3_canvasapi_quick_reference.py

In this example, we demonstrate how to make Python functions that manages Canvas with canvasapi package.

Function Descriptions

1. Course Information

get_course_info()

```
def get_course_info(course):
  print("\n1. 📊 COURSE INFORMATION")
  name = course.name;
  students = list(course.get_users(enrollment_type=['student']))
  assignments = list(course.get_assignments())
  print(f"Students: {len(students)}")
  print(f"Assignments: {len(assignments)}")
  print(f"Course Name: {name}")
```

- The course.get_users() also returns the PaginatedList.
- We need to turn the PaginatedList into Python List to get the students count.

2. CREATE ASSIGNMENT

get_assignment_groups()

```
def get_assignment_groups(course):
  print("\n2-1. 📄 ASSIGNMENT GROUPS")
   # /courses/:course_id/assignment_groups
  assignment_groups = course.get_assignment_groups()
  for group in assignment_groups:
    print(group.id, group.name)
```

```
2-1. 📄 ASSIGNMENT GROUPS
237085 Assignments
237086 Exam
237087 Feedback & Bonus
237088 Quiz
237089 Project
```

create_assignment()

```
def create_assignment(course, name, description, due_days_from_now=7, points=100):
  print("\n2. 📝 CREATE ASSIGNMENT")
  due_date = (datetime.now()
    + timedelta(days=due_days_from_now)).strftime("%Y-%m-%dT23:59:59Z")

  assignment_data = {
    'name': name,
    'description': description,
    'due_at': due_date,
    'points_possible': points,
    'submission_types': ['online_upload'],
    'allowed_extensions': ['pdf', 'doc', 'docx', 'zip'],
    'assignment_group_id': 237086, # exam
  }

  assignment = course.create_assignment(assignment_data)
  return assignment
```

Usage

We can call this function.

```
create_assignment(course,
  "Sample Assignment",
  "<p>This is a test assignment</p>")
```

3. Creating Quizzes

Basic Quiz Creation

```
def create_quiz(course, title, description, time_limit=60):
  quiz_data = {
    'title': title,
    'description': description,
    'quiz_type': 'assignment',
    'time_limit': time_limit,  # minutes
    'allowed_attempts': 1,
    'scoring_policy': 'keep_highest'
  }

  quiz = course.create_quiz(quiz_data)
  print(f"✅ Created quiz: {title}")
  return quiz
```

Question/Answers of the Quiz

```
def add_mc_question(quiz, question_text, answers, correct_answer):
  question_data = {
    'question_name': question_text[:50] + "...",
    'question_text': question_text,
    'question_type': 'multiple_choice_question',
    'points_possible': 1,
    'answers': answers
  }

  question = quiz.create_question(question=question_data)
  return question
```

create_simple_quiz()

```
def create_simple_quiz(course, title, questions_list, time_limit=30):
  print("\n3. ❓ CREATE QUIZ")
  # Create quiz
  quiz_data = {
    'title': title,
    'quiz_type': 'assignment',
    'time_limit': time_limit,
    'allowed_attempts': 1
  }
  quiz = course.create_quiz(quiz_data)
  # Add questions
  for q in questions_list:
    question_data = {
      'question_name': q['question'][:50],
      'question_text': q['question'],
      'question_type': 'multiple_choice_question',
      'points_possible': 1,
      'answers': q['answers']
    }
    quiz.create_question(question=question_data)

  print(f"✅ Created quiz: {title} with {len(questions_list)} questions")
  return quiz
```

Usage

```
quiz = create_simple_quiz(course, "Sample Quiz", sample_questions)
```

sample questions

```
sample_questions = [
  {
    'question': 'What is 2 + 2?',
    'answers': [
      {'answer_text': '3', 'answer_weight': 0},
      {'answer_text': '4', 'answer_weight': 100},  # Correct answer
      {'answer_text': '5', 'answer_weight': 0}
    ]
  }
]
```

Publish quiz

```
quiz.edit(quiz={'published': True})
```

4. Creating Pages

create_course_page()

```
def create_course_page(course, title, content, \
  is_front_page=False, published=False):
  print("\n4. 📄 CREATE PAGE")
  page_data = {
    'title': title,
    'body': content,
    'published': published,
    'front_page': is_front_page
  }
  page = course.create_page(wiki_page=page_data)
  print(f"✅ Created page: {title}")
  return page
```

Usage

```
# Example: Course syllabus
syllabus_content = """
<h2>Course Syllabus</h2>
<h3>Course Description</h3>
<p>This course covers...</p>
<h3>Learning Objectives</h3>
<ul>
  <li>Understand basic concepts</li>
  <li>Apply theoretical knowledge</li>
</ul>
"""

create_page(course, "Syllabus", syllabus_content, front_page=True)
```

Using Markdown Instead of HTML

Why Markdown?

