Module 3 Exercise Solutions

February 2, 2023

1 Solutions of the Exercises of Module 3

1. Define a class named “Person” with attributes “name”, “age”, and “gender”. Define a method called “introduction” which returns a string introducing the person. Create an instance of the class and call the “introduction” method.

[ ]:
```
class Person:
    def __init__(self, name, age, gender):
        self.name = name
        self.age = age
        self.gender = gender

    def introduction(self):
        return "Hi, I am {}. I am {} years old and my gender is {}.".
 ↪format(self.name, self.age, self.gender)

person = Person("John", 32, "Male")
print(person.introduction())
```

2. Define a class named “Rectangle” with attributes “length” and “width”. Define methods “area” and “perimeter” which return the area and perimeter of the rectangle respectively. Create an instance of the class and call both methods.

[ ]:
```
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def area(self):
        return self.length * self.width

    def perimeter(self):
        return 2 * (self.length + self.width)

rectangle = Rectangle(5, 10)
print("Area:", rectangle.area())
print("Perimeter:", rectangle.perimeter())
```

3. Define a class named “Student” with attributes “name”, “age”, and “grades”. Define a method

"average_grade" which returns the average of the grades. Create an instance of the class and call the method.

[ ]:
```
class Student:
    def __init__(self, name, age, grades):
        self.name = name
        self.age = age
        self.grades = grades

    def average_grade(self):
        return sum(self.grades) / len(self.grades)

student = Student("Jane", 25, [95, 85, 90])
print("Average grade:", student.average_grade())
```

4. Create a base class called `Animal` with a method called `make_sound()` that prints "Animal making noise". Then create two subclasses `Dog` and `Cat` that both inherit from `Animal`. Override the `make_sound()` method in both subclasses to print "Woof" for `Dog` and "Meow" for `Cat`.

[ ]:
```
class Animal:
    def make_sound(self):
        print("Animal making noise")

class Dog(Animal):
    def make_sound(self):
        print("Woof")

class Cat(Animal):
    def make_sound(self):
        print("Meow")

dog = Dog()
dog.make_sound()  # Output: Woof
cat = Cat()
cat.make_sound()  # Output: Meow
```

5. Create a class `Rectangle` with methods to calculate the area and perimeter. Then create a class `Square` that inherits from `Rectangle`. Override the methods in `Square` to only use one side length as the width and height.

[ ]:
```
class Rectangle:
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def area(self):
        return self.width * self.height
```

```
    def perimeter(self):
        return 2 * (self.width + self.height)

class Square(Rectangle):
    def __init__(self, side):
        super().__init__(side, side)

square = Square(5)
print("Area:", square.area())  # Output: Area: 25
print("Perimeter:", square.perimeter())  # Output: Perimeter: 20
```

6. Create a class `Person` with attributes `name` and `age`. Then create a subclass `Student` that inherits from `Person`. Add an attribute `student_id` to the `Student` class. Override the `__str__` method in both classes to return a custom string representation of each class.

[ ]:
```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def __str__(self):
        return f"Name: {self.name}, Age: {self.age}"

class Student(Person):
    def __init__(self, name, age, student_id):
        super().__init__(name, age)
        self.student_id = student_id

    def __str__(self):
        return f"Name: {self.name}, Age: {self.age}, Student ID: {self.
 ↪student_id}"

person = Person("John Doe", 30)
print(person)  # Output: Name: John Doe, Age: 30
student = Student("Jane Doe", 25, 123456)
print(student)  # Output: Name: Jane Doe, Age: 25, Student ID: 123456
```

7. Write a function to divide two numbers. If the denominator is 0, raise an exception with the message "division by zero error".

[ ]:
```
def divide(a, b):
    try:
        result = a / b
    except ZeroDivisionError:
        print("division by zero error")
        result = None
    return result
```

```
print(divide(10, 2))  # Output: 5.0
print(divide(10, 0))  # Output: division by zero error
```

8. Write a function to calculate the square root of a number. If the number is negative, raise an exception with the message "negative number error".

[ ]:
```
import math

def sqrt(x):
    try:
        result = math.sqrt(x)
    except ValueError:
        print("negative number error")
        result = None
    return result

print(sqrt(16))  # Output: 4.0
print(sqrt(-16))  # Output: negative number error
```

9. Write a function to read a file and count the number of lines in it. If the file does not exist, raise an exception with the message "file not found error".

[ ]:
```
def count_lines(filename):
    try:
        with open(filename) as f:
            lines = f.readlines()
            result = len(lines)
    except FileNotFoundError:
        print("file not found error")
        result = None
    return result

print(count_lines("file.txt"))  # Output: 5
print(count_lines("non_existing_file.txt"))  # Output: file not found error
```