notes P 1.20

Creating computer programs in Python

★★☆☆

Beginner level

Materials prepared by the department
of methodological development department

Classes

A class is a custom data type that describes some object (game character, person, animal, machine, etc.).

To create a class, you need to write the keyword class and the name of the class.
The name has to start with a capital letter:

```
class Person:
```

A class can describe both the object’s properties (name, height and age of the person, type of animal, brand of car) and its abilities (what the object can do - talk, run, fight).

Variables created within a class are called properties and functions are called methods. Each class has a special method __init__ - this is the constructor of the class, it is automatically called when the object is created and is needed to specify its original properties.

The object is an instance of the class; to create it, you have to create a variable and after the equal sign write the name of the class and parentheses:

```
student = Person()
```

To make the properties created within one method available within the entire class, you must associate them with the object of the class using the word self. The name of the property is spelled with a dot - self.name. By the way, self is an obligatory argument of any method. But you need to define it only in method declaration, when you call it self is passed automatically.

```
def __init__(self, name):
self.name = name
```

To call a method of an object or access a property, you must write the name of the object and the name of the method or property separated by a dot:

```
student.say_hello() # call the say_hello method of the student object
```

Inheritance allows you not to write the same properties and methods to classes with similar behavior, but to inherit them from a common parent. The parent class is specified when creating an inherited class in parentheses after its name:

```
class Dog(Animal):
```

It is important to keep in mind that by writing a method with the same name as the parent class in the descendant class, we will override it, i.e., the new method will replace the parent's method. To preserve the behavior of the parent method, we need to call it in the method of the descendant class using the super() command:

```
def __init__(self):
super().__init__()
```

```python
class Car:
  speed = 100
```

Creating a class.

```python
porshe = Car()
```

Creating an instance (object) of a class.

```python
super()
```

Creating an Oval.

```python
__init__()
```

Creating a rectangle.

```python
self
```

Makes the created property available in all methods of the class.