Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Програмування

Лабораторна робота №6

«Основи об’єктно-орієнтованого програмування. Модулі та пакети»

Виконала:
Студентка групи ІО-15
Кушнерик.Є.О.
Залікова книжка №1508

Перевірив:
Пономаренко А. М.

Київ – 2021

Лабораторна робота №6

Тема: «Основи об’єктно-орієнтованого програмування. Модулі та пакети».

Мета: вивчити способи створення та підключення модулів та пакетів. Основи ООП. Методи і атрибути класів та робота з ними. Побудова програми у стилі ООП.

Завдання:

1. Вивчити матеріал лекцій 18, 19, 20 та 21.
2. Виконати індивідуальне завдання лабораторної роботи, вибране відповідно до варіанту.

Теоретичні основи:

Об'єктно-орієнтоване програмування

Об'єктно-орієнтоване програмування (ООП) – це спосіб організації програми, що дозволяє використовувати той самий код багаторазово. На відміну від функцій і модулів, ООП дозволяє не тільки розділити програму на фрагменти, але й описати предмети реального світу у вигляді зручних сутностей – об'єктів, а також організувати зв'язки між цими об'єктами.

Основною «цеглинкою» ООП є клас. *Клас* – це складний тип даних, що включає набір змінних і функцій для керування значеннями, що зберігаються в цих змінних.

Змінні називають *атрибутами,* а функції – *методами.* Клас є фабрикою об'єктів, тобто дозволяє створити необмежену кількість екземплярів, заснованих на цьому класі.

ООП ґрунтується на трьох основних концепціях розробки: інкапсуляція, спадкування й поліморфізм.

Визначення класу й створення екземпляра класу

Клас описують за допомогою ключового слова `class` за наступною схемою:

```
class <Назва класу>[(<Клас1>[, ... ,
<Класn>])]: ["""  Рядок документування """]
<Опис атрибутів і методів>
```

Інструкція створює новий об'єкт і присвоює посилання на нього ідентифікатору, зазначеному після ключового слова `class`.

Це означає, що назва класу повинна повністю відповідати правилам іменування змінних. Після назви класу в круглих дужках можна вказати один або кілька базових класів через кому. Якщо ж клас не успадковує базові класи, то круглі дужки можна не вказувати.

Всі вирази всередині інструкції `class` виконуються при створенні класу, а не його екземпляра.

Створення атрибута класу аналогічно створенню звичайної змінної.

Метод всередині класу створюється так само, як і звичайна функція, за допомогою інструкції `def`.

Методам класу в першому параметрі, який обов'язково слід указати явно, автоматично передають посилання на екземпляр класу. Загальноприйнято цей параметр називати ім'ям `self` (не обов'язково).

Доступ до атрибутів і методів класу всередині обумовленого методу проводиться через змінну `self` за допомогою точкової нотації – до атрибута `x` з методу класу можна звернутися так: `self.x`.

Щоб використовувати атрибути й методи класу, необхідно створити екземпляр класу згідно з наступним синтаксисом:

```
<Екземпляр класу> = <Назва класу>((<Параметри>])
```

При доступі до методів класу використовують такий формат:

```
<Екземпляр класу>.<Ім'я методу>((<Параметри>])
```

При виклику методу не потрібно передавати посилання на екземпляр класу як параметр, як у визначенні методу всередині класу. Посилання на екземпляр класу інтерпретатор передає автоматично.

Доступ до атрибутів класу здійснюється аналогічно:

```
<Екземпляр класу>.<Ім'я атрибута>
```

Методи `__init__()` і `__del__()`

При створенні екземпляра класу інтерпретатор автоматично викликає метод ініціалізації `__init__()`. В інших мовах програмування такий метод прийнято називати конструктором класу. Формат методу:

```
def__init__(self[, <Значення1>[, ... ,
<Значеннян>]]): <Інструкції>
```

За допомогою методу `__init__ ()` можна присвоїти початкові значення атрибутам класу. При створенні екземпляра

класу параметри цього методу вказують після імені класу в круглих дужках:

```
<Екземпляр класу> = <Ім'я класу>([<Значення1>[, ... , <Значеннян>]])
```

Якщо конструктор викликають при створенні об'єкта, то перед знищенням об'єкта автоматично викликається метод, називаний деструктором. У мові `Python` деструктор реалізується у вигляді визначеного методу `__del__()`. Метод не буде викликаний, якщо на екземпляр класу існує хоча б одне посилання.

Індивідуальні завдання

Відповідно до **номера у списку** вибрати індивідуальне завдання. Написати програму. Забезпечити ввід даних з клавіатури комп'ютера та друк результатів. При виводі використовувати форматування.

