Skills assessment: Object-oriented programming tasks

Criteria

Unit code, name and release number

ICTPRG430 - Apply introductory object-oriented language skills (1)

Qualification/Course code, name and release number

ICT50418 - Diploma of Information Technology Networking (1)

Student details

Student number

Student name

Assessment Declaration

- This assessment is my original work and no part of it has been copied from any other source except where due acknowledgement is made.
- No part of this assessment has been written for me by any other person except where such collaboration has been authorised by the assessor concerned.
- I understand that plagiarism is the presentation of the work, idea or creation of another person as though it is your own. Plagiarism occurs when the origin of the material used is not appropriately cited. No part of this assessment is plagiarised.

Student signature and Date

Version:	20200120
Date created:	6 February 2020
Date modified:	20 January 2020

For queries, please contact:

Technology and Business Services SkillsPoint

Ultimo

This assessment can be found in the: Learning Bank

Assessment instructions

Table 1 Assessment instructions

Assessment details	Instructions
Assessment overview	The objective of this assessment is to assess your skills required to complete object-oriented programming tasks.
Assessment Event number	3 of 3
Instructions for this assessment	This is a skill-based assessment and will be assessing you on your ability to demonstrate skills required in the unit. This assessment is in one part: 1. Practical tasks. The assessment also contains: • Assessment Feedback.
Submission instructions	On completion of this assessment, you are required to submit it to your assessor for marking. Ensure you have written your name at the bottom of each page of this assessment. Submit the following documents for each part: • Part 1: Practical tasks o A zipped folder including: ▪ completed Python modules for Tasks 1-5 ▪ character-count.txt file for Task 4. It is important that you keep a copy of all electronic and hardcopy assessments submitted to TAFE and complete the assessment declaration when submitting the assessment.

Assessment details	Instructions
What do I need to do to achieve a satisfactory result?	To successfully complete this assessment the student will be available at the arranged time to complete all the assessment criteria as outlined in the assessment instructions. All parts of the observable task must be performed to a satisfactory level. All oral questions must be answered correctly to be deemed satisfactory in this assessment task: however, Assessors may ask questions to clarify understanding.
Assessment conditions	Assessment conditions will be safe and replicate the workplace. Noise levels, production flow, interruptions and time variances must be typical of those experienced in the programming and software development field of work. Assessment may be undertaken in normal classroom conditions, which is assumed to be noisy and similar to workplace conditions, or within the workplace. This may include phones ringing, people talking and other interruptions.
What do I need to provide?	• USB drive or other storage method with enough free space to save work to.
What will the assessor provide?	• An integrated development environment (IDE) • Applications for software development (PyCharm Community Edition) • Files provided in the resource folder (Dip_Nwk_Programming_AE_Sk_3of3_SR1.zip)
Due date/time allowed	Time allowed is two hours.
Assessment location	This assessment will be completed in the classroom. The student may access their referenced text, learning notes and other resources.
Supervision	This is a supervised assessment.

Assessment details	Instructions
Reasonable adjustment	If you have a permanent or temporary condition that may prevent you from successfully completing the assessment event(s) in the way described, you should talk to your assessor about 'reasonable adjustment'. This is the adjustment of the way you are assessed to take into account your condition, which must be approved BEFORE you attempt the assessment.
Assessment feedback, review or appeals	In accordance with the TAFE NSW policy *Manage Assessment Appeals,* all students have the right to appeal an assessment decision in relation to how the assessment was conducted and the outcome of the assessment. Appeals must be lodged within **14 working days** of the formal notification of the result of the assessment. If you would like to request a review of your results or if you have any concerns about your results, contact your Teacher or Head Teacher. If they are unavailable, contact the Student Administration Officer. Contact your Head Teacher for the assessment appeals procedures at your college/campus.

Specific task instructions

Download and unzip the resource folder (Dip_Nwk_Programming_AE_Sk_3of3_SR1.zip) for files referred to within the assessment. This folder includes a folder with five Python modules. Each module is related to an assessment task. The assessment has five tasks in total.

