

УНИВЕРСИТЕТ ИТМО

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ  
НАПРАВЛЕНИЕ ПОДГОТОВКИ «ПРОГРАММНАЯ ИНЖЕНЕРИЯ»

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3**  
**курса «Информатика»**

**Вариант 335050**

Выполнил студент:

Берестовский Святослав Сергеевич  
группа: Р3111

Преподаватель:

Малышева Татьяна Алексеевна

Санкт-Петербург, 2021 г.

# Содержание

<b>3. Регулярные выражения</b>	<b>2</b>
3.1. Описание заданий	2
3.1.1. Задание "Смайлики"	2
3.1.2. Доп. задание №1	2
3.1.3. Доп. задание №2	3
3.2. Задание "Смайлики"	3
3.2.1. Исходный код	3
3.2.2. Пример вывода	4
3.2.3. Тесты	5
3.3. Доп. задание №1	5
3.3.1. Исходный код	5
3.3.2. Пример вывода	6
3.3.3. Тесты	6
3.4. Доп. задание №2	6
3.4.1. Исходный код	6
3.4.2. Пример вывода	7
3.4.3. Тесты	7
3.4.4. Дополнение	7
3.5. Выводы	8
<b>Литература</b>	<b>9</b>

## Лабораторная работа 3

# Регулярные выражения

### 3.1. Описание заданий

#### 3.1.1. Задание "Смайлики"

1. Реализуйте программный продукт на языке Python, используя регулярные выражения.
2. Для своей программы придумайте минимум 5 тестов. Каждый тест является отдельной сущностью, передаваемой регулярному выражению для обработки. Для каждого теста необходимо самостоятельно (без использования регулярных выражений) найти правильный ответ. После чего сравнить ответ, выданный программой, и полученный самостоятельно
3. Программа должна считать количество смайликов вида :-{0 в предложенном тексте.
4. Нарисовав смайлик при помощи средств языка программирования Python, можно заработать дополнительные баллы.

#### 3.1.2. Доп. задание №1

1. Реализуйте программный продукт на языке Python, используя регулярные выражения.
2. Для своей программы придумайте минимум 5 тестов. Каждый тест является отдельной сущностью, передаваемой регулярному выражению для обработки. Для каждого теста необходимо самостоятельно найти правильный ответ. После чего сравнить ответ, выданный программой, и полученный самостоятельно.

№ ИСУ % 6	Задание
4	Анатолий выложил пост с расписанием доп. занятий по информатике, но везде перепутал время. Поэтому нужно заменить все вхождения времени на строку (TBD). Время – это строка вида HH:MM

### 3.1.3. Доп. задание №2

1. Реализуйте программный продукт на языке Python, используя регулярные выражения.
2. Для своей программы придумайте минимум 5 тестов.
3. Протестируйте свою программу на этих тестах.

№ ИСУ % 4	Задание
2	Студент Вася очень любит курс «Компьютерная безопасность». Однажды Васе задали домашнее задание зашифровать данные, переданные в сообщение. Недолго думая, Вася решил заменить все целые числа на функцию от этого числа. Функцию он придумал не сложную $3x^2 + 5$ , где $x$ - исходное число. Помогите Васе с его домашним заданием.

## 3.2. Задание "Смайлики"

### 3.2.1. Исходный код

Исходный код скрипта доступен также на моем GitHub: <https://github.com/anim3boy/itmo-labs/blob/main/computer-science/lab-3/task-1/script.py>.

Листинг 3.1: Исходный код

```

1 from re import findall, escape
2
3 EYES = [":", ";", "X", "8", "="]
4 NOSES = ["_", "<", "-{", "<{"]
5 MOUTHS = ["(", ") ", "O", "|", "\\ ", "/", "P"]
6
7 log = lambda x: print(decorator(x))
8 inp = lambda x: input("[?] %s:\n > " % x)
9
10 def decorator(text):
11     if type(text) is tuple:
12         if text[0] == "ok":
13             return "[" + text[1]
14         if text[0] == "err":
15             return "-" + text[1]
16     else:
17         return "!" + text
18
19 def generate_pattern(isu=335050):
20     ISU = isu
21     eye = EYES[ISU % 5]
22     nose = NOSES[ISU % 4]
23     mouth = MOUTHS[ISU % 7]
24     pattern = eye + nose + mouth
25     return ("ok", pattern)
26
27 def blackbox(pattern=generate_pattern()[1], string=""):
28     try:
29         return ("ok", len(findall(escape(pattern), string)))
30     except:

```

```

31         return ("err", "Something went wrong!")
32
33 def main():
34     isu = inp("Enter your ISU number")
35     if isu.isdecimal() and len(isu) == 6:
36         isu = int(isu)
37     else:
38         log("err", "Wrong ISU number!")
39     pattern = generate_pattern(isu)
40     if pattern[0] == "ok":
41         log(("ok", "Your emoji is %s" % pattern[1]))
42     else:
43         log(pattern)
44         exit()
45     pattern = pattern[1]
46     try:
47         from pyfiglet import figlet_format as cool_font_print
48         print(cool_font_print(' '.join([i for i in pattern])))
49     except ModuleNotFoundError:
50         log("For best experience please install pyfiglet module and run script again! (pip3 install pyfiglet)")
51
52     while True:
53         string = inp("Enter your string")
54         res = blackbox(pattern, string)
55         if res[0] == "ok":
56             log(("ok", "Number of matches is %s" % res[1]))
57         else:
58             log(res)
59             exit()
60
61 if __name__ == "__main__":
62     try:
63         main()
64     except KeyboardInterrupt:
65         print("\nBye!")

