Python

Class 6

Introduction to Python

Azmain Adel

August 28, 2024

01.

Review of Previous Class

Review Topics

- Dictionary
- File Handling
- OOP Basics

02.

Error Handling

Syntax Errors

- A syntax error in Python (or any programming language) is an error that occurs when the code does not follow the syntax rules of the language.

Syntax Errors

```
# Error: Missing colon (:) after the if statement
if True
   print("This will cause a syntax error")
```

```
# Error: The print statement is not correctly indented
def example_function():
print("This will cause a syntax error")
```

```
# Error: The closing parenthesis is missing.
print("This will cause a syntax error"
```

Exceptions

- An **exception** is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions.

Some Types of Exceptions

1. Exception
2. StopIteration
3. SystemExit
4. SyntaxError
5. ArithmeticError
6. OverflowError
7. FloatingPointError
8. ZeroDivisionError
9. AssertionError
10. AttributeError
11. IOError etc

Error Handling

```
try:
   You do your operations here
   ......................
except ExceptionI:
   If there is ExceptionI, then execute this block.
except ExceptionII:
   If there is ExceptionII, then execute this block.
   ......................
else:
   If there is no exception then execute this block.
```

Try-Except Block

```
try:
    # Code that might cause an exception
    risky_code()
except SomeException as e:
    # Code that runs if an exception occurs
    handle_exception(e)
```

Raising Exception

```
raise Exception("This is a general exception")
```

03.

OOP: Class

```python
class Parrot:

    # class attribute
    name = ""
    age = 0

# create parrot1 object
parrot1 = Parrot()
parrot1.name = "Blu"
parrot1.age = 10

# create another object parrot2
parrot2 = Parrot()
parrot2.name = "Woo"
parrot2.age = 15

# access attributes
print(f"{parrot1.name} is {parrot1.age} years old")
print(f"{parrot2.name} is {parrot2.age} years old")
```

04.

OOP: Inheritance

```
# base class
class Animal:

    def eat(self):
        print( "I can eat!")

    def sleep(self):
        print("I can sleep!")

# derived class
class Dog(Animal):

    def bark(self):
        print("I can bark! Woof woof!!")

# Create object of the Dog class
dog1 = Dog()

# Calling members of the base class
dog1.eat()
dog1.sleep()

# Calling member of the derived class
dog1.bark();
```

05.

OOP: Encapsulation

```
class Computer:

    def __init__(self):
        self.__maxprice = 900

    def sell(self):
        print("Selling Price: {}".format(self.__maxprice))

    def setMaxPrice(self, price):
        self.__maxprice = price

c = Computer()
c.sell()

# change the price
c.__maxprice = 1000
c.sell()

# using setter function
c.setMaxPrice(1000)
c.sell()
```

06.

OOP: Polymorphism

```python
class Polygon:
    # method to render a shape
    def render(self):
        print("Rendering Polygon...")

class Square(Polygon):
    # renders Square
    def render(self):
        print("Rendering Square...")

class Circle(Polygon):
    # renders circle
    def render(self):
        print("Rendering Circle...")

# create an object of Square
s1 = Square()
s1.render()

# create an object of Circle
c1 = Circle()
c1.render()
```

07.

Recap and Q&A

Open floor for questions and clarifications

08.

To-do at Home

Problem 1

Write a Python program to create a class representing a Circle. Include methods to calculate its area and perimeter.

Problem 2

Write a Python program to create a class that represents a **shape**. Include methods to calculate its area and perimeter.

Implement subclasses for different shapes like **circle**, **triangle**, and **square**.

Thank you.

azmainadel47@gmail.com
linkedin.com/in/azmainadel
azmainadel.com