

**Министр науки и высшего образования Российской
Федерации**

**Федеральное государственное автономное образовательное
учреждение высшего образования**

«Национальный исследовательский университет ИТМО»

Факультет информационных технологий и программирования

Лабораторная работа №2

Использование функций IDE

Выполнил студент группы № М3102

Харлуниин Александр Александрович

Санкт-Петербург
2021

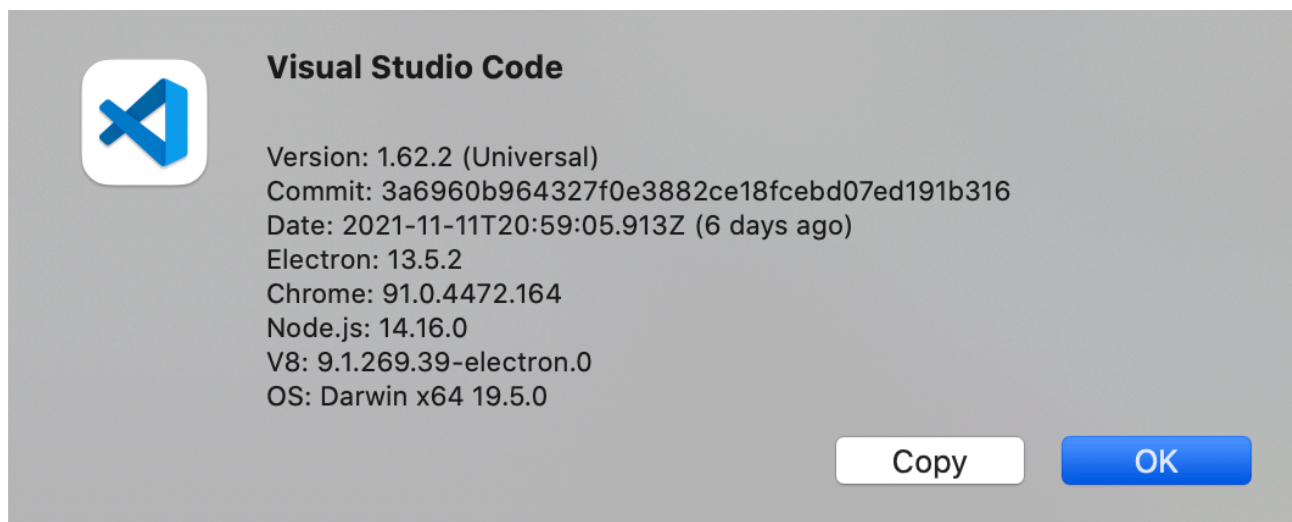
Текст задания

1. Скачать и установить *Visual Studio Code* (можно другую IDE / главное произвести настройку всех нижеописанных пунктов)
2. Сконфигуровать для запуска и отладки кода (любой из 3 языков *java*, *python*, *c++*).
3. Написать простой проект (чтение файла, простой консольный калькулятор)
4. Написать тесты к проекту.
5. Изучить *Visual Studio Code CLI*:
описать не менее 10 команд используемых в данном интерфейсе.
6. Написать 5 собственных Снимков
7. Изучить и описать наиболее распространённые *HotKeys*
8. Задokumentировать код с помощью расширения
<<https://marketplace.visualstudio.com/items?itemName=cschlosser.doxdocgen>> (или аналоги)
9. Установить шрифт *FiraCode* (или другой)
10. Установить <<https://marketplace.visualstudio.com/items?itemName=MS-vsliveshare.vsliveshare>> и
показать чего получилось) (любое другое расширение или
встроенные возможности ide для шеринга вашего кода и
совместной работы)

