



Object Oriented Programming (Python 1) Lab 6



ASST.LEC Fatima Mohammed & Adnan Habeeb

Python Classes and Objects

We can delete the object property by using the del keyword. After deleting it, if we try to access it, we will get an error.

```
class Fruit:
    def __init__(self, name, color):
        self.name = name
        self.color = color
    def show(self):
        print("Fruit is", self.name, "and Color is", self.color)
```

```
obj = Fruit("Apple", "red")
```

← creating object of the class

```
del obj.name
```

← Deleting Object Properties

```
print(obj.name)
```

← Accessing object properties after deleting

Output :

AttributeError: 'Fruit' object has no attribute 'name'

Python Classes and Objects...

Write python program to calculate Perimeter and area of circle use constructor and two method.

```
class Circle_Erea_Perimeter:
    def __init__(self, radius):
        self.radius = radius

    def calculate_perimeter(self):
        perimeter = 2 * 3.14* self.radius
        return perimeter

    def calculate_area(self):
        area = 3.14* self.radius ** 2
        return area
```

Output :

Enter the radius of the circle: 5
The perimeter of the circle is: 31.400000000000002
The area of the circle is: 78.5

```
radius = float(input("Enter the radius of the circle: "))
circle = Circle_Erea_Perimeter(radius)
perimeter = circle.calculate_perimeter()
area = circle.calculate_area()

print(f"The perimeter of the circle is: {perimeter}")
print(f"The area of the circle is: {area}")
```

← Ask the user for input

← Create an instance of the Circle_Erea_Perimeter class with user input as radius

Calculate perimeter and area using the methods of the Circle_Erea_Perimeter class

Display the calculated perimeter and area

Python Classes and Objects...

Create a Python program that counts the number of vowel characters in a user-entered statement using a class called 'VowelCounter'. This class should have two functions: one to retrieve the user's input and another to count the vowels in the entered text. Use an instance of the 'VowelCounter' class to perform the operations.

```
class VowelCounter: ← Class name
    def __init__(self):
        self.statement = ""

    def get_user_input(self):
        self.statement = input("Enter a statement: ")

    def count_vowels(self):
        vowels = 'aeiouAEIOU'
        count = 0

        for char in self.statement:
            if char in vowels:
                count += 1

        return count
```

Use for loop to count vowel characters in statement

```
counter = VowelCounter() ← Creating an instance of the class

counter.get_user_input() ← Get input from the user

vowel_count = counter.count_vowels() ← Count vowels in the statement

print(f"The number of vowels in the statement is: {vowel_count}")
```

Output :

Enter a statement: Ahmed

The number of vowels in the statement is: 2

Python Classes and Objects...

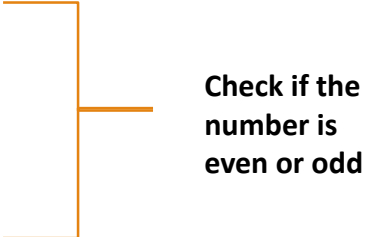
Write python program that use class and method to Separate Odd and Even Numbers in two lists.
use user-Input.

```
class NumberSeparator:
    def __init__(self):
        self.odd_numbers = []
        self.even_numbers = []

    def separate_numbers(self, numbers):
        for num in numbers:
            if num % 2 == 0:
                self.add_even(num)
            else:
                self.add_odd(num)

    def add_odd(self, num):
        self.odd_numbers.append(num)

    def add_even(self, num):
        self.even_numbers.append(num)
```



