Программирование. Язык Python.
Введение.
Лабораторная работа № 1

Комплект 1: Установка среды программирования и разработки

Python и менеджер пакетов PIP:

```
Microsoft Windows [Version 10.0.22631.4169]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\stepa>pip help

Usage:
  pip <command> [options]

Commands:
  install                     Install packages.
  download                    Download packages.
  uninstall                   Uninstall packages.
  freeze                      Output installed packages in requirements format.
  inspect                     Inspect the python environment.
  list                        List installed packages.
  show                        Show information about installed packages.
  check                       Verify installed packages have compatible dependencies.
  config                      Manage local and global configuration.
  search                      Search PyPI for packages.
  cache                       Inspect and manage pip's wheel cache.
  index                       Inspect information available from package indexes.
  wheel                       Build wheels from your requirements.
  hash                        Compute hashes of package archives.
  completion                  A helper command used for command completion.
  debug                       Show information useful for debugging.
  help                        Show help for commands.

General Options:
  -h, --help                  Show help.
  --debug                     Let unhandled exceptions propagate outside the main subroutine, instead of logging them
                              to stderr.
  --isolated                  Run pip in an isolated mode, ignoring environment variables and user configuration.
  --require-virtualenv        Allow pip to only run in a virtual environment; exit with an error otherwise.
  --python <python>           Run pip with the specified Python interpreter.
  -v, --verbose               Give more output. Option is additive, and can be used up to 3 times.
  -V, --version               Show version and exit.
  -q, --quiet                 Give less output. Option is additive, and can be used up to 3 times (corresponding to
                              WARNING, ERROR, and CRITICAL logging levels).
  --log <path>                Path to a verbose appending log.
  --no-input                  Disable prompting for input.
  --keyring-provider <keyring_provider>
```

Аккаунт на Replit: https://replit.com/@stepanna

Комплект 3: Задачи для самостоятельной работы.

Задача 3.1: Создайте простую программу калькулятор, которая позволяет из функции main() ввести два числа и тип арифметической операции, а потом вычисляет результат. Свой код опубликуйте на KttSs reSlit. cRm и предоставьте ссылку в ответах на лабораторную работу в Moodle в документе-отчёте. Реализацию арифметических действий и вычисление результата с его возвратом сделайте в отдельной функции calculate(...). Протестируйте свой калькулятор с помощью вызова нескольких своих простых функций test_∗() с ключевым словом assert внутри. Обязательно напишите хорошую документацию к своему коду.