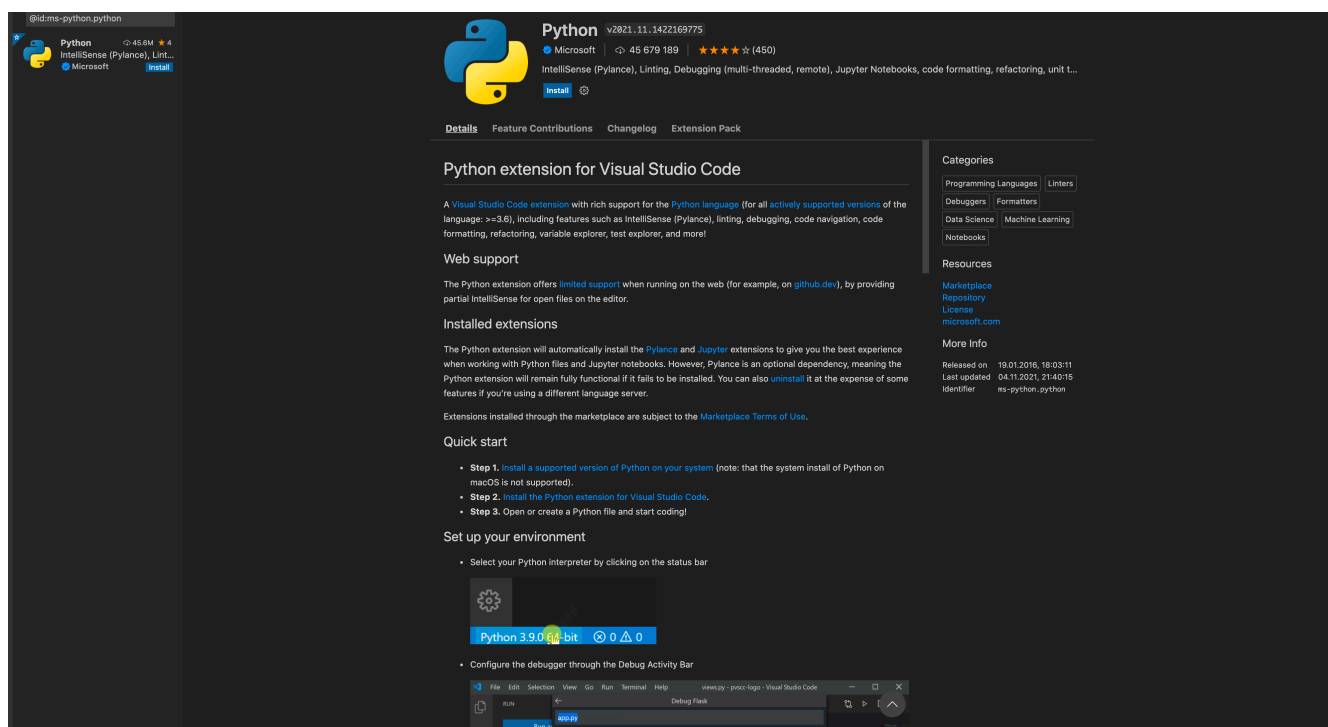
Решение с комментариями

I. Установка IDE и конфигурация под Python 3

Для выполнения работы использовалась данная версия ПО:



Для работы с Python было установлено расширение от Microsoft



II. Создание простого проекта и его тестирование

Простой консольный калькулятор

```
def simple_calculator(string):
    try:
        return eval(string, {}, {})
    except:
        return 'ERR'

def main():
    equation = input()
    while equation:
        print(simple_calculator(equation))
        equation = input()

if __name__ == '__main__':
    main()
```

Калькулятор поддерживает следующие функции:

- Основные арифметические операции, в т.ч. DIV и MOD
- Работа с «длинными» целыми и рациональными числами
- Элементы булевой алгебры (1 and 0)
- Поддержка встроенной функции sum()
- Поддержка условий (напр, $2 + 2 == 4$)

III. Набор юнит-тестов

```
import calc
print('File imported!')
assert calc.simple_calculator("2+2") == 4, "correct addition: should be 4"
assert calc.simple_calculator("2-2") == 0, "correct subtraction: should be 0"
assert calc.simple_calculator("2/2") == 1, "correct division: should be 1"
assert calc.simple_calculator("2*2") == 4, "correct multiplication: should be 4"
assert calc.simple_calculator("8/2") == 4, "correct DIV: should be 4"
assert calc.simple_calculator("2%2") == 0, "correct MOD: should be 0"
assert calc.simple_calculator("2+2*2") == 6, "support of operations priorities"
assert calc.simple_calculator("(2+2)*2") == 8, "support of brackets priorities"
assert calc.simple_calculator("10-15") == -5, "should support negative numbers"
assert calc.simple_calculator("0.1 + 0.2") - 0.3 < 1/(10**8), "bad accuracy for float"
assert calc.simple_calculator("2+2 == 4") == True, "should support boolean"
assert calc.simple_calculator("0**0") == 1, "tricky exponentiation!"
assert calc.simple_calculator("import os") == 'ERR', "should be safe"
assert calc.simple_calculator("1/0") == 'ERR', "mustn't do unsupported operations"
assert calc.simple_calculator("1%0") == 'ERR', "mustn't do unsupported operations"
assert calc.simple_calculator("1//0") == 'ERR', "mustn't do unsupported operations"
assert calc.simple_calculator("temp = 10") == 'ERR', "mustn't support declaration"
assert calc.simple_calculator("equation = '2+2'") == 'ERR', "mustn't change local variables"
assert calc.simple_calculator("2**1024") == 2**1024, "should support long numbers"
assert calc.simple_calculator("sum([2,3])") == 5, "should support built-in functions"
assert calc.simple_calculator("0 or 0") == 0, "should support boolean algebra: answer is 0"
assert calc.simple_calculator("1 and 0") == 0, "should support boolean algebra: answer is 0"
assert calc.simple_calculator("1 and 0 or 0") == 0, \
"should support boolean algebra (priorities): answer is 0"
print('all tests passed!')
```

