

CSIT881

Programming and Data Structures

Class & Object (II)



UNIVERSITY
OF WOLLONGONG
AUSTRALIA

Objectives

- Problem solving with Class & Object

Example:

We want to build a program to let children practice mathematics.

- The program generates random questions of 4 types:
 - Addition question: $5 + 7 = ?$
 - Subtraction question: $25 - 3 = ?$
 - Multiplication question: $4 \times 5 = ?$
 - Division question: $20 / 10 = ?$
- The program checks the user answer and indicates the answer is correct or not

```
Welcome to cool math:
```

```
3 + 12 = 15
```

```
Correct
```

```
14 / 2 = 8
```

```
Incorrect
```

```
5 x 8 = 40
```

```
Correct
```

```
10 - 4 = 9
```

```
Good bye!
```

Designing the Class

Class `MathQuestion` has the following attributes:

- First number 4,
- Second number 6,
- Operation +,
- Solution 10

```
class MathQuestion:
    """
    Represents a question such as 4 + 6 = ?
    with the following attributes:
        number1: 1st number
        number2: 2nd number
        operation: +, -, x or /
        solution: the correct solution
    """

    def __init__(self, number1, number2, operation, solution):
        ...
```

Designing the Class

```
class MathQuestion:
```

```
    def __init__(self, number1, number2, operation, solution):  
        self.number1 = number1  
        self.number2 = number2  
        self.operation = operation  
        self.solution = solution
```

```
# create question objects  
question1 = MathQuestion(4, 6, "+", 10)  
  
question2 = MathQuestion(17, 10, "-", 7)  
  
question3 = MathQuestion(5, 3, "x", 15)  
  
question4 = MathQuestion(35, 5, "/", 7)
```

Special (dunder) method

```
class MathQuestion:

    def __str__(self):
        return "{0} {2} {1} = {3}".format(
            self.number1,
            self.number2,
            self.operation,
            self.solution
        )
```

```
question1 = MathQuestion(4, 6, "+", 10)
question2 = MathQuestion(17, 10, "-", 7)
question3 = MathQuestion(5, 3, "x", 15)
question4 = MathQuestion(35, 5, "/", 7)

# testing str
print(str(question1))
print(str(question2))
print(str(question3))
print(str(question4))
```

```
4 + 6 = 10
17 - 10 = 7
5 x 3 = 15
35 / 5 = 7
```

Special (dunder) method

```
class MathQuestion:

    def __repr__(self):
        return "MathQuestion({0}, {1}, '{2}', {3})".format(
            self.number1,
            self.number2,
            self.operation,
            self.solution
        )
```

```
question1 = MathQuestion(4, 6, "+", 10)
question2 = MathQuestion(17, 10, "-", 7)
question3 = MathQuestion(5, 3, "x", 15)
question4 = MathQuestion(35, 5, "/", 7)

# testing repr
print(repr(question1))
print(repr(question2))
print(repr(question3))
print(repr(question4))
```

```
MathQuestion(4, 6, '+', 10)
MathQuestion(17, 10, '-', 7)
MathQuestion(5, 3, 'x', 15)
MathQuestion(35, 5, '/', 7)
```

Instance method

Some information belong to individual object instance.
Some other information is common to all objects.

Instance attribute: data belong to individual object instance.

Class attribute: data that is common to all objects.

Instance method:

- Deal with individual object instance attributes
- Automatically pass the object instance (**self**) as the first parameter

Object instance method

```
class MathQuestion:
    def question_text(self):
        """
        Returns the question text (without the solution)
        """
        return "{0} {2} {1} = ".format(
            self.number1,
            self.number2,
            self.operation
        )
```

```
question1 = MathQuestion(4, 6, "+", 10)
question2 = MathQuestion(17, 10, "-", 7)
question3 = MathQuestion(5, 3, "x", 15)
question4 = MathQuestion(35, 5, "/", 7)

# testing question text
print(question1.question_text())
print(question2.question_text())
print(question3.question_text())
print(question4.question_text())
```

```
4 + 6 =
17 - 10 =
5 x 3 =
35 / 5 =
```

Object instance method

```
class MathQuestion:
    def check_answer(self, answer):
        """
        Returns true if the answer is equal to the solution
        """
        if(answer == self.solution):
            return True

        return False
```

```
question1 = MathQuestion(4, 6, "+", 10)
question2 = MathQuestion(17, 10, "-", 7)
question3 = MathQuestion(5, 3, "x", 15)
question4 = MathQuestion(35, 5, "/", 7)

# testing check answer
print(question1.check_answer(10))
print(question2.check_answer(1))
print(question3.check_answer(15))
print(question4.check_answer(2))
```

```
True
False
True
False
```

Static / Class method

Some information belong to individual object instance.
Some other information is common to all objects.