8	Створити клас, який описує планету Сонячної системи. Цей клас повинен містити назву планети, її порядковий номер, рахуючи від Сонця, відстань від Сонця, об'єм та масу, орбітальну швидкість. Методи класу мають визначати густину планети, відношення маси Землі до маси даної планети, відношення орбітальної швидкості Землі до о.ш. планети. Створити об'єкти даного класу для планет Сонячної системи та ввести дані, користуючись даними з Вікіпедії. Програма повинна обчислювати та виводити за запитом густину планети, відношення маси та о.ш. Землі до маси та о.ш. планети, введеної за запитом. Програма повинна також виводити за запитом список планет у порядку зростання їх густини з зазначенням номеру планети та її орбітальної швидкості.

Код програми

```
class Planet:
    def __init__(self, name, num, distance, volume, speed, mass):
        self.name = name
        self.num= num
        self.volume = volume
        self.mass = mass
        self.speed = speed
        self.distance = distance

    def density(self):
        return self.mass/self.volume

    def ratiomass(self):
        return zem.mass/self.mass

    def ratiospeed(self):
        return zem.speed/self.speed

mer = Planet("Меркурій", 1, 46001200, 6.083e10, 47.87, 3.285e23)
ven = Planet("Венера", 2, 107476259, 9.38e11, 35.02, 4.867e24)
zem = Planet("Земля", 3, 147098290, 10.8321e11, 29.783, 5.972e24)
mar = Planet("Марс", 4, 2.06655e8, 1.6318e11, 24.13, 6.39e23)
yup = Planet("Юпітер", 5, 7.405736e8, 1.43128e15, 13.07, 1.898e27)
sat = Planet("Сатурн", 6, 1353572956, 8.2713e14, 9.69, 5.683e26)
ura = Planet("Уран", 7, 2748938461, 6.833e13, 6.81, 8.681e25)
nep = Planet("Нептун", 8, 4452940833, 6.254e13, 5.4349, 1.024e26)
planets = [mer, ven, zem, mar, yup, sat, ura, nep]

x = input("=> Введіть назву планети, щоб дізнатись її особливості\n"
          "=> Введіть 1, щоб побачити список планет у порядку зростання
              їх густини\n")
if(x == "1"):
    new_list = sorted(planets, key=lambda planet: planet.density())
    print("(км/с) (№) (Назва)")
    for i in new_list:
        print("{0:.3} - {1} - {2}".format(i.speed, i.num, i.name))
else:
    for i in planets:
        if x == i.name:
            y = int(input("=> Введіть 2, щоб дізнатись густину
                              планети\n"
                          "=> Введіть 3, щоб дізнатись відношення маси
                              Землі до маси даної планети\n"
                          "=> Введіть 4, щоб дізнатись відношення
                              орбітальної швидкості Землі до о.ш.
                              планети\n"))
            if (y==2):
                print("{} (км^3/кг)".format(i.density()))
            elif (y==3):
                print("{} (кг)".format(i.ratiomass()))
            elif (y==4):
                print("{} (км/с)".format(i.ratiospeed()))
```

Результат програми

```
=> Введіть назву планети, щоб дізнатись її особливості
=> Введіть 1, щоб побачити список планет у порядку зростання їх густини
1
(км/с) (№) (Назва)
9.69 - 6 - Сатурн
6.81 - 7 - Уран
13.1 - 5 - Юпітер
5.43 - 8 - Нептун
24.1 - 4 - Марс
35.0 - 2 - Венера
47.9 - 1 - Меркурій
29.8 - 3 - Земля
```

```
=> Введіть назву планети, щоб дізнатись її особливості
=> Введіть 1, щоб побачити список планет у порядку зростання їх густини
Меркурій
=> Введіть 2, щоб дізнатись густину планети
=> Введіть 3, щоб дізнатись відношення маси Землі до маси даної планети
=> Введіть 4, щоб дізнатись відношення орбітальної швидкості Землі до о.ш. планети
3
18.179604261796044 (кг)
```

```
=> Введіть назву планети, щоб дізнатись її особливості
=> Введіть 1, щоб побачити список планет у порядку зростання їх густини
Юпітер
=> Введіть 2, щоб дізнатись густину планети
=> Введіть 3, щоб дізнатись відношення маси Землі до маси даної планети
=> Введіть 4, щоб дізнатись відношення орбітальної швидкості Землі до о.ш. планети
2
1326085741434.2407 (км^3/кг)
```

```
=> Введіть назву планети, щоб дізнатись її особливості
=> Введіть 1, щоб побачити список планет у порядку зростання їх густини
Земля
=> Введіть 2, щоб дізнатись густину планети
=> Введіть 3, щоб дізнатись відношення маси Землі до маси даної планети
=> Введіть 4, щоб дізнатись відношення орбітальної швидкості Землі до о.ш. планети
4
1.0 (км/с)
```

Висновки:

Я вивчила способи створення та підключення модулів та пакетів, основи ООП, методи і атрибути класів та те, як з ними працювати. Здобула базові навички побудови програми у стилі ООП.