IV. Visual Studio Code CLI

Установка

>shell command:

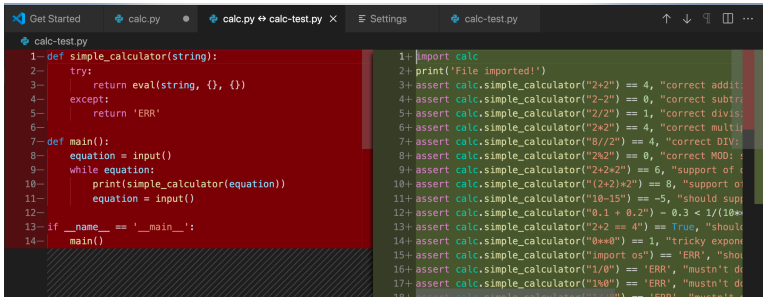
Shell Command: Install 'code' command in PATH



Для первичной установки необходимо выполнить следующие действия:

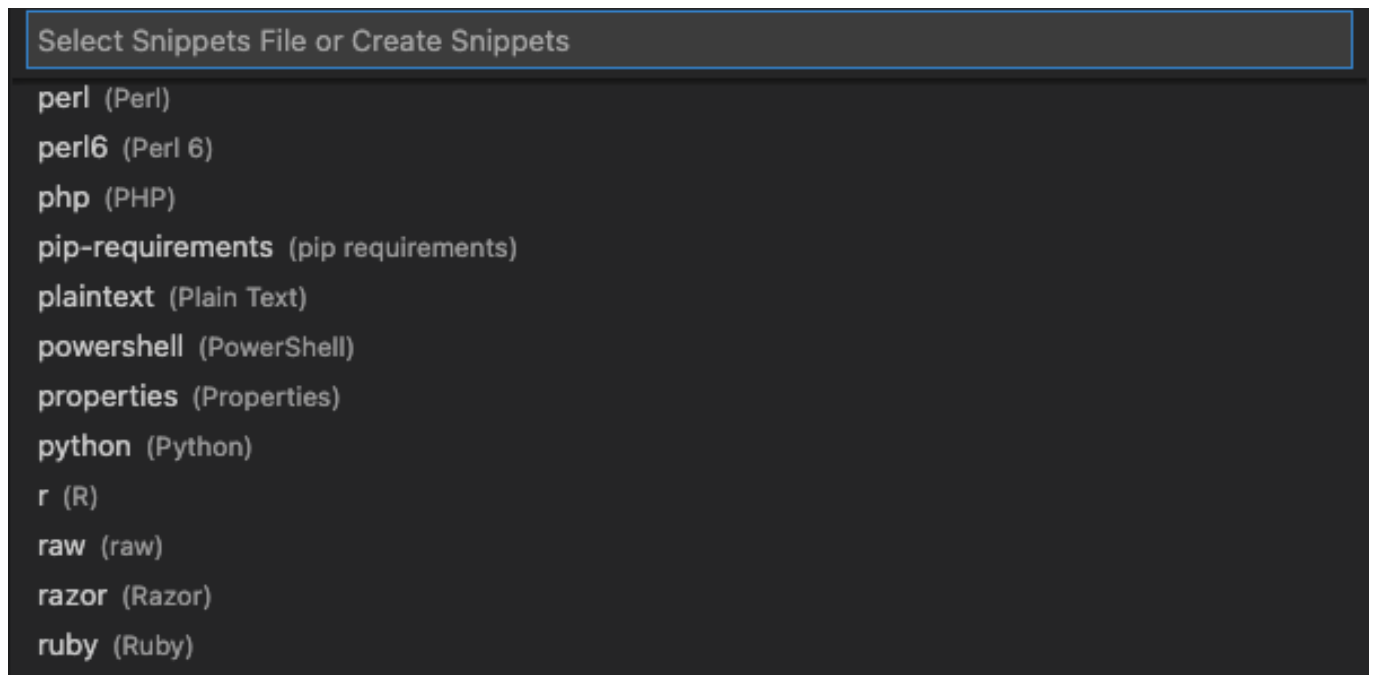
- Открыть меню команд (cmd + shift + P)
- Выбрать Shell Command: Install 'code' command in PATH

