Лабораторная работа №5-3*.

Основные требования:

- каждая функция должна иметь docstring множ-й комментарий (по типу что делает данная функция)
- написание функций должно быть компактным (не в 100 строчек кода)
- именования функций должны быть нормальными и четко отражать смысл самой функции
- код должен соответствовать стандарту языка Python (PEP).
- классы и объекты должны соответствовать стандартам языка программирования Python.

Общее задание к Л/Р:

Работа с (*argv, **kwargs).

Основная цель работы получить надпись на экране "Access Granted!" путем изменения передаваемых значений argv или kwargs. (возможно потребуется и то и то использовать)

Что нужно сделать?

В лабе 5-3 уже было что-то подобное теперь код стал лишь сложнее. Задача остается прежней нужно подобрать параметры для __call__access__() так чтобы выдало сообщение в консоли "Access Granted!"

Итак **дан код** (Caller.py класс):

ВНИМАНИЕ!!! КОД МЕНЯТЬ СТРОГО ЗАПРЕЩЕНО (КРОМЕ УКАЗАНИЯ ПУТИ К ФАЙЛ .txt)

```
class Caller:
   def __init__(self, base_address: any) -> None:
        pass
```

Блок схема (красное (нужное) - если гах != rbx, зеленая если гах = rbx):

cmp_z для сравнения регистров **rax** и **rbx** используется.

Далее второй блок (__next__block__): красное (нужное) если rax <= rbx и зеленое если rax > rbx

```
public function __call__access__ (call_object, *args, **kwargs)
                       stack = []
                       rax = args[0] * 2
                       r13, r14, r15 = 0, 1, 0
                       stack.append(r13)
                       stack.append(r14)
                       stack.append(r15)
                       rbx = args[len(args) - 1] % 100 + 1
                       stack[0], stack[1] = rax, rax + rbx * 8
                       stack[2] = ctn - stack[0] / stack[1]
                       rbx = 0
                       del stack[0], stack[1]
                       r12 = int(call_object.value)
                       stack.append(r12)
                       rcx = args[kwargs["arg"]] - 1000 * stack[1] + stack[0]
                       rcx -= -1
                       rbx = rcx
                       rcx = 0
                       return call_object.cmp_z(call_object, rax, rbx)
                                                                                              public function __next__block__ (rax, rbx)
                                                                               rdx = rax * 5 + 1
                                                                               rdi, rsi = -1, 0
                                                                               rbx -= rdi + rsi - rdx
                                                                               return call_object.cmp_j(rax, rbx)
return "Access Denied!"
                                                                                return "Access Granted!"
exit(-1)
                                                                                exit(-1)
```

Основной код (решения) будет находиться например в файле (main.py):

from Caller import Caller

```
call_object = Caller(...)
out = call_object.__call__access__(call_object, a1,
a2,..)
print(out)
```