Основы и методология программирования

Семинар 12

Классы. Наследование. Исключения.

Переопределение методов в классах

__str__(self) - вызывается функциями str, print и format. Возвращает строковое представление объекта.

__repr__(self) - вызывается для отображения состояния объекта в консоли

```
class A:
    def __init__(self):
        self.b = 0

    def __str__(self):
        return str(self.b)

    def __repr__(self):
        return "repr: " + str(self.b)


print(A())   # печать объекта класса __str__(self)

a = A()
a   # печать "внутреннего представления", задаваемого функцией __repr__(self)
```

Переопределение методов в классах

__str__(self) - вызывается функциями str, print и format. Возвращает строковое представление объекта.

__lt__(self, other) - x < y вызывает x.__lt__(y)

__le__(self, other), **__eq__(self, other)**, **__ne__(self, other), __gt__(self, other), __ge__(self, other) –** аналогично

__hash__(self) - получение хэш-суммы объекта, например, при добавления в словарь

__bool__(self) - вызывается при проверке истинности. Если этот метод не определен, вызывается метод __len__ (объекты, имеющие ненулевую длину, считаются истинными)

__getitem__(self, key) , **__setitem__(self, key, value)** - получение/назначение элемента по индексу.

__delitem__(self, key) - удаление элемента по индексу.

__iter__(self) - возвращает итератор для контейнера.

__reversed__(self) - итератор из элементов, следующих в обратном порядке.

__contains__(self, item) - проверка на принадлежность элемента контейнеру (item in self).

Перегрузка арифметических операторов

__add__(self, other) - сложение. x + y вызывает x.__add__(y).

__sub__(self, other), **__mul__(self, other)** - аналогично

__truediv__(self, other) - деление (x / y).

__floordiv__(self, other) - целочисленное деление (x // y).

__mod__(self, other) - остаток от деления (x % y).

__pow__(self, other) - возведение в степень (x ** y, pow(x, y)).

__lshift__(self, other), **__rshift__(self, other)** - битовый сдвиг влево (x << y) и вправо (x >> y).

__and__(self, other) - битовое И (x & y) и др.

__iadd__(self, other) - сокращенное сложение x+=y, и др.

__neg__(self) - унарный -.

__round__(self[, n]) - округление.

Задача 1. Пример реализации вектора

```
import math

class Vector2D:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __str__(self):
        return '({}, {})'.format(self.x, self.y)

    def __add__(self, other):
        return Vector2D(self.x + other.x, self.y + other.y)

    def __iadd__(self, other):
        self.x += other.x
        self.y += other.y
        return self
```

Задача 1. Пример реализации вектора

```
def __sub__(self, other):
    return Vector2D(self.x - other.x, self.y - other.y)

def __isub__(self, other):
    self.x -= other.x
    self.y -= other.y
    return self

def __abs__(self):
    return math.hypot(self.x, self.y) # sqrt(x*x + y*y)

def __bool__(self):
    return self.x != 0 or self.y != 0

def __neg__(self):
    return Vector2D(-self.x, -self.y)
```

Наследование

```
class Tree():
    def __init__(self, kind, height):
        self.__kind = kind
        self.__age = 0
        self.__height = height

    def info(self):
        print("Tree" + str((self.__age, self.__kind, self.__height)))

    def grow(self):
        self.__age += 1
        self.__height += 0.5

    def getKind(self):
        return self.__kind

class FruitTree(Tree):  # FruitTree - наследник класса Tree
    def __init__(self, kind, height, group="unknown"):
        super().__init__(kind, height)
        self.__group = group

    def giveFruits(self):
        print("Collected 10kg of %ss" % self.getKind()) # нельзя __kind, т.к. он private!
```

Наследование

```
tree1 = Tree("oak", 2)
tree2 = FruitTree("apple", 0.7)

tree2.info()  # Есть доступ к методам родителя
tree2.grow()
tree2.info()
tree2.giveFruits()
# А для родительского экземпляра метод give_fruits() недоступен
# tree1.give_fruits() # Вызовет ошибку
```

Исключения

```
"ttt" + 5

100 / 0

int("hello")
```

```
try:
    x = 100 / 0
except ZeroDivisionError:
    print("Errror: division by 0")
    x = 0

try:
    x = 100 / 0
except ArithmeticError: # FloatingPointError,
# OverflowError, ZeroDivisionError.
    print("Errror: division by 0")
    x = 0
```

Иерархия исключений

BaseException

- **…**
- **Exception**
  - **StopIteration**
  - …
  - **ArithmeticError**
    - **FloatingPointError**
    - **OverflowError**
    - **ZeroDivisionError**
  - …

Исключения

```
file = open('1.txt')
ints = []
try:
    for line in file:
        ints.append(int(line))
except ValueError:
    print('Это не число!')
except Exception:
    print('Что это?')
else:
    print('Сейчас все хорошо')
finally:
    file.close()
    print('Файл закрыт.')
```