Реализация: https://replit.com/@stepanna/Lab-1#T1.py

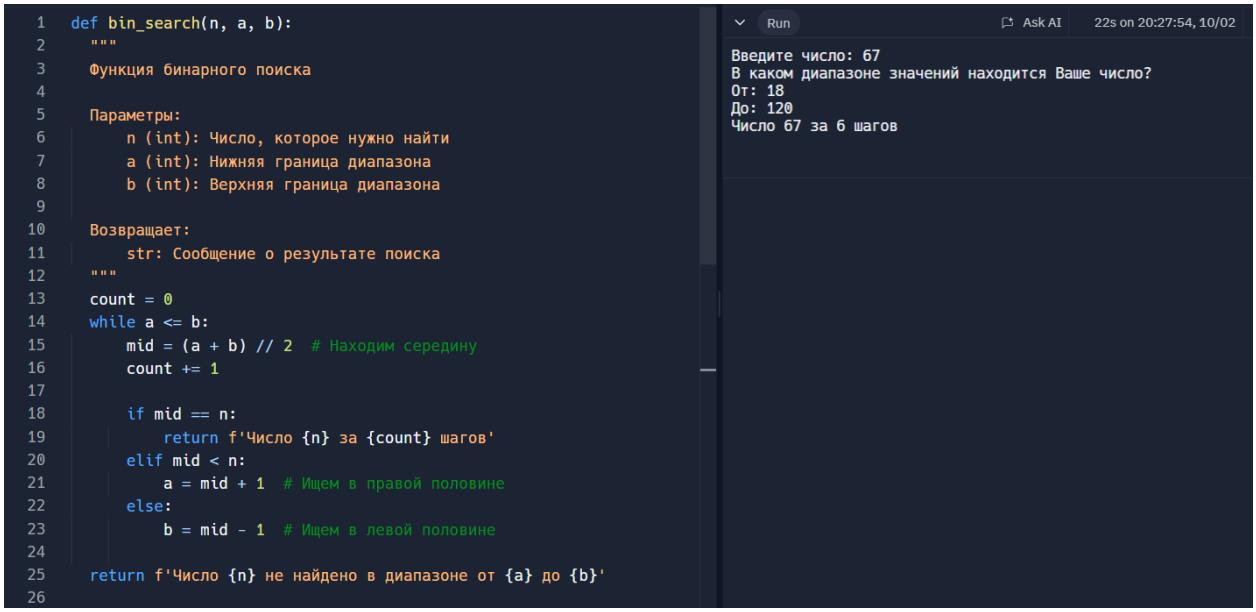
```python
def calculate(a, b, operand):

    """
    Функция вычисления

     Параметры:
        a (int): Первое число
        b (int): Второе число
        operand (str): Операция над числами

    Возвращает:
        int: Результат вычиислений
        или str: сообщение об ошибке при делении на ноль

    """

    if operand == '+':
      return a + b
    if operand == '-':
      return a - b
    if operand == '*':
      return a * b
    elif operand == '/':
      if b == 0:
        return 'Ошибка: на ноль делить нельзя'
      else:
        return a / b
```

```python
def test_calculator():

    """
    Функция для тестирования функции calculate

    Проверяет корректность выполнения операций сложения, вычитания,
    умножения и деления, а также обработку ошибок

    """
    assert calculate(5, 3, '+') == 8
    assert calculate(8, 3, '-') == 5
    assert calculate(7, 6, '*') == 42
    assert calculate(3, 3, '/') == 1

test_calculator()
```

Задача 3.2: Реализуйте программно классическую простую игру "угадай число" (guess number) с помощью алгоритма медленного перебора (инкремента) по одному числа, либо с помощью алгоритма бинарного поиска. Алгоритм принимает на вход само число, которое он должен угадать, интервал значений в котором оно загадано и в цикле делает угадывания тем или иным выбранным вами способом. После угадывания из функции алгоритма возвращается угаданное число и число угадываний/сравнений, которые

пришлось проделать. Обязательно напишите хорошую документацию к своему коду

Реализация: https://replit.com/@stepanna/Lab-1#T2.py



```python
def bin_search(n, a, b):
    """
    Функция бинарного поиска

    Параметры:
        n (int): Число, которое нужно найти
        a (int): Нижняя граница диапазона
        b (int): Верхняя граница диапазона

    Возвращает:
        str: Сообщение о результате поиска
    """
    count = 0
    while a <= b:
        mid = (a + b) // 2  # Находим середину
        count += 1

        if mid == n:
            return f'Число {n} за {count} шагов'
        elif mid < n:
            a = mid + 1  # Ищем в правой половине
        else:
            b = mid - 1  # Ищем в левой половине

    return f'Число {n} не найдено в диапазоне от {a} до {b}'
```

```
Run    Ask AI    22s on 20:27:54, 10/02
Введите число: 67
В каком диапазоне значений находится Ваше число?
От: 18
До: 120
Число 67 за 6 шагов
```

```python
# Основная часть программы
num = int(input('Введите число: '))
print('В каком диапазоне значений находится Ваше число?')
lim_1 = int(input('От: '))
lim_2 = int(input('До: '))

# Проверка, что лимиты корректные
if lim_1 > lim_2:
    print('Неверный диапазон. Нижняя граница должна быть меньше верхней')
else:
    result = bin_search(num, lim_1, lim_2)
    print(result)
```