Before starting the assessment ensure you can open and successfully run each of the modules. It is recommended you use PyCharm for this assessment, by opening the folder titled **ictprg430-skills-test** as a project.

Each task below provides example console output of what is expected.

Part 1: Practical tasks

Task 1

Add logic to the **main** function of the Task 1 module for processing a mathematical expression on two integer values with a single operator, as follows:

The order of operations is as follows:

1. Prompt the end-user to enter an integer value.
2. Prompt the end-user to select an operator – the available operators are minus, plus, multiplication and division [- + * /].
3. Prompt the end-user to enter a second integer value.
4. Display the result of the expression in words.

See **Figures 1 – 4** for example expected console output.

```
Enter an integer value: 10
Enter an operator [ Subtract - , Plus + , Multiply * or Division / ]: -
Enter another integer value: 5
10 minus 5 equals 5
```

Figure 1 - Subtraction Example: 10 – 5 = 5

```
Enter an integer value: 5
Enter an operator [ Subtract - , Plus + , Multiply * or Division / ]: +
Enter another integer value: 5
5 plus 5 equals 10
```

Figure 2 - Addition Example: 5 + 5 = 10

```
Enter an integer value: 5
Enter an operator [ Subtract - , Plus + , Multiply * or Division / ]: *
Enter another integer value: 5
5 times 5 equals 25
```

Figure 3 - Multiplication Example: 5 * 5 = 25

```
Enter an integer value: 25
Enter an operator [ Subtract - , Plus + , Multiply * or Division / ]: /
Enter another integer value: 5
25 divided 5 equals 5
```

Figure 4 - Division Example: 25 / 5 = 5

Task 2

Add logic to the main function of the Task 2 module for calculating the cost of booking a room.

- **Parallel lists** must be used for storing the name and price of rooms.
- You must iterate through the lists and display the name and price of each room.
- The end-user can select from the list of rooms and associated prices provided below:
    1. Single $45.50 per night
    2. Double $99.99 per night
    3. Luxury $165.25 per night
    4. Penthouse $1,095.50 per night

You are not required to validate values entered by the end-user and it is assumed only valid values will be entered i.e. the room type (1, 2, 3 or 4) and the number of nights (a positive integer value).

See **Figures 5 – 8** for example expected console output.

```
Room Types

1. Single $45.50 per night
2. Double $99.99 per night
3. Luxury $165.25 per night
4. Penthouse $1,095.50 per night

Please select a room type 1, 2, 3 or 4 :1
Please enter the number of nights :5

Thank you, the total cost for staying in the Single room for 5 nights is $227.50
```

Figure 5 - Single room for 5 nights - $45.50 * 5 = $227.50

```
Room Types

1. Single $45.50 per night
2. Double $99.99 per night
3. Luxury $165.25 per night
4. Penthouse $1,095.50 per night

Please select a room type 1, 2, 3 or 4 :2
Please enter the number of nights :3

Thank you, the total cost for staying in the Double room for 3 nights is $299.97
```

Figure 6 - Double room for 3 nights - $99.99 * 3 = $299.97

```
Room Types

1. Single $45.50 per night
2. Double $99.99 per night
3. Luxury $165.25 per night
4. Penthouse $1,095.50 per night

Please select a room type 1, 2, 3 or 4 :3
Please enter the number of nights :2

Thank you, the total cost for staying in the Luxury room for 2 nights is $330.50
```

Figure 7 - Luxury room for 2 nights - $165.25 * 2 = $330.50

```
Room Types

1. Single $45.50 per night
2. Double $99.99 per night
3. Luxury $165.25 per night
4. Penthouse $1,095.50 per night

Please select a room type 1, 2, 3 or 4 :4
Please enter the number of nights :1

Thank you, the total cost for staying in the Penthouse room for 1 nights is $1,095.50
```

Figure 8 - Penthouse room for 1 night - $1,095.50 * 1 = $1,095.50

Task 3

Complete the logic for the classes Person, Student and Teacher (shown in **Figure 9 - Person, Student and Teacher Inheritance)** as follows:

- The Person, Student and Teacher classes have already been added to the Task 3 module. Ensure they have all necessary initializers, attributes, and methods, as per **Figure 9**
- Implement inheritance, initializer delegation and polymorphism so the print_details method for Student and Teacher behave differently as shown in **Figure 10 - Console Output, Inheritance and polymorphism**.

