CoGrammar

SE PORTFOLIO SESSION 12

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

Software Engineering Lecture Housekeeping

- The use of disrespectful language is prohibited if asking a question. This is a supportive, learning environment for all – please engage accordingly! **(FBV: Mutual Respect.)**
- No question is 'silly' – **ask away!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Open Classes. You can submit these questions here: **Open Class Questions**

Software Engineering Lecture Housekeeping cont.

- For all **non-academic questions**, please submit a query: **www.hyperiondev.com/support**
- Report a **safeguarding** incident: **www.hyperiondev.com/safeguardreporting**
- We would love your **feedback** on lectures: **Feedback on Lectures**

Unlock Prestigious Co-Certification Opportunities

New Partnerships Unveiled!

- **University of Manchester & Imperial College London** join our circle along with The University of Nottingham Online.

Exclusive Opportunity:

- Co-certification spots awarded on a first-come basis.
- Meet the criteria early to gain eligibility for the co-certification.

Key Deadlines:

- **11 March 2024**: 112 Guided Learning Hours & 'Build Your Brand' tasks completion.
- **18 March 2024**: Record interview invitation or self-employment.
- **15 July 2024**: Submit verified job offer or new contract.

CoGrammar

Which of the following statements are true about the __str__ method in Python classes?

A. It converts an object to an integer.
B. It is automatically called when an object is created.
C. It returns a string representation of an object.
D. It is used for defining private methods.

Which function in Python can be used to check if an object has a specific attribute?

A. hasattr()
B. checkattr()
C. attribute_exists()
D. obj_attr_exists()

Recap of OOP

Encapsulation

- The bundling of data (attributes) and the methods (functions) that operate on the data into a single unit known as a class. It involves restricting direct access to some of an object's components and preventing the accidental modification of the object's internal state.

Polymorphism

- The ability for a function to receive different types of objects and exhibit different behaviours based on the given object type.

Recap of OOP

Print function

You have used functions with polymorphic behaviour before. A good example is the print() function. Below we can see how print will work regardless of the data type we provide it.

```
print("Hello")
print(123123)
print(12.342)
print(True)
print([1,2,3,4,5])
```

```
Hello
123123
12.342
True
[1, 2, 3, 4, 5]
```

Interactive Odyssey

- **Background:** Put on your author hat and start telling stories. Your goal is to produce an interactive digital storybook with various characters and plots that will captivate readers.

- **Challenge:** Build a system where characters in the story can interact and have different outcomes by using polymorphism and encapsulation.

- **Objective:**
  - Create character classes with encapsulated interactions and outcomes.
  - Implement polymorphism to allow characters to respond differently to user choices.
  - Develop an interactive storybook that offers readers a personalised and engaging narrative.

Creating the Base Character Class

This is a basic set up for a Base Character class. Here each character will have a name, description and response.

```
class BaseCharacter():

    def __init__(self, name, desc):
        self.name = name
        self.desc = desc
        self.response = ""

    def __str__(self):
        return f"{self.name}\n{self.desc}"
```

Inheriting From Base Class

We can inherit from our BaseCharacter class to create other characters for our story.

```
class Worker(BaseCharacter):

    def __init__(self, name, desc):
        super().__init__(name, desc)
        self.happiness = 0
        self.response = "Sweep all day, sweep all night, sweep is all we do."
        self.retrieve_questions()

    def retrieve_questions(self):
        try:
            with open("""Characters\Worker\Questions\questions.txt""",
                      "r", encoding="utf-8") as questions_file:
                self.questions = [line.replace("\n", "") for line in questions_file]
        except FileNotFoundError:
            print(f"Error creating {self.name}. Questions file not found.")
            self.questions = []
```

Inheriting From Base Class

Here is another class we created by inheriting from our BaseCharacter class.

```
class Manager(BaseCharacter):

    def __init__(self, name, desc):
        super().__init__(name, desc)
        self.happiness = 0
        self.retrieve_questions()
        self.response = "What do you think you are doing in my facility?"


    def retrieve_questions(self):
        try:
            with open("""Characters\Manager\Questions\questions.txt""",
                      "r", encoding="utf-8") as questions_file:
                self.questions = [line.replace("\n", "") for line in questions_file]
        except FileNotFoundError:
            print(f"Error creating {self.name}. Questions file not found.")
            self.questions = []
```

Polymorphic Function

Below is a function that displays polymorphic behaviour as we can use any of our character types we have created as an argument and the functions behaviour would adjust accordingly.

```
def question_character(character):
    print(character)
    draw_line()
    print(character.response)
    draw_line()
    for i, question in enumerate(character.questions, 1):
        print(i, question, sep=": ")
    print(len(character.questions)+1, "Walk Away", sep=": ")
    draw_line()
```

Output Example

```
James
Worker at mystery facility.
-----------------------------------------------------
Sweep all day, sweep all night, sweep is all we do.
-----------------------------------------------------
1: Where am I?
2: Do you know how I got here?
3: Is there a way to get out?
4: Walk Away
-----------------------------------------------------
```

```
Sandra
Manager at mystery facility.
-----------------------------------------------
What do you think you are doing in my facility?
-----------------------------------------------
1: Who are you?
2: What is this place?
3: How did I get here?
4: Walk Away
-----------------------------------------------
```

Output Example

```
James
Worker at mystery facility.
-------------------------------------------------------
Sweep all day, sweep all night, sweep is all we do.
-------------------------------------------------------
1: Where am I?
2: Do you know how I got here?
3: Is there a way to get out?
4: Walk Away
-------------------------------------------------------
1
-------------------------------------------------------
James
Worker at mystery facility.
-------------------------------------------------------
HAHAHA! WHERE ARE YOU?? WHAT DO YOU MEAN?? HAHAHAHA!
-------------------------------------------------------
1: Do you know how I got here?
2: Is there a way to get out?
3: Walk Away
-------------------------------------------------------
```

Interactive Odyssey

Build a system where characters in the story can interact and have different outcomes by using polymorphism and encapsulation.

Important features:

1. **Story:** Think of an interesting story you would like to tell through a series of characters. A good idea can be to have the characters help you solve a mystery.
2. **Multiple Characters:** Allow the user to interact with different characters using polymorphism.
3. **User Actions:** Give the user a set of actions they can take when interacting with characters such as ask, gift, trade etc.
4. **User Experience:** Try to make the experience as interesting as possible for the user.

Advanced Challenge:

- Ask the user to collect gifts and give them to different characters to have them reveal more of the story.

Summary

Interactive Odyssey

- ★ Create an interactive story where the user can interact with different characters to reveal the story.

Polymorphism

- ★ Use polymorphism to provide the use with a set of interactions they can have with each character regardless of their class type.

Which method is called when using the len() function on an object in Python?

A. __size__
B. __length__
C. __len__
D. __count__

What is a static method?

A. A method that cannot be accessed by a class instance
B. A method that can only be accessed by an instance variable
C. A method that belongs to a class rather than an instance of a class
D. A method that does not belong to a class

Questions and Answers

Questions around the Case Study