AMERICAN JOURNALS
Open Access, Peer Reviewed Journals

American Journal of Technology and Applied Sciences
ISSN (E): 2832-1766 Volume 39, August - 2025

USING USER MODULES IN THE PYTHON PROGRAMMING LANGUAGE

Badriddinova Gulnoza Mehriddinovna
Bukhara State of University, Faculty of Physics
Mathematics and Information Technologies, Student of the Group 1-1PM-23.

ABSTRACT	KEYWORDS
This article talks about the use of user modules in applications, their advantages and ease of this method in programming.	Python, programming, module, user module, standard module, preference and convenience.

Introduction

Today, the Python programming language is gaining popularity for its simple and understandable syntax, ease of the user in building a program. This programming language is leaving everyone dumb with the compactness of the program using its standard libraries and modules, and at the same time the ease with which you read code. The concept of a module is a collection of codes written for a special task, in which the asoson is a file with an extension ".py", in which functions, classes and even variables are stored. Standard libraries primarily collect modules that are needed by all users. But if a certain function needs to be used for two different projects, it is darcor that the programmer maintains this function as a module so that he does not have to rewrite this code part. Another advantage of user modules lies in the fact that each project code is regulated and reduced complexity.

M. Jahongir Sherzodovich, in his article "Advantages of the Python programming language", cited his opinion that python has a large standard library, which includes areas of various types, that the most desirable programs for all users are written in it, and that the written codes in these modules significantly reduce the length of the main program [1].

G. Van Rossum, in his book "Python programming language", mentioned that there is a specific name space and global variables for each module. If the variable in the main program is identical to the name of the global variable in the module, the user has also stated that he does not worry about it, that they will not be confused and separate [2].

Suzi Roman Avrievich, in her book "Python programming language", describes the python programming language as having a large library of moddules and packages, thus "attached batteries". He suggested that it was more useful to assemble functions and classes with many connections into a single user module than to write that function over and over again [3].

Based on the analysis of the scientific work of many scientists, we can conclude that it is effective to create a program using modules.

Methods

Let's take a look at the user module and the module-free program in practical applications.
Consider the following issue: an arbitrary integer n is entered. Let the row of Fibonacci numbers up to that number be projected onto the screen and the nearest Fibonacci number to the n number that is entered be found.
Let's do this task without modules.

```
def fibonaci(n):
    k=[0]
    a=0; b=1
    while b<=n:
        a,b=b,a+b
        k.append(b)
    print(k)
    if abs(k[-2]-n)>abs(k[-1]-n):
        return k[-1]
    else:
        return k[-2]
```

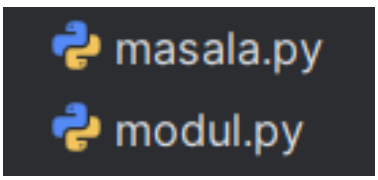
If we want to work with this issue using a user module, we first create a file with the .py extension under an arbitrary name (for example, module.py) and place the functional part in it.

```
class Fibonachi:
    def __init__(self, n):
        self.n=n
        self.k=[0,1]

    def foy_mod1(self):
        a,b=0,1
        while b<=self.n:
            a,b=b,a+b
            self.k.append(b)
        return self.k

    def foy_mod2(self):
        if abs(self.k[-2]-self.n)>abs(self.k[-1]-self.n):
            return self.k[-1]
        else:
            return self.k[-2]
```

Then we place another file with the .py extension (for example, masala.py) in the same directory as our module.py file.



We continue the program by importing our module into the newly opened Python file.

```
import modul
n=int(input("Son kiriting: "))
f=modul.Fibonachi(n)
print("Fibonachi ketma-ketligi:", f.foy_mod1())
print(f"{n} ga eng yaqin Fibonachi soni:", f.foy_mod2())
```

As you can see, this method is much simpler and shorter. The user can also apply this module in other applications.

Result

If we analyze the above programs, then we have a program that we will run masala.py (Figure 3), which screens Fibonacci numbers up to that number when an arbitrary number is entered, and the Fibonacci number close to that number.

```
Son kiriting: 14
Fibonachi ketma-ketligi: [0, 1, 1, 2, 3, 5, 8, 13, 21]
14 ga eng yaqin Fibonachi soni: 13

Process finished with exit code 0
```

Here, when the number 14 is entered, the previous number 14 is subtracting the sequence, depending on which Fibonacci number falls into the range. Then the Fibonacci closest to the 14th issue is subtracting the number

```
Son kiriting: 83
Fibonachi ketma-ketligi: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
83 ga eng yaqin Fibonachi soni: 89

Process finished with exit code 0
```

Here, the sequence when the number 83 is entered, and then the Fibonacci closest to it, are subtracting the number 89.

Discussion

The example given above is a program to output a series of Fibonacci numbers up to the number n included, and this is a Fibonacci number close to the number n, worked in two different ways. The first method is simple, modular, worked through a function. The second method used a user module. This method is much more convenient than the previous one, works without duplicate code parts. The programmer will be able to keep this Fibonacci class in a separate midule and apply it in their next, other projects. This is much more convenient for the programmer. That is, the main topic, logic was preserved as a module.

In these projects, it should be remembered that the compiled module and the project should stand in the same directory. Standard modules are loaded with python, while the user module is for the convenience of its own work of this programmer. The moment the programmer submits his project, if the module and the project are not in the same place, an error may go in the program. The programmer is reluctant to pay attention to this.

The programmer achieves much more convenience when using modules or functions, procedures in his projects. Does not use the same program code over and over again. And when working with a module, only the function that it needs, calls the methods, does not need to know their internal operation. The use of modules, functions in applications makes the program much better quality and easier to read. The use of ready-made modules, their methods in projects eliminates the complexity of the program. Because the complexity in its interiors is not important for the project, it simply increases the length of the program, complicates its reading and leads to the fact that it spends a lot of time even for the programmer.

Conclusion

In conclusion, I can say that the use of ready-made modules in projects helps daturists save time on rewriting codes. And one of the advantages is that it can save the written code as a module, while in its later projects it can be used as a user module. The use of modules in applications makes the application understandable and easy to read. If a project is built using modular functions, it saves a lot of time in writing the program and greatly improves the quality of the code, making complex parts easier to understand.

References

1. Mamarajabov J. Sh. Advantages of the Python programming language //INTERNATIONAL CONFERENCE ON INTERDISCIPLINARY SCIENCE - 2024
2. Россум Г., Дрейк Ф. Л. Д., Откидач Д. С. Python programming language //М.: Вильяме. – 2001.
3. Сузи Р. А. Python programming language //М.:-2005.
4. Kamalova N. I. PEDAGOGLARGA DASTURLASH TILINI O'QITISHDA INTERFAOL USULLARDAN FOYDALANISH //Academic research in educational sciences. – 2021. – Т. 2. – №. 12. – С. 1271-1275.
5. Камалова Н. И. и др. МЕТОДИКА ОБУЧЕНИЯ ОПЕРАТОРУ IF В ЯЗЫКЕ ПРОГРАММИРОВАНИЯ PYTHON //International Conference on Economics, Finance, Banking and Management. – 2025. – С. 86-91.