# Pituophis Documentation

**dotcomboom**

**Feb 13, 2019**

# Contents:

- genindex

**class** pituophis.**Request**(*host='127.0.0.1'*, *port=70*, *path='/'*, *query=''*, *itype='9'*, *tls=False*, *tls_verify=True*, *client=''*, *pub_dir='pub/'*, *alt_handler=False*)

Bases: object

*Client/Server.* Represents a request to be sent to a Gopher server, or received from a client.

**client = None**
*Server.* The IP address of the connected client.

**get**()
*Client.* Sends the Request and returns a Response object.

**host = None**
*Client/Server.* The hostname of the server.

**path = None**
*Client/Server.* Path on the target server to request, or being requested.

**port = None**
*Client/Server.* The port of the server. For regular Gopher servers, this is most commonly 70, and for S/Gopher servers it is typically 105.

**pub_dir = None**
*Server.* The default handler uses this as which directory to serve. Default is 'pub/'.

**query = None**
*Client/Server.* Search query for the server to process. Omitted when blank.

**tls = None**
*Client/Server.* Whether the request is to be, or was sent to an S/Gopher server over TLS.

**tls_verify = None**
*Client.* Whether to verify the certificate sent from the server, rejecting self-signed and invalid certificates.

**type = None**
*Client.* Item type of the request. Purely for client-side usage, not used when sending or receiving requests.

**url**()
Returns a URL equivalent to the Request's properties.

**class** pituophis.**Response**(*stream*)

Bases: object

*Client.* Returned by Request.get() and get(). Represents a received binary object from a Gopher server.

**binary = None**
The data received from the server as a Bytes binary object.

**menu**()
Decodes the binary as text and parses it as a Gopher menu. Returns a List of Gopher menu items parsed as the Selector type.

**text**()
Returns the binary decoded as a UTF-8 String.

**class** pituophis.**Selector**(*itype='i'*, *text=''*, *path='/'*, *host='error.host'*, *port=0*, *tls=False*)

Bases: object

*Server/Client.* Represents a selector in a parsed Gopher menu.

**host = None**
The hostname of the target server.

**path = None**
Where the item links to on the target server.

**port = None**
The port of the target server. For regular Gopher servers, this is most commonly 70, and for S/Gopher servers it is typically 105.

**request**()
Returns a Request equivalent to where the selector leads.

**source**()
Returns a representation of what the selector looks like in a Gopher menu.

**text = None**
The name, or text that is displayed when the item is in a menu.

**tls = None**
True if the selector leads to an S/Gopher server with TLS enabled.

**type = None**
The type of item.

pituophis.**get**(*host*, *port=70*, *path='/'*, *query=''*, *tls=False*, *tls_verify=True*)
*Client.* Quickly creates and sends a Request. Returns a Response object.

pituophis.**handle**(*request*)
*Server.* Default handler function for Gopher requests while hosting a server. Serves files and directories from the pub/ directory by default, but the path can be changed in serve's pub_dir argument or changing the Request's pub_dir directory.

pituophis.**parse_gophermap**(*source*, *def_host='127.0.0.1'*, *def_port='70'*, *gophermap_dir='/'*)
*Server.* Converts a Bucktooth-style Gophermap (as a String or List) into a Gopher menu as a List of Selectors to send.

pituophis.**parse_menu**(*source*)
*Client.* Parses a String as a Gopher menu. Returns a List of Selectors.

pituophis.**parse_url**(*url*)
*Client.* Parses a Gopher URL and returns an equivalent Request.

pituophis.**serve**(*host='127.0.0.1'*, *port=70*, *handler=<function handle>*, *pub_dir='pub/'*, *alt_handler=False*, *send_period=False*, *tls=False*, *tls_cert_chain='cacert.pem'*, *tls_private_key='privkey.pem'*, *debug=True*)
*Server.* Listens for Gopher requests. Allows for using a custom handler that will return a Bytes, String, or List object (which can contain either Strings or Selectors) to send to the client, or the default handler which can serve a directory. Along with the default handler, you can set an alternate handler to use if a 404 error is generated for dynamic applications.

# Python Module Index

## p

# Index