Обзор команд

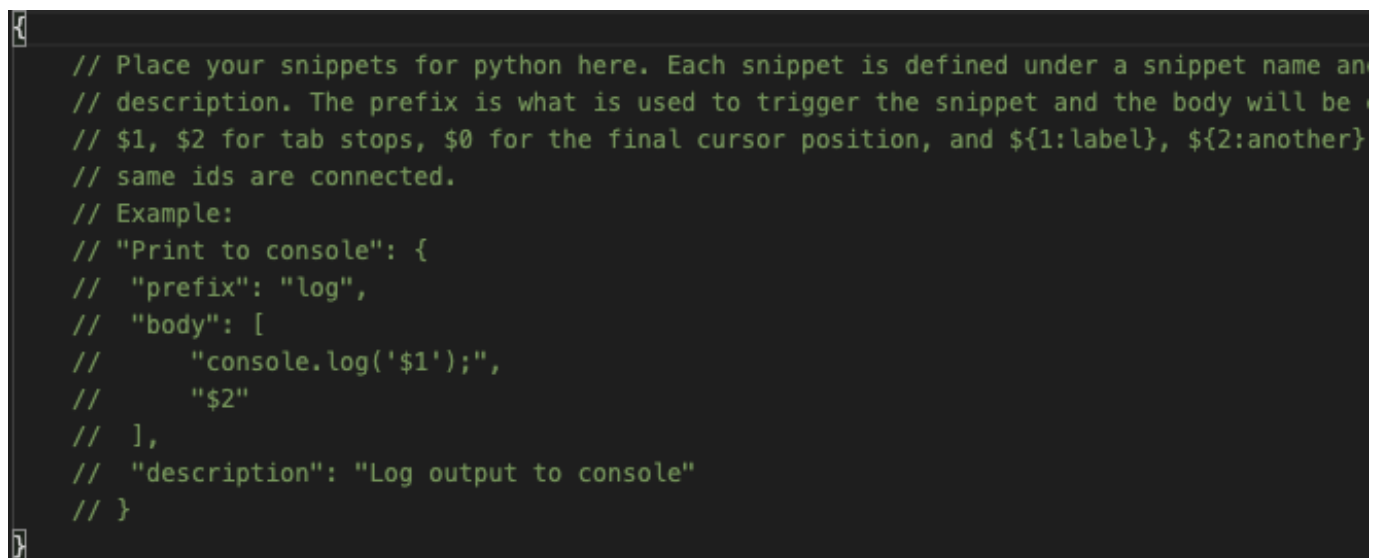
Команда	Комментарии
<code>code --help</code>	Помощь. Показать свои команды
<code>code --version</code>	Напечатать версию в консоль
<code>code --diff <file1> <file2></code>	Сравнить 2 файла 
<code>code --goto <file:line[:character]></code>	Открыть файл на конкретной строке и позиции в строке (опционально)
<code>code --new-window</code>	Открыть новое окно
<code>code --list-extensions</code>	Показать установленные расширения
<code>code --list-extensions --show-versions</code>	Показать установленные расширения и их версии
<code>code --list-extensions --category <category></code>	Показать установленные расширения и отсортировать их по категориям
<code>code --status</code>	Вывести диагностическую информацию и текущие процессы
<code>code --disable-extensions</code>	Отключить все расширения

V. Снимпеты

Чтобы создать свой снимпет (привязанные к языку) необходимо войти в Code > Preferences > User snippets и выбрать свой язык



Далее необходимо ввести свой снимпет в .json файл



У снимпета есть следующие атрибуты:

- prefix - набор символов, при котором он вызывается (hot symbols)
- body - массив и строк снимпета, снимет выводится построчно с \n
- description - описание снимпета