- **Easier to write** - Natural syntax
- **More readable** - Clean source files
- **Version control friendly** - Better diffs
- **Template reusable** - Save as `.md` files

markdown.markdown()

```
import markdown

def markdown_to_html(markdown_content):
  return markdown.markdown(markdown_content,
                extensions=['extra', 'codehilite', 'toc'])
```

Usage

```
markdown_content = """
## Course Syllabus
...
"""
```

```
def create_page_from_markdown(course,
  title, markdown_content):
  html_content = markdown_to_html(markdown_content)
  return create_page(course, title, html_content)
```

5. Upload File

Display folders in the Course

```
def display_folders(course):
  folders = course.get_folders()
  for folder in folders:
  print(f"Name: {folder.name},
    ID: {folder.id}, Parent ID: {folder.parent_folder_id},
    Hidden: {folder.hidden}")
```

- In this example, the `flutter_advanced` has parent 143254, which is the `lectures` directory, so we can see that `flutter_advanced` is a subfolder in the `lectures`.

```
Name: course_image, ID: 1460528, Parent ID: 1432286, Hidden: None
Name: flutter_advanced, ID: 1432355, Parent ID: 1432354, Hidden: None
Name: lectures, ID: 1432354, Parent ID: 1432286, Hidden: None
```

This examples shows how to find the folder from the folder_name and create the folder if it does not exist.

```
folders = course.get_folders()
target_folder = None
for folder in folders:
  if folder.name == folder_name: target_folder = folder
    break

if not target_folder:
  target_folder = course.create_folder(folder_name)
```

upload_file_to_course()

```
def upload_file_to_course(course, file_path,
    folder_name="Lectures"):
  try:
    folders = course.get_folders()
    target_folder = None
    for folder in folders:
      if folder.name == folder_name: target_folder = folder
        break

    if not target_folder:
      target_folder = course.create_folder(folder_name)

    # Upload file
    response = target_folder.upload(file_path)
    print(f"✅ Uploaded: {file_path}")
    return response
  except Exception as e:
    print(f"❌ Error: {e}")
```

- We can specify the folder name.

```python
def upload_file_to_default_course(course, file_path):
  try:
    response = course.upload(file_path)
    print(f"✅ Uploaded: {file_path}")
    return response[1]['url']  # Return file URL
  except Exception as e:
    print(f"❌ Upload failed: {e}")
    return None
```

- Without specifying the target folder, the file is uploaded to the default folder ("unfiled").

response[1]['url']

We can get the URL to download the image from response[1]['url'].

https://nku.instructure.com/files/12717290/download?download_frd=1&verifier=hQthlY1D1wKeW84NoFnnSY3GuNzfNFXNYk9FG5OZ

The course image can be embedded as follows:

```
<img id="12717290" src="/courses/81923/files/12717290/preview" alt="code.png" /></p>
```

6. Creating Assignments (Homework)

create_homework()

```
def create_homework(course, title, description, due_date, points=100):
    assignment_data = {
        'name': title,
        'description': description,
        'due_at': due_date,  # Format: "2024-12-31T23:59:59Z"
        'points_possible': points,
        'submission_types': ['online_upload'],
        'allowed_extensions': ['pdf', 'doc', 'docx']
    }

    assignment = course.create_assignment(assignment_data)
    print(f"✅ Created assignment: {title}")
    return assignment
```

usage

```
# Example usage
homework = create_homework(
    course=course,
    title="Week 5 Problem Set",
    description="<p>Solve problems 1-10 from Chapter 5</p>",
    due_date="2024-03-15T23:59:59Z",
    points=50
)
```

Checking if the assignment exists

```
def assignment_exists(course, title):
    assignments = course.get_assignments()  # This fetches all assignments in the course
    for assignment in assignments:
        if assignment.name == title:
            return assignment.id
    return -1
```

We can get the assignment using the assignment.id.

```
if id > 0:
    print(f"❌ Assignment '{title}' already exists")
    hw = course.get_assignment(id)
```

Using the ID to Make a Link

With three pieces of information, we can create an internal link to the homework assignment.