Complete the following tasks:

1. Initialise the *_name* field in the *Person* class initializer – this requires assigning the *name* parameter passed in as an initializer argument to the *_name* attribute.
2. Add an initializer to the *Student* class; the initializer must have two parameters, *str* name and *List[str]* subjects. Because the *Student* class inherits from *Person,* the *name* parameter needs to be passed to the base class initializer. The *subjects* parameter needs to be assigned to the *_subjects* attribute.
3. Complete the logic in the *Student print_details* method so the student's name and list of subjects studied is printed to the console as shown in **Figure 10**.

4. Add an initializer to the *Teacher* class; the initializer must have two parameters, *str* name and *str* faculty. Because the Teacher class inherits from Person, the *name* parameter needs to be passed to the base class initializer. The *faculty* parameter needs to be assigned to the *_faculty* attribute.
5. Complete the logic in the *Teacher print_details* method so the teacher's name and faculty are printed to the console as shown in **Figure 10**.
6. Add logic to the main function for instantiating and adding a new *Student* object to the existing list of *Person* objects – *people*. The student name and list of subjects have been provided.
7. Add logic to the main function for instantiating and adding a new *Teacher* object to the existing list of *Person* objects – *people*. The teacher name and faculty have been provided.
8. Iterate through the people collection and invoke the *print_details* method for each element in the collection. The output must match that provided in **Figure 10 - Console Output, Inheritance and polymorphism**

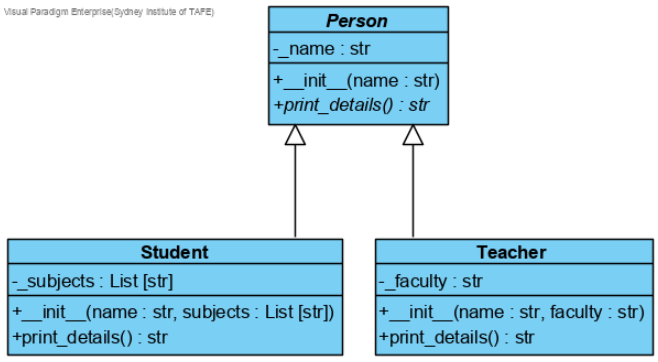

Figure 9 - Person, Student and Teacher Inheritance

```
Hi my name is Sue and I am studying Math,Science,English
Hi my name is Tim and I teach in the Computer Science faculty
```

Figure 10 - Console Output, Inheritance and polymorphism

Task 4

Add logic to the main function of the Task 4 module for counting instances of alphabetic characters in a provided text file and writing the result of the count to another text file. The two text files have already been provided in the module as seen in **Figure 11 - Task 4, characters and character-count text files**. All characters in the characters.txt file are uppercase so there is no need to consider casing.

- Read in the text from the provided *characters.txt* file.
- Iterate through each alphabetic character [A – Z].
- Count each instance of a specific character, for example, there are 40 instances of the character 'P' in the *characters.txt* file.
- Write the result to the *character-count.txt* text file.

