Programming Techniques

Keyword Arguments

```
course.get_users(search_term='John Doe')
course.get_users(
    search_term='John Doe',
    enrollment_role='StudentEnrollment'
)
```

List Parameters

```
course.get_users(enrollment_type=['teacher', 'student'])
course.get_users(
    enrollment_type=['teacher', 'student'],
    search_term='John',
    include=['email']
)
```

Nested Parameeters

Think of it as the JSON dictionary.

```
account.create_course(course={'name': 'Example Course'})

# pass the course[name],
# course[course_code],
# and course[is_public] arguments
account.create_course(
    course={
        'name': 'Example Course',
        'course_code': 'TST1234',
        'is_public': True
    }
)
```

Exceptions

```
from canvasapi.exceptions import CanvasException

try:
    canvas.get_course(1)
except CanvasException as e:
    print(e)
```

Debugging

```
import logging
import sys

logger = logging.getLogger("canvasapi")
handler = logging.StreamHandler(sys.stdout)
formatter = logging.Formatter(
  '%(asctime)s - %(name)s - %(levelname)s - %(message)s')

handler.setLevel(logging.DEBUG)
handler.setFormatter(formatter)
logger.addHandler(handler)
logger.setLevel(logging.DEBUG)
```

Step 1: Create/Get a logger

```
logger = logging.getLogger("canvasapi")
```

- Gets a named logger called "canvasapi".
- If the canvasapi package internally logs messages, this logger will capture them.
- If it doesn’t exist yet, Python creates it.

Step 2: Create a stream handler

```
handler = logging.StreamHandler(sys.stdout)
```

- A handler decides where logs go.
- StreamHandler(sys.stdout) means logs will be printed to the console (standard output).

```
# File handler
file_handler = logging.FileHandler("canvasapi.log", mode="w")  # "a" for append
```

- To print out results in a file, we use FileHandler.

Step 3: Define the log format

```
formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
```

- %(asctime)s → the timestamp
- %(name)s → the logger name (here: "canvasapi")
- %(levelname)s → severity (DEBUG, INFO, WARNING, etc.)
- %(message)s → the actual log message

Example output:

```
2025-08-18 21:00:00,123 - canvasapi - DEBUG - GET request to /api/v1/courses
```

Step 4: Configure the handler

```
handler.setLevel(logging.DEBUG)
handler.setFormatter(formatter)
```

- The handler will only emit messages at DEBUG level or higher.
- Messages will follow the formatter style.

Step 5: Attach handler to logger

```
logger.addHandler(handler)
```

- Connects the handler (console output) to the canvasapi logger.

Step 6: Set logger's level

```
logger.setLevel(logging.DEBUG)
```

- Ensures the logger captures DEBUG and above messages.
- Without this, the default level (WARNING) would filter out lower-level messages.

```
# A previously configured `Canvas` client
>>> canvas.get_current_user()
2019-07-08 14:22:24,517 - canvasapi.requester - INFO - Request: GET https://base/api/v1/users/self
2019-07-08 14:22:24,517 - canvasapi.requester - DEBUG - Headers: {'Authorization': '****4BSt'}
2019-07-08 14:22:24,748 - canvasapi.requester - INFO - Response: GET https://base/api/v1/users/self 200
2019-07-08 14:22:24,749 - canvasapi.requester - DEBUG - Headers: {'Cache-Control': 'max-age=0, private, must-revalidate',
'Connection': 'keep-alive',
'Content-Encoding': 'gzip',
'Content-Length': '329',
'Content-Type': 'application/json; charset=utf-8',
2019-07-08 14:22:24,749 - canvasapi.requester - DEBUG - Data: {'avatar_url': 'https://base/images/thumbnails/43244/Umo5dyAg0OS3tpDtDN',
'created_at': '2014-08-22T08:02:00-04:00',
'effective_locale': 'en',
'email': '',
'id': XXXX181,
'integration_id': None,
'locale': None,
'login_id': 'XXXXXXXX181',
'name': 'Some User',
'permissions': {'can_update_avatar': True, 'can_update_name': False},
'root_account': 'xxxxxx.edu',
'short_name': 'Some User',
'sis_user_id': 'XXXXXXXX181',
'sortable_name': 'User S'}
```