- course ID (81929)
- assignments
- assignment ID (1440127)

```
href="/courses/81929/assignments/1440127"
```

7. Download Files from Course

download_assignment_submissions()

From the assignment_name, we can find the assignment object.

```
def download_assignment_submissions(course, assignment_name, download_folder="downloads"):
  # Find assignment
  target_assignment = None
  for assignment in course.get_assignments():
    if assignment.name == assignment_name:
      target_assignment = assignment
      break

  if not target_assignment:
    print(f"❌ Assignment '{assignment_name}' not found")
    return
```

Submission Object

```
{
  'id': 44696615,
  'assignment_id': 1378002,
  'assignment_id_date': datetime.datetime(1378, 1, 2, 0, 0, tzinfo=<UTC>),
  'user_id': 13454,
  'preview_url':
    'https://nku.instructure.com/courses/81929/assignments/1378002/submissions/13454?preview=1&version=0',
  'course_id': 81929
}
```

The Submission Object has all the information, including student id (user_id), course (course_id), and assignment(assignment_id).

Download submissions

- From the assignment object we can get all the submissions.
- From the submission, we can get the attachment and student id (user_id).

```
os.makedirs(download_folder, exist_ok=True)
submissions = target_assignment.get_submissions()
count = 0

for submission in submissions:
  if submission.attachments:
    for attachment in submission.attachments:
      filename = attachment.filename if hasattr(attachment, 'filename') else f"file_{count}"
      student_name = getattr(submission, 'user_id', 'unknown_student')
      student_folder = os.path.join(download_folder, '') # f"student_{student_name}")

      print(f"  📄 Downloading: {student_name}:{filename}")
      file_path = os.path.join(student_folder, filename)
      attachment.download(file_path)
      count += 1

print(f"✅ Would download {count} submissions to {download_folder}")
```

8. Student Analytics

show_course_analytics

```
def show_course_analytics(course):
  students = list(course.get_users(enrollment_type=['student']))
  assignments = list(course.get_assignments())

  print(f"📊 Course Analytics:")
  print(f"   Total students: {len(students)}")
  print(f"   Total assignments: {len(assignments)}")

  # Show submission rates for recent assignments
  assignments = sorted(course.get_assignments(), key=lambda a: (a.due_at or ""), reverse=True)
  recent_assignments = assignments[:3] if len(assignments) >= 3 else assignments
  for assignment in recent_assignments:
    submissions = list(assignment.get_submissions())
    submitted_count = sum(1 for s in submissions if s.submitted_at)
    percentage = (submitted_count / len(students)) * 100 if students else 0
    print(f"   {assignment.name}: {submitted_count}/{len(students)} ({percentage:.1f}%)")
```

Useful Bulk Operations

Create Multiple Assignments

```
assignments_data = [
    {"name": "Assignment 1", "due": "2024-03-01T23:59:59Z", "points": 50},
    {"name": "Assignment 2", "due": "2024-03-08T23:59:59Z", "points": 75},
    {"name": "Assignment 3", "due": "2024-03-15T23:59:59Z", "points": 100}
]

for assignment_info in assignments_data:
    create_homework(
        course=course,
        title=assignment_info["name"],
        description=f"<p>Complete {assignment_info['name']}</p>",
        due_date=assignment_info["due"],
        points=assignment_info["points"]
    )
```

Bulk Grade Update

In this example, we can update (edit) the students in the `grades_dict` list.

```
grades_dict = {
    12345: 95,          # User 12345 gets 95 points
    23456: 87,          # User 23456 gets 87 points
    45678: 76.5,        # User 45678 gets 76.5 points
}
```

```
def update_grades(course, assignment_id, grades_dict):
    assignment = course.get_assignment(assignment_id)
    submissions = assignment.get_submissions()

    for submission in submissions:
        user_id = submission.user_id
        if user_id in grades_dict:
            submission.edit(submission={'posted_grade': grades_dict[user_id]})
            print(f"Updated grade for user {user_id}")
```