Instance attribute: data belong to individual object instance.

Class attribute: data that is common to all objects.

Static / Class method:

- Do NOT deal with individual object instance attributes
- Class method: automatically pass the class (**cls**) as the first parameter
- Static method: no automatic parameter passing

Static method

```
class MathQuestion:
    @staticmethod
    def generate_question_add():
        """
        Generate a random addition question
        """
        operation = "+"
        number1 = random.randint(0, 20)
        number2 = random.randint(0, 20)
        solution = number1 + number2

        return MathQuestion(number1, number2, operation, solution)
```

← **import random**

```
# testing generate add question
add_question = MathQuestion.generate_question_add()

print(str(add_question))
print(add_question.question_text())
print(add_question.check_answer(-77))
```

```
12 + 6 = 18
12 + 6 =
False
```

Static method

```
class MathQuestion:
    @staticmethod
    def generate_question_subtract():
        """
        Generate a random subtraction question
        """
        operation = "-"
        solution = random.randint(0, 20)
        number2 = random.randint(0, 20)
        number1 = solution + number2

        return MathQuestion(number1, number2, operation, solution)
```

```
# testing generate subtract question
subtract_question = MathQuestion.generate_question_subtract()

print(str(subtract_question))
print(subtract_question.question_text())
print(subtract_question.check_answer(-77))
```

```
19 - 3 = 16
19 - 3 =
False
```

Static method

```
class MathQuestion:
    @staticmethod
    def generate_question_multiply():
        """
        Generate a random multiplication question
        """
        operation = "x"
        number1 = random.randint(0, 10)
        number2 = random.randint(0, 10)
        solution = number1 * number2

        return MathQuestion(number1, number2, operation, solution)
```

```
# testing generate multiply question
multiply_question = MathQuestion.generate_question_multiply()

print(str(multiply_question))
print(multiply_question.question_text())
print(multiply_question.check_answer(-77))
```

```
2 x 5 = 10
2 x 5 =
False
```

Static method

```
class MathQuestion:
    @staticmethod
    def generate_question_divide():
        """
        Generate a random division question
        """
        operation = "/"
        solution = random.randint(0, 10)
        number2 = random.randint(1, 10)
        number1 = solution * number2

        return MathQuestion(number1, number2, operation, solution)
```

```
# testing generate divide question
divide_question = MathQuestion.generate_question_divide()

print(str(divide_question))
print(divide_question.question_text())
print(divide_question.check_answer(-77))
```

```
14 / 2 = 7
14 / 2 =
False
```

Static method

```
class MathQuestion:
    @staticmethod
    def generate_question():
        """
        Generate a random question
        """
        question_type = random.randint(1, 4)

        if (question_type == 1):
            question = MathQuestion.generate_question_add()
        elif (question_type == 2):
            question = MathQuestion.generate_question_subtract()
        elif (question_type == 3):
            question = MathQuestion.generate_question_multiply()
        else:
            question = MathQuestion.generate_question_divide()

        return question
```


Static method

```
class MathQuestion:
    @staticmethod
    def generate_question():
        """
        Generate a random question
        """
        ...

        return question
```

```
# testing generate random question
random_question1 = MathQuestion.generate_question()
print(str(random_question1))

random_question2 = MathQuestion.generate_question()
print(str(random_question2))
```

```
16 - 7 = 9
4 x 5 = 20
```

Class method

```
class MathQuestion:
    @staticmethod
    def generate_question_add():
        """
        Generate a random addition question
        """
        operation = "+"
        number1 = random.randint(0, 20)
        number2 = random.randint(0, 20)
        solution = number1 + number2

        return MathQuestion(number1, number2, operation, solution)
```

Re-write using class method



```
class MathQuestion:
    @classmethod
    def generate_question_add(cls):
        """
        Generate a random addition question
        """
        operation = "+"
        number1 = random.randint(0, 20)
        number2 = random.randint(0, 20)
        solution = number1 + number2

        return cls(number1, number2, operation, solution)
```

The main program

```
print("Welcome to cool math:")

while True:
    # generate a random question
    question = MathQuestion.generate_question()

    # get the question text and prompt the user to answer the question
    prompt = question.question_text()

    # ask user to enter solution or quit
    user_input = input(prompt)

    # check if student want to quit
    if(user_input == "q"):
        print("Good bye!")
        break

    # user don't want to quit - translate string to integer for answer
    answer = int(user_input)

    # check user answer correct or not
    correct = question.check_answer(answer)

    if(correct):
        print("Correct")
    else:
        print("Incorrect")
```

The main program

Welcome to cool math:

$3 + 12 = \mathbf{15}$

Correct

$14 / 2 = \mathbf{8}$

Incorrect

$5 \times 8 = \mathbf{40}$

Correct

$10 - 4 = \mathbf{q}$

Good bye!