Сниппет	Описание
<pre>"open file": { "prefix": "op", "body": ["open('\$1', '\$0')"], "description": "open file" },</pre>	Открыть файл, указатели \$1 и \$0 указывают на положение курсора после ввода сниппета
<pre>"for range": { "prefix": "frange", "body": ["for \$1 in range(\$2): \n\t\$t0"], "description": "for in a particular range" },</pre>	Список for в определенном диапазоне
<pre>"easy_number lambda": { "prefix": "easy()", "body": ["easy_num = lambda x: not len([i for i in range(2, round(x ** 0.5) + 1) if x % i == 0])"], "description": «easy number boolean lambda function" },</pre>	Создание лямбда-функции, определяющую число на простоту
<pre>"Stack class": { "prefix": "class Stack", "body": ["class Stack:", "\tdef __init__(self):\n\t\tself.data = []\n", "\tdef add(self, x):\n\t\tself.data.append(x)\n", "\tdef take(self):\n\t\tif self.data:\n\t\t\treturn self.data.pop()"], "description": "Stack class" },</pre>	Создать класс стека с методами add и take
<pre>"Queue class": { "prefix": "class Queue", "body": ["class Queue:", "\tdef __init__(self):\n\t\tself.stack_in = []\n\t\tself.stack_out = []\n", "\tdef add(self, x): \n\t\tself.stack_in.append(x)\n", "\tdef take(self):\n\t\tif self.stack_out: \n\t\t\treturn self.stack_out.pop()\n\t\telif self.stack_in:", "\t\t\twhile self.stack_in: \n\t\t\t\tself.stack_out.append(self.stack_in.pop()), "\t\t\tself.take()"], "description": "Queue class using two stacks" },</pre>	Создать класс очереди (на двух стеках) с методами add и take

VI. Горячие клавиши

Клавиши	Описание
Cmd N	Новый файл
Cmd O	Открыть файл
Cmd W	Закрыть файл
Cmd S	Сохранить
F5	Запустить с дебаггингом
^F5	Запустить без дебаггинга
Cmd A	Выбрать все
Cmd F	Найти
Cmd shift F	Найти и заменить

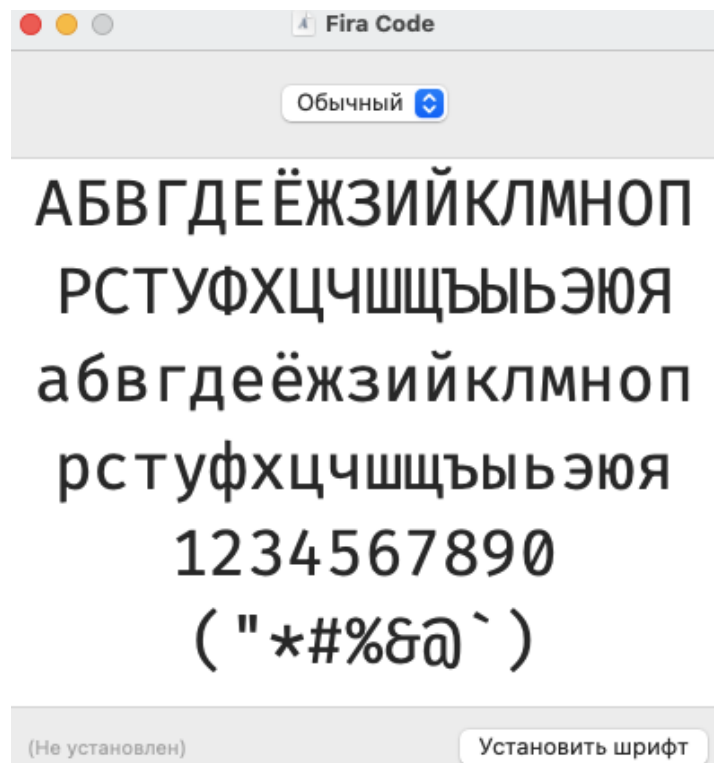
VII. Doxygen

Расширение позволяет автоматически комментировать код

```
/**
 * @brief
 *
 * @param multiplic
 * @param uint
 * @return uint1024_t
 */
uint1024_t multiply_by_uint(uint1024_t *multip, unsigned int u) {
    uint1024_t result = from_uint(0);
    bool leading_zeroes = true;
    char i;
    for (i=RAZR_LEN_OF_UINT1024-1; i>=0; i--) {
        if (multip->arr[i] != 0) {
            leading_zeroes = false;
            break;
        }
    }
}

/**
 * @brief
 *
 * @param x
 */
void printf_value(uint1024_t *x) {
    char i;
    bool leading_zeroes = true;
    for (i = RAZR_LEN_OF_UINT1024 - 1; i >= 0; i--) {
        if ((leading_zeroes) && (x->arr[i] != 0)) {
            leading_zeroes = false;
            printf("%u", x->arr[i]);
            continue;
        }
        if (!leading_zeroes) {
            printf("%09u", x->arr[i]);
        }
    }
    if (leading_zeroes) {
        printf("0");
    }
    putchar('\n'); //Переход на новую строку!!
}
```

VIII. Шрифт Fira Code



```
"editor.fontFamily": "Fira Code",  
"editor.fontLigatures": true,
```

IX. Расширение Live Share

Расширение позволяет поделиться кодом в реальном времени и пригласить других людей для редактирования

