e-Portfolio Activities

1. **Review** the article by Di Silvestro & Nadir (2021). Discuss one aspect of this article which you find unexpected.

My observations:

This article discusses various topics related to adult education, including how ePortfolios might encourage deeper learning and reflection in graduate adult education. The article also looks at Saudi Arabian adult learners' motivations for learning. It focuses on the acculturation experiences of Syrian Muslim refugee women in the United States from the standpoint of intersectionality theory. The article analyses these women's issues and the importance of providing long-term education courses for immigrants and refugees to aid their effective acculturation. The study also highlights how ePortfolios might be a helpful addition to graduate adult education programmes to foster thoughtful and in-depth learning.

The fact that students were surprised by the demand for collaboration in developing their ePortfolios is an unexpected aspect of this article.

Building ePortfolios, based on the pleasant experiences of the students, effectively enhances our learning because it requires substantial reading and research, allowing us to learn and master the subject in depth.

Speaking from my own experience as an immigrant and native Portuguese speaker, full-time worker, and father of three children who need constant attention, I could learn about time management in a short time.

Studying this taught course raised my knowledge above average, increasing my understanding so quickly in such a short time. When I started researching to answer questions about the units I'm studying and to update my ePortfolio with study materials,

I came across so many research topics, which motivated and delighted me to be learning. I learnt about the value of resilience.

2. **Develop a Python program and apply protected and unprotected variables.**

Protected variables are data members of a class that can only be accessed within that class and its descendants. In Python, there are no "Public" instance variables. For determining who has access control of a data member in a class, however, we use the underscore '_' symbol. Thus, any member prefixed with an underscore, whether a function, method, or data member, must be handled as a non-public API or any Python code component (GeeksforGeeks, 2020).

Access control is a concept in most object-oriented programming languages; this has something to do with abstraction. Some attributes and methods on an object are designated as private, meaning only that object can access them. Others are marked protected, meaning only that class and its subclasses can access them. The rest are public, meaning any other object can access them (Phillips, 2018).

```
class Person:
    def __init__(self, name, age):
        self.name = name                    # Unprotected variable
        self._age = age                     # Protected variable

    def get_age(self):
        return self._age

    def set_age(self, age):
        self._age = age


person1 = Person("Hainadine", 45)

print(person1.name)                         # Output: Hainadine
print(person1.get_age())                    # Output: 45

person1._age = 40                           # Unprotected access to protected variable
print(person1.get_age())                    # Output: 40

person1.set_age(50)                         # Protected access to protected variable
print(person1.get_age())                    # Output: 50
```

References:

GeeksforGeeks. (2020). *Protected variable in Python*. [online] Available at: https://www.geeksforgeeks.org/protected-variable-in-python/ [Accessed 6 May 2023].

Phillips, D. (2018). *Python 3 object-oriented programming: Build robust and maintainable software with object-oriented design patterns in Python 3.8*. Packt Publishing Ltd.

Unit 1 - Reflection

During my initial week of studying Object-oriented Programming, I was introduced to the Lecture cast, *Introduction to Python and the Object-Oriented Programming Philosophy,* to be an excellent resource for learning about Object-oriented programming. It offered valuable insights into the emergence of OOP and its role in the context of other programming languages. I also learned about the significant advantages of an object-oriented approach, such as encapsulation, abstraction, inheritance, and polymorphism. Moreover, I understood the types of data that can be used in an object-oriented program and the challenges associated with writing one.

In the second topic of the first unit, I delved into the intriguing realm of Python programming. The focus was on refreshing my understanding of Classes and Objects in Codio. This gave me a thorough introduction to Python programming and valuable insights into 'programmer-defined types', 'attributes', 'rectangles', 'instances as return values', 'objects are mutable' and 'copying'.

Additionally, I participated in a Collaborative Discussion Forum focused on "Factors which Influence Reusability". The discussion was based on an article by Padhy et al. (2018) and revolved around a list of factors presented in Table 1 of the article. During the discussion, I prioritised this list and explained my reasoning for the assigned priorities. My insights and rationale have been documented in my ePortfolio.

In the final activities of this unit, I focused on e-Portfolio Activities. This involved reviewing an article by Di Silvestro & Nadir (2021) and discussing one specific aspect. I have documented my findings on my ePortfolio.

I have just finished Unit 1 of my Object-oriented Programming module, and I am confident about my grasp of Python programming and its crucial role in the OOP

paradigm. I am excited to put my newfound knowledge to use and keep progressing in my programming journey with Python. With a strong foundation in Python programming and OOP principles, I will be ready to confidently tackle more complex projects. I am fully committed to deepening my understanding of Python's fundamental concepts and OOP. My skills in developing reliable and efficient software solutions will only improve with time.