MARKING GUIDE - Reflective Journals

ISYS2002 Introduction to Business Programming

Table of contents

General Guidelines for Giving Feedback 1
Feedback Examples 2
Example Feedback Template 3

General Guidelines for Giving Feedback

Providing constructive and effective feedback is crucial for student learning, especially in courses that involve practical skills like programming. Here's some general guidance for markers on how to give feedback to students:

1. **Be Specific and Constructive:**
   - **Specificity:** Avoid vague feedback. Instead, specify exactly what the student did well or needs to improve. For example, if a student's Python notebook has errors, point out the specific lines or logic that caused the issue.
   - **Constructiveness:** Feedback should not just criticise but also guide. Offer suggestions on how they can improve, such as recommending resources or techniques to master difficult concepts.
2. **Balance Positives and Negatives:**
   - **Positive Reinforcement:** Always try to find something positive to say, even in work that needs significant improvement. This encourages students and makes them more receptive to criticism.
   - **Critical Feedback:** Balance your feedback by also pointing out areas that need improvement, but do so in a way that motivates and educates.
3. **Encourage Self-Reflection:**

- Ask questions in your feedback that encourage students to reflect on their own learning processes. For example, “What do you think went wrong here?” or “How could this code be improved for efficiency?”
- Encouraging self-assessment helps students develop critical thinking and self-learning skills.

4. **Use Clear and Respectful Language:**

   - Avoid jargon unless you are sure the student understands it. Use clear, straightforward language.
   - Always maintain a respectful and professional tone, even when providing negative feedback.

5. **Offer Examples and Resources:**

   - When possible, provide examples that clearly illustrate high-quality work or correct approaches to problems.
   - Recommend specific resources such as tutorials, books, or lectures that can help the student understand the concepts better.

6. **Follow-Up:**

   - Offer to discuss the feedback in more detail if the student has questions or needs further clarification. This can be through office hours, emails, or scheduled appointments.
   - Checking back on progress after feedback shows students that you are invested in their improvement and success.

Feedback Examples

Below are examples of feedback comments for journals submitted by beginner programmers, categorised into poor, good, and best levels based on the quality of their reflective writing and the functionality of attached Python notebooks. These comments aim to provide constructive feedback to help students understand their performance and areas for improvement.

Poor Journal Feedback

Feedback for Reflective Writing: “Your journal entries need more depth in reflecting on the learning process. It’s important to not only describe what you did but also to consider what you learned from the experience, the challenges you faced, and how you can apply this knowledge in the future. Try to make your reflections more personal and insightful.”

Feedback for Python Notebooks: “The attached Python notebooks contain several errors that prevent them from running successfully. It looks like you might be struggling with some basic concepts. I recommend reviewing the recent modules on Python syntax and looping structures. Please visit the learning resources section for additional tutorials to help you understand these concepts better."

Good Journal Feedback

Feedback for Reflective Writing: "Your journal entries show a good understanding of the topics covered each week. You've done well describing the activities and summarising the content. To enhance your reflections, try to include more about your personal learning experience, such as any difficulties you encountered and how you overcame them. Reflecting on these experiences can provide deeper insights into your learning process."

Feedback for Python Notebooks: "Your Python notebooks are mostly functional, but there are a few minor errors that could be easily fixed. It seems like you understand the basic programming concepts, which is great. To improve, make sure to test your code thoroughly before submission and consider edge cases that might cause errors. Keep practising, and you'll continue to improve."

Best Journal Feedback

Feedback for Reflective Writing: "Excellent reflective writing! You have not only detailed what you have learned each week but also critically analysed how these new skills can be applied to real-world problems. Your reflections show a deep understanding of the learning material and a thoughtful consideration of your personal growth as a programmer. Keep up the great work!"

Feedback for Python Notebooks: "Your Python notebooks are well-organised, cleanly coded, and function flawlessly. It's clear that you have put a lot of effort into understanding the course material and applying it effectively in your code. Your ability to incorporate advanced features and maintain readability is impressive. Excellent job on demonstrating your programming skills!"

These examples aim to guide students on how they can improve their submissions and recognize the strengths and weaknesses in their current work.

Example Feedback Template

If you are stuck, here's a template that can help structure your feedback:

> Great effort on [specific task]. I noticed that you [mention something done well], which shows your understanding of [specific concept]. This is a great approach because [reason why it was good].

I would like to see some improvement in [area of improvement]. Currently,[describe the issue briefly]. You might find it helpful to review [specific resource or topic]. For example, [give a brief example of how to improve].

If you have any questions about this feedback or need further assistance, please don't hesitate to reach out.

Assessing Consistency

- **Check Creation Date:** Verify that each journal entry was created on or before the due date listed in the teaching schedule of the Unit Outline. Journals are due the night before the Monday listed.
  - **Identifying Dates:** The date displayed under the journal title is the creation date. If two dates are shown, the first is the creation date and the second is the last edited date, which you should ignore.
  - **Scoring:** Award 100% for consistency if all creation dates are valid. Deduct 15% for each late journal entry, then select the appropriate mark in the "Consistency" row of the rubric.

Assessing Completeness

- **Review Journal Content:** Focus on the body of the journal and any attached evidence of learning. Assess engagement with the topic and functionality of included notebooks.
  - **Notebook Functionality:** It's acceptable if some notebooks have minor issues. Generally, the majority should be functional.
  - **Sampling:** Randomly check a few notebooks across the six entries. If they function properly and the journal entry is satisfactory, award full marks. Otherwise, subtract 5-15% for each incomplete week.
  - **Scoring:** Select the appropriate mark on the "Completeness" row in the rubric.

Accessing and Using the Rubric

- **Locate the Rubric:** The rubric is located in the grade centre.
  - Navigate to the grade centre, find the student, and hover over the icon under 'Reflective Journal'.
  - Click the down arrow that appears and select 'Mark User Activity'. This opens the student's journal.
- **Review Journals:** The student's latest journal entry appears in the centre, with other entries shown towards the bottom right. Inspect all journals to make your assessment. Journal' link, located at the top right below the student's name and next to

- **Complete the Rubric:** The rubric can be accessed by clicking the ‘Weekly ’used for marking’.
  - Repeat this process for each student.

These steps will guide you through the assessment process for each student’s weekly journal submissions. Please ensure accuracy and fairness in your evaluations.