Полиморфизм

Свойство кода работать с разными типами данных называют полиморфизмом.

Мы уже неоднократно пользовались этим свойством многих функций и операторов, не задумываясь о нем. Например, оператор + является полиморфным:

```
print(1 + 2)          # 3
print(1.5 + 0.2)      # 1.7
print("abc" + "def")  # abcdef
```

```
def f(x, y):
    return x + y


print(f(1, 2))          # 3
print(f(1.5, 0.2))      # 1.7
print(f("abc", "def"))  # abcdef
```

```python
class Book:
    def __init__(self, name, author):
        self.name = name
        self.author = author

    def get_name(self):
        return self.name

    def get_author(self):
        return self.author


book = Book('Война и мир', 'Толстой Л. Н.')
print('{}, {}'.format(book.get_name(), book.get_author()))
# Война и мир, Толстой Л. Н.
```

```python
from math import pi


class Circle:
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return pi * self.radius ** 2

    def perimeter(self):
        return 2 * pi * self.radius
```

```python
class Square:
    def __init__(self, side):
        self.side = side

    def area(self):
        return self.side * self.side

    def perimeter(self):
        return 4 * self.side
```

```python
def print_shape_info(shape):
    print("Area = {}, perimeter = {}.".format(
        shape.area(), shape.perimeter()))


square = Square(10)
print_shape_info(square)
# Area = 100, perimeter = 40.

circle = Circle(10)
print_shape_info(circle)
# Area = 314.1592653589793, perimeter = 62.83185307179586.
```

```python
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)


rect = Rectangle(10, 15)
print_shape_info(rect)  # Area = 150, perimeter = 50.
```

```python
for person in people:
    if isinstance(person, Student):
        print(person.university)
    elif isinstance(person, Employee):
        print(person.company)
    else:
        print(person.name)
    print()
```

ЗАДАЧ
И

Выборки

```
class Selector:
    def __init__(self, lst):
        self.evens = []
        self.odds = []
        for i in lst:
            if i % 2 == 0:
                self.evens.append(i)
            else:
                self.odds.append(i)

    def get_odds(self):
        return self.odds

    def get_evens(self):
        return self.evens
```

Форматы дат

```
class EuropeanDate:
    def __init__(self, year, month, day):
        self.year = year
        self.month = month
        self.day = day

    def set_year(self, year):
        self.year = year

    def set_month(self, month):
        self.month = month

    def set_day(self, day):
        self.day = day

    def get_year(self):
        return self.year

    def get_month(self):
        return self.month

    def get_day(self):
        return self.day

    def format(self):
        return '{:02}.{:02}.{:04}'.format(self.day, self.month, self.year)
```

```
class AmericanDate:
    def __init__(self, year, month, day):
        self.year = year
        self.month = month
        self.day = day

    def set_year(self, year):
        self.year = year

    def set_month(self, month):
        self.month = month

    def set_day(self, day):
        self.day = day

    def get_year(self):
        return self.year

    def get_month(self):
        return self.month

    def get_day(self):
        return self.day

    def format(self):
        return '{:02}.{:02}.{:04}'.format(self.month, self.day, self.year)
```

Вывод предложений

```
class LeftParagraph:
    def __init__(self, text_width):
        self.text_width = text_width
        self.first_word = True
        self.words = []
        self.lines = []
        self.current_width = 0

    def add_word(self, word):
        addition = 1 if (self.current_width > 0) else 0
        if self.current_width + addition + len(word) <= self.text_width:
            self.words.append(word)
            self.current_width += len(word) + addition
        else:
            self.lines.append(' '.join(self.words))
            self.words = [word]
            self.current_width = len(word)

    def end(self):
        print(*self.lines, sep='\n')
        print(' '.join(self.words))
        self.words = []
        self.lines = []
        self.current_width = 0
```

```
class RightParagraph:
    def __init__(self, text_width):
        self.text_width = text_width
        self.first_word = True
        self.words = []
        self.current_width = 0
        self.lines = []

    def add_word(self, word):
        addition = 1 if (self.current_width > 0) else 0
        if self.current_width + addition + len(word) <= self.text_width:
            self.words.append(word)
            self.current_width += len(word) + addition
        else:
            line = ' '.join(self.words)
            space_prefix = ' ' * (self.text_width - len(line))
            self.lines.append(space_prefix + line)
            self.words = [word]
            self.current_width = len(word)

    def end(self):
        line = ' '.join(self.words)
        space_prefix = ' ' * (self.text_width - len(line))
        print(*self.lines, sep='\n')
        print(space_prefix + line)
        self.words = []
        self.lines = []
        self.current_width = 0
```