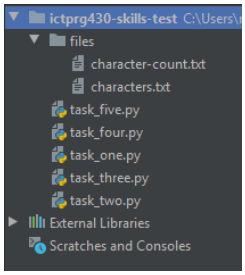

Figure 11 - Task 4, characters and character-count text files

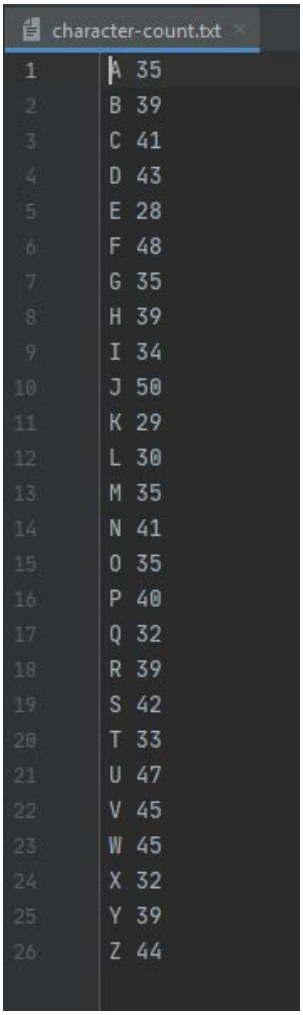

Figure 12 - character-count.txt

Task 5

Add logic to the main function of the Task 5 module to display the numbers 50 down to 25 with the following conditions:

- If the number is a multiple of 3, display [3] on the right-side of the number.
- If the number is a multiple of 5, display [5] on the right-side of the number.
- If the number is a multiple of 3 and 5, display [3 - 5] on the right-side of the number.

Note: A number is a multiple when it can be evenly divided without a remainder.

Your logic must use a **loop** and only use the *print* function a maximum of four times.

For example:

- 9 is a multiple of 3, as it can be divided evenly by 3: **9 / 3 = 3**.
- 10 is a multiple of 5 as it can be divided evenly by 5: **10 / 5 = 2.**
- 15 is a multiple of 3 and 5 as it can be evenly divided by both numbers: **15 / 3 = 5 and 15 / 5 = 3.**

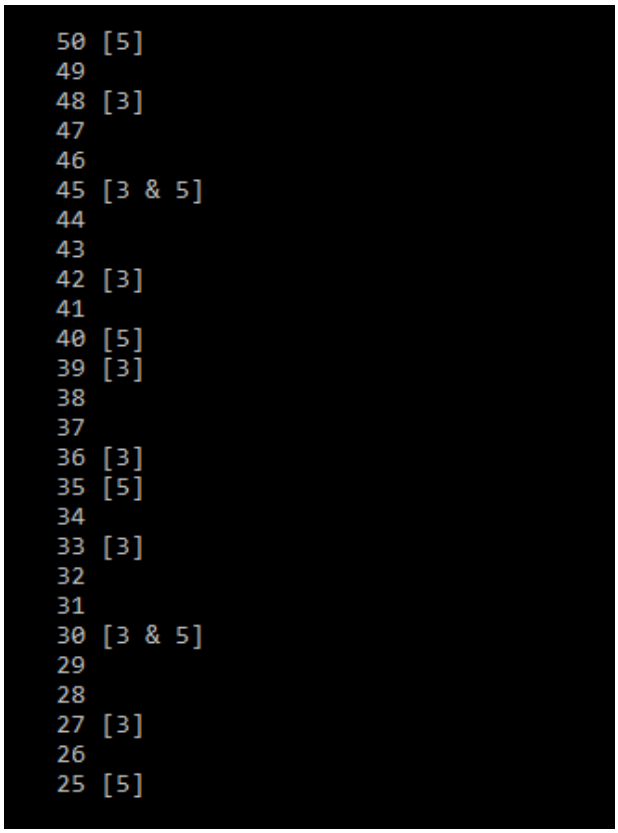

Figure 13 – Task 5, Example Console Output

Assessment Feedback

*NOTE: This section **must** have the assessor signature and student signature to complete the feedback.*

Assessment outcome

☐ Satisfactory

☐ Unsatisfactory

Assessor feedback

☐ Has the Assessment Declaration been signed and dated by the student?

☐ Are you assured that the evidence presented for assessment is the student's own work?

☐ Was the assessment event successfully completed?

☐ If no, was the resubmission/re-assessment successfully completed?

☐ Was reasonable adjustment in place for this assessment event?

If yes, ensure it is detailed on the assessment document.

Comments:

Assessor name, signature and date:

Student acknowledgement of assessment outcome

Would you like to make any comments about this assessment?

Student name, signature and date

NOTE: Make sure you have written your name at the bottom of each page of your submission before attaching the cover sheet and submitting to your assessor for marking.