Check if the number is even or odd

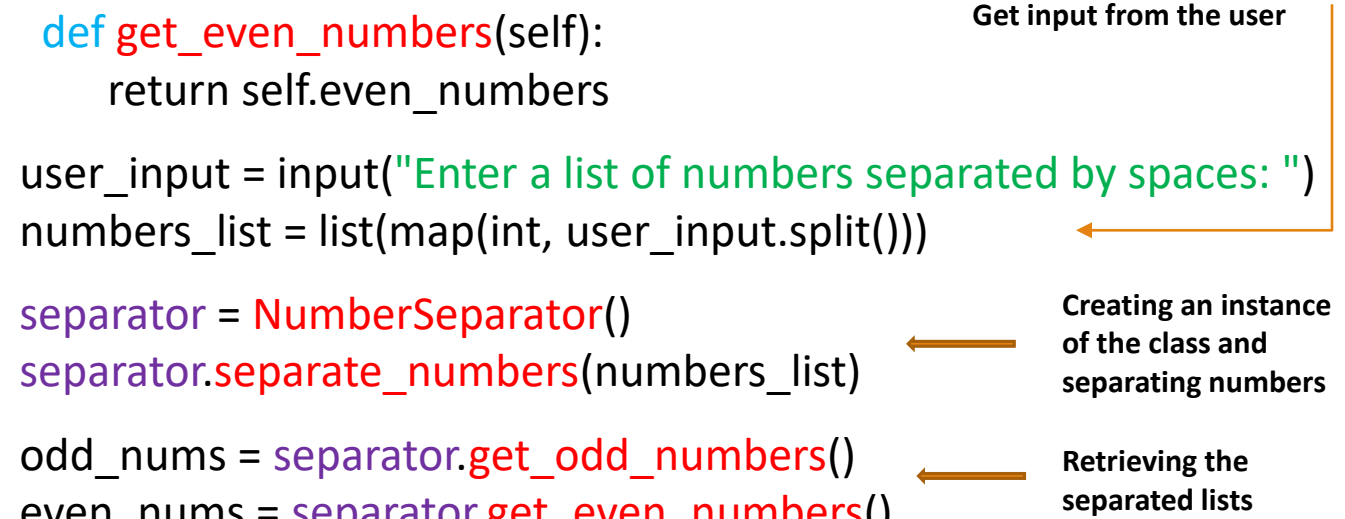
```
def get_odd_numbers(self):
    return self.odd_numbers

def get_even_numbers(self):
    return self.even_numbers
```

```
user_input = input("Enter a list of numbers separated by spaces: ")
numbers_list = list(map(int, user_input.split()))

separator = NumberSeparator()
separator.separate_numbers(numbers_list)

odd_nums = separator.get_odd_numbers()
even_nums = separator.get_even_numbers()
print("Odd numbers:", odd_nums)
print("Even numbers:", even_nums)
```



Get input from the user

Creating an instance of the class and separating numbers

Retrieving the separated lists

Output :

```
Enter a list of numbers separated by spaces: 1 2 3 4 5 6 7 8
Odd numbers: [1, 3, 5, 7]
Even numbers: [2, 4, 6, 8]
```

Python Classes and Objects...

write python class "Fibonacci Sequence" for generating Fibonacci sequences based on user input by using constructor and generate sequence() method.

```
class FibonacciSequence:
    def __init__(self, length):
        self.length = length

    def generate_sequence(self):
        a, b = 1, 1
        sequence = []
        for i in range(self.length):
            sequence.append(a)
            a, b = b, a + b
        return sequence
```

length = int(input("Enter the length of the Fibonacci sequence: "))

Ask the user for input

fib = FibonacciSequence(length)

Create an instance of the FibonacciSequence class with user input as length

result = fib.generate_sequence()
print(result)

Generate and print the Fibonacci sequence based on the provided length

Output :

Enter the length of the Fibonacci sequence: 8
[1, 1, 2, 3, 5, 8, 13, 21]

Python Inheritance

Inheritance allows us to define a class that inherits all the methods and properties from another class.

Parent class is the class being inherited from, also called base class.

Child class is the class that inherits from another class, also called derived class.

```
class Person: ← Create a Parent Class
```

```
    def __init__(self, fname, lname):  
        self.firstname = fname  
        self.lastname = lname
```

```
    def printname(self):  
        print(self.firstname, self.lastname)
```

```
class Student(Person): ← Create a Child Class  
    pass
```

```
x = Student("Ali", "Ahmed")  
x.printname()
```

Output :
Ali Ahmed

Note: Use the pass keyword when you do not want to add any other properties or methods to the class.

Python Inheritance...

