' % request.method

        if isinstance(self.method_decorators, Mapping):
            decorators = self.method_decorators.get(request.method.lower(), [])
        else:
            decorators = self.method_decorators

        for decorator in decorators:
            meth = decorator(meth)

        resp = meth(*args, **kwargs)

        if isinstance(resp, ResponseBase):  # There may be a better way to test
            return resp

        representations = self.representations or OrderedDict()

        #noinspection PyUnresolvedReferences
        mediatype = request.accept_mimetypes.best_match(representations, default=
        if mediatype in representations:
            data, code, headers = unpack(resp)
            resp = representations[mediatype](data, code, headers)
            resp.headers['Content-Type'] = mediatype
            return resp

        return resp
```

Example 27

Project: *junior_project* Author: *tishq* File: *__init__.py* MIT License 5 vo

```
def make_response(self, data, *args, **kwargs):
        """Looks up the representation transformer for the requested media
        type, invoking the transformer to create a response object. This
        defaults to default_mediatype if no transformer is found for the
        requested mediatype. If default_mediatype is None, a 406 Not
        Acceptable response will be sent as per RFC 2616 section 14.1

        :param data: Python object containing response data to be transformed
        """
        default_mediatype = kwargs.pop('fallback_mediatype', None) or self.default
        mediatype = request.accept_mimetypes.best_match(
            self.representations,
            default=default_mediatype,
        )
        if mediatype is None:
            raise NotAcceptable()
        if mediatype in self.representations:
            resp = self.representations[mediatype](data, *args, **kwargs)
            resp.headers['Content-Type'] = mediatype
            return resp
        elif mediatype == 'text/plain':
            resp = original_flask_make_response(str(data), *args, **kwargs)
            resp.headers['Content-Type'] = 'text/plain'
            return resp
        else:
            raise InternalServerError()
```

Example 28

Project: *junior_project* Author: *tishq* File: *__init__.py* MIT License 5 vo

```
def mediatypes(self):
        """Returns a list of requested mediatypes sent in the Accept header"""
        return [h for h, q in sorted(request.accept_mimetypes,
                                     key=operator.itemgetter(1), reverse=True)]
```

Example 29

Project: *bluemix-python-eve-sample* Author: *ibmjstart* File: *home.py* Apache License 2.0 5 vo

```
def request_wants_json():
    best = request.accept_mimetypes \
        .best_match(['application/json',
                     'application/json; charset=utf-8',
                     'text/html'])
    return best == 'application/json' and \
        request.accept_mimetypes[best] > \
        request.accept_mimetypes['text/html']
```

Example 30

Project: *goalkeeper* Author: *jiazhuangle* File: *handlers.py* MIT License 5 vo

```
def needs_json():
    return request.accept_mimetypes['application/json'] >= \
           request.accept_mimetypes['text/html']
```

Example 31

Project: *bluemix-promocodes* Author: *sebschrader* File: *__init__.py* MIT License 5 vo

```
def request_wants_json():
    best = request.accept_mimetypes \
        .best_match(['application/json', 'text/html'])
    return best == 'application/json' and \
        request.accept_mimetypes[best] > \
        request.accept_mimetypes['text/html']
```

Example 32

Project: *flask-raml* Author: *salsita* File: *flask_raml.py* MIT License 5 vo

```
def get_response_mimetype(self, response, accept=None, request=request):
        if accept is None:
            if request and has_request_context():
                accept = map(itemgetter(0), request.accept_mimetypes)
        return super(API, self).get_response_mimetype(response, accept)
```

Example 33

Project: *ztf-alert-server* Author: *LCOGT* File: *ztf.py* GNU General Public License v3.0 5 vo

```
def request_wants_json():
    if request.args.get('format', 'html', type=str) == 'json':
        return True
    else:
        best = request.accept_mimetypes.best_match(['application/json', 'text/htm
        return best == 'application/json' and \
            request.accept_mimetypes[best] > \
            request.accept_mimetypes['text/html']
```

Example 34

Project: *kaku* Author: *bear* File: *main.py* MIT License 5 vo

```
def request_wants_json():
    best = request.accept_mimetypes \
        .best_match(['application/json', 'text/html'])
    return best == 'application/json' and \
        request.accept_mimetypes[best] > \
        request.accept_mimetypes['text/html']
```

Example 35

Project: *do-portal* Author: *certeu* File: *__init__.py* BSD 3-Clause "New" or "Revised" License 5 vo

```
def before_api_request():
    if 'application/json' not in request.accept_mimetypes:
        return errors.not_acceptable()
```

Example 36

Project: *do-portal* Author: *certeu* File: *__init__.py* BSD 3-Clause "New" or "Revised" License 5 vo

```
def before_api_request():
    if 'application/json' not in request.accept_mimetypes:
        return errors.not_acceptable()
```

Example 37

Project: *do-portal* Author: *certeu* File: *__init__.py* BSD 3-Clause "New" or "Revised" License 5 vo

```
def before_auth_request():
    if 'application/json' not in request.accept_mimetypes:
        return errors.not_acceptable()
```

Example 38

Project: *libresign* Author: *this-is-ari* File: *audit.py* MIT License 4 vo

```
def audit_get(docId: str):
    """ Fetch the audit log for a given document.
        This endpoint can fetch the audit log in
        multiple formats depending on the mimetypes
        given by the client in the `Accept` header.

        Formats:
            * `application/pdf`
            * `application/json`

        Arguments:
            docId (str): The document ID

        Response:
            If successful, this endpoint will respond
            with HTTP status 200 and the body will
            contain the audit log in the requested
            format.

            If an error occurs then this will respond
```

```
        with a 4XX error code and a JSON body
        describing the error.
"""

session = Session()
try:
    doc_id = UUID(hex=docId)
except ValueError:
    return ErrorMessage("Invalid document ID"), 400

if not verify_permission(session, doc_id):
    return ErrorMessage("Not Authorized"), 401

doc_exists = (
    session
        .query(Document)
        .filter(Document.id == doc_id.bytes)
        .with_entities(Document.id)
        .one_or_none()
    is not None
)

if not doc_exists:
    return ErrorMessage("Not Found"), 404

# TODO: Properly do content negotiation
if 'application/json' in request.accept_mimetypes:
    return get_audit_log_json(session, doc_id)
elif 'application/pdf' in request.accept_mimetypes:
    return get_audit_log_pdf(session, doc_id)

return ErrorMessage(
    msg='Acceptable types are "application/pdf" or "application/json"'
), 406
```