Создание своего модуля на Python
Теперь пришло время создать свой модуль. Создадим файл mymodule.py, в которой определим какие-нибудь функции:
(для создания модуля рекомендую использовать Spider. Hf удобном для вас диске выбелите католог под проект и создайте в нем файл с указанным названием).
Как назвать модуль?
Помните, что вы (или другие люди) будут его импортировать и использовать в качестве переменной. Модуль нельзя именовать также, как и ключевое слово (их список можно посмотреть тут). Также имена модулей нельзя начинать с цифры. И не стоит называть модуль также, как какую-либо из встроенных функций. То есть, конечно, можно, но это создаст большие неудобства при его последующем использовании.
Куда поместить модуль?
Туда, где его потом можно будет найти. Пути поиска модулей указаны в переменной sys.path. В него включены текущая директория (то есть модуль можно оставить в папке с основной программой), а также директории, в которых установлен python. Кроме того, переменную sys.path можно изменять вручную, что позволяет положить модуль в любое удобное для вас место (главное, не забыть в главной программе модифицировать sys.path).
Можно ли использовать модуль как самостоятельную программу?
Можно. Однако надо помнить, что при импортировании модуля его код выполняется полностью, то есть, если программа что-то печатает, то при её импортировании это будет напечатано. Этого можно избежать, если проверять, запущен ли скрипт как программа, или импортирован. Это можно сделать с помощью переменной **__name__**, которая определена в любой программе, и равна "**__main__**", если скрипт запущен в качестве главной программы, и имя, если он импортирован. Например, mymodule.py может выглядеть вот так:

```
def hello():
    print('Hello, world!')

def fib(n):
    a = b = 1
    for i in range(n - 2):
        a, b = b, a + b
    return b

if __name__ == "__main__":
    hello()
    for i in range(10):
        print(fib(i))
```

```
def hello():
    print('Hello, world!')

def fib(n):
    a = b = 1
    for i in range(n - 2):
        a, b = b, a + b
    return b
```

Теперь в этой же папке создадим другой файл, например, main.py:

In [ ]:

```
```

```
import mymodule

mymodule.hello()
print(mymodule.fib(10))
```

```
Результат работы программы
Hello, world!
55
```

Рассмотрим возможность использования модулей для классов. Создадим файл classed.py со следующим кодом:

```
class Person1:
    name = "Дмитрий"
    last = "Донской"
    year_bith = 1367
    year_dead = 1389
     def display_info(self):
        print("Привет, меня зовут ", self.name, self.last)
    def age_info(self):
        print("Я прожил", self.year_dead - self.year_bith, "года")
```

и файл test.py

```
import classed
person1 = Person1()
person1.display_info()
person1.age_info()
person2 = Person1()
person2.name = "Петр"
person2.last = "I"
person2.year_bith = 1672
person2.year_dead = 1725
person2.display_info()
person2.age_info()
```

```
Результат выполнения программы:
Привет, меня зовут  Дмитрий Донской
Я прожил 22 года
Привет, меня зовут  Петр I
Я прожил 53 года
Привет, меня зовут  Дмитрий Донской
Я прожил 22 года
Привет, меня зовут  Петр I
Я прожил 53 года
```

Модернизируем код программы, вынесем некоторые методы класса в отдельный файл с именем metodas.py

```
    def display_info(self):
        print("Привет, меня зовут ", self.name, self.last)
    def age_info(self):
        print("Я прожил", self.year_dead - self.year_bith, "года"
```

в коде classed.py подключим данный модуль

```
import metodas
class Person1:
    name = "Дмитрий"
    last = "Донской"
    year_bith = 1367
    year_dead = 1389
```

Файл test.py оставим без изменения

```
import classed
```

```
person1 = Person1()
person1.display_info()
person1.age_info()
person2 = Person1()
person2.name = "Петр"
person2.last = "I"
person2.year_bith = 1672
person2.year_dead = 1725
person2.display_info()
person2.age_info()

Результат выполнения программы не поменялся:
Привет, меня зовут  Дмитрий Донской
Я прожил 22 года
Привет, меня зовут  Петр I
Я прожил 53 года
Привет, меня зовут  Дмитрий Донской
Я прожил 22 года
Привет, меня зовут  Петр I
Я прожил 53 года
```

```
Рассмотрим еще один пример:
    создадим файл test2.py

def info(self):
        print("Rectangle")
        print("Color: " + self.color)
        print("Width: " + str(self.width))
        print("Height: " + str(self.height))
        print("Area: " + str(self.area()))
def area(self):
        return self.__width * self.__height
создадим файл toper.py

import test2
class Figure:
    def __init__(self, color):
        self.__color = color
    @property
    def color(self):
        return self.__color
    @color.setter
    def color(self, c):
        self.__color = c
    def info(self):
       print("Figure")
       print("Color: " + self.__color)

class Rectangle(Figure):
    def __init__(self, width, height, color):
        super().__init__(color)
        self.__width = width
        self.__height = height
    @property
    def width(self):
        return self.__width
    @width.setter
    def width(self, w):
        if w > 0:
            self.__width = w
        else:
```

```
            raise ValueError
    @property
    def height(self):
        return self.__height
    @height.setter
    def height(self, h):
        if h > 0:
            self.__height = h
        else:
            raise ValueError
создадим файл main.py
import toper
fig = Figure("orange")
fig.info()
rect = Rectangle(10, 20, "green")
rect.info()

Результат выполнения программы:
Figure
Color: orange
Figure
Color: green
```

Задание: в соответствии с заданиями и выполнить программную реализацию с разбиением кода программ на модули, требование к разбиению: код модулей не должен превышать 25 строк