So far we have created a child class that inherits the properties and methods from its parent. We want to add the `__init__()` function to the child class (instead of the pass keyword).

`class Person:` ← Create a Parent Class

```
def __init__(self, fname, lname):  
    self.firstname = fname  
    self.lastname = lname
```

```
def printname(self):  
    print(self.firstname, self.lastname)
```

`class Student(Person):` ← Create a Child Class

```
def __init__(self, fname, lname, age):  
    self.firstname = fname  
    self.lastname = lname  
    self.age = age
```

```
x = Student("Ali", "Ahmed", 23)  
x.printname()  
print(x.age)
```

Output :
Ali Ahmed
23

Note: When you add the `__init__()` function, the child class will no longer inherit the parent's `__init__()` function.

Python Inheritance...

To keep the inheritance of the parent's `__init__()` function, add a call to the parent's `__init__()` function:

`class Person:` ← Create a Parent Class

```
def __init__(self, fname, lname):
```

```
    self.firstname = fname
```

```
    self.lastname = lname
```

```
def printname(self):
```

```
    print(self.firstname, self.lastname)
```

`class Student(Person):` ← Create a Child Class

```
def __init__(self, fname, lname):
```

```
    Person.__init__(self, fname, lname) ← Calling parent's __init__() function
```

```
x = Student("Ali", "Ahmed")
```

```
x.printname()
```

Output :

Ali Ahmed

Note: Now we have successfully added the `__init__()` function, and kept the inheritance of the parent class, and we are ready to add functionality in the `__init__()` function.

Use the super() Function

Python also has a super() function that will make the child class inherit all the methods and properties from its parent:

```
class Person: ← Create a Parent Class
```

```
    def __init__(self, fname, lname):
```

```
        self.firstname = fname
```

```
        self.lastname = lname
```

```
    def printname(self):
```

```
        print(self.firstname, self.lastname)
```

```
class Student(Person): ← Create a Child Class
```

```
    def __init__(self, fname, lname):
```

```
        super().__init__(fname, lname) ← using the super() function
```

```
x = Student("Ali", "Ahmed")
```

```
x.printname()
```

Output :

Ali Ahmed

Note: By using the super() function, you do not have to use the name of the parent element, it will automatically inherit the methods and properties from its parent.

Use the super() Function...

In the example below, the year 2019 should be a variable, and passed into the Student class when creating student objects. To do so, add another parameter in the `__init__()` function:

`class Person:` ← Create a Parent Class

```
def __init__(self, fname, lname):
```

```
    self.firstname = fname
```

```
    self.lastname = lname
```

```
def printname(self):
```

```
    print(self.firstname, self.lastname)
```

`class Student(Person):` ← Create a Child Class

```
def __init__(self, fname, lname, year):
```

```
    super().__init__(fname, lname) ← using the super() function
```

```
    self.graduationyear=year ← Add a year parameter, and pass the correct  
                                year when creating objects
```

```
x = Student("Ali", "Ahmed" , 2023)
```

```
x.printname()
```

```
print(x.graduationyear)
```

Output :

Ali Ahmed

2023

Use the super() Function...

In the example below, the year 2019 should be a variable, and passed into the Student class when creating student objects. To do so, add another parameter in the `__init__()` function:

`class Person:` ← Create a Parent Class

```
def __init__(self, fname, lname):  
    self.firstname = fname  
    self.lastname = lname
```

```
def printname(self):  
    print(self.firstname, self.lastname)
```

`class Student(Person):` ← Create a Child Class

```
def __init__(self, fname, lname, year):  
    super().__init__(fname, lname) ← using the super() function  
    self.graduationyear=year ← Add a year parameter, and pass the correct year when creating objects
```

```
def welcome(self): ← Add a method called welcome to the Student class  
    print("Welcome", self.firstname, self.lastname, "to the class of", self.graduationyear)
```

```
x = Student("Ali", "Ahmed", 2023)
```

```
x.printname()
```

```
print(x.graduationyear)
```

```
x.welcome()
```

Output :

Ali Ahmed

2023

Welcome Ali Ahmed to the class of 2023

Thank you