```

### 3.2.2. Пример вывода

```

/mnt/c/wsl/itmo-labs/computer-science/lab-3/task-1(main*) » python3 script.py
[?] Enter your ISU number:
> 335050
[+] Your emoji is :-{0
( )  |_____| < <  | | | |
( )  |_____| < <  | | | |
      \  \      \  \
[?] Enter your string:
> My favorite emoji is :-{0
[+] Number of matches is 1
[?] Enter your string:
> :-{0:-{0:-{0
[+] Number of matches is 3
[?] Enter your string:
> ^C
Bye!

```

Рис. : Скриншот фрагмента сессии в ZSH

### 3.2.3. Тесты

Ознакомиться с исходным кодом тестов можно на моем GitHub: <https://github.com/anim3boy/itmo-labs/blob/main/computer-science/lab-3/task-1/tests.py>.

```
/mnt/c/wsl/itmo-labs/computer-science/lab-3/task-1(main*) » python3 -m unittest tests.py
.....
-----
Ran 5 tests in 0.006s

OK
```

Рис. : Скриншот фрагмента сессии в ZSH

## 3.3. Доп. задание №1

### 3.3.1. Исходный код

Исходный код скрипта доступен также на моем GitHub: <https://github.com/anim3boy/itmo-labs/blob/main/computer-science/lab-3/task-2/script.py>.

#### Листинг 3.2: Исходный код

```
1 from re import sub
2
3 REGEX = r"([0-1]?[0-9]|[2][0-3]):([0-5][0-9]):([0-5][0-9])?"
4 PLUG = "(TBD)"
5
6 log = lambda x: print(decorator(x))
7 inp = lambda x: input("[?] %s:\n > " % x)
8
9 def decorator(text):
10     if type(text) is tuple:
11         if text[0] == "ok":
12             return "[+] %s" % text[1]
13         if text[0] == "err":
14             return "[-] %s" % text[1]
15     else:
16         return "[!] %s" % text
17
18 def fix(text):
19     try:
20         return ("ok", sub(REGEX, PLUG, text))
21     except:
22         return ("err", "Something went wrong!")
23
24 def main():
25     while True:
26         text = inp("Please enter source text")
27         fixed_text = fix(text)
28         if fixed_text[0] == "ok":
29             log(("ok", "Fixed text: %s" % fixed_text[1]))
30         else:
31             log(fixed_text)
32
33 if __name__ == "__main__":
34     try:
35         main()
36     except KeyboardInterrupt:
37         print("\nBye!")
```

### 3.3.2. Пример вывода

```
/mnt/c/wsl/itmo-labs/computer-science/lab-3/task-2(main*) » python3 script.py riot@anim3boy
[?] Please enter source text:
> Уважаемые студенты! В эту субботу в 15:00 планируется доп. занятие на 2 часа. То есть в 17:00:01 оно уже точно кончит
ся.
[+] Fixed text: Уважаемые студенты! В эту субботу в (TBD) планируется доп. занятие на 2 часа. То есть в (TBD) оно уже точ
но кончится.
[?] Please enter source text:
> ^C
Bye!
```

Рис. : Скриншот фрагмента сессии в ZSH

### 3.3.3. Тесты

Ознакомиться с исходным кодом тестов можно на моем GitHub: <https://github.com/anim3boy/itmo-labs/blob/main/computer-science/lab-3/task-2/tests.py>.

```
/mnt/c/wsl/itmo-labs/computer-science/lab-3/task-2(main*) » python3 -m unittest tests.py
.....
-----
Ran 6 tests in 0.008s

OK
```

Рис. : Скриншот фрагмента сессии в ZSH

## 3.4. Доп. задание №2

### 3.4.1. Исходный код

Исходный код скрипта доступен также на моем GitHub: <https://github.com/anim3boy/itmo-labs/blob/main/computer-science/lab-3/task-3/script.py>.

#### Листинг 3.3: Исходный код

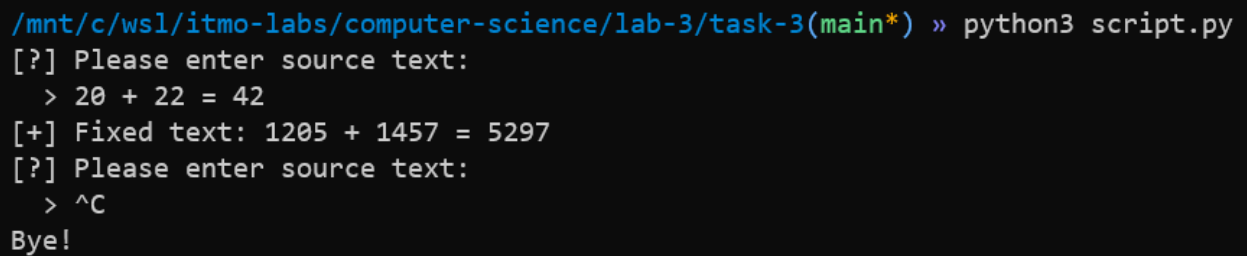
```
1 from re import sub
2
3 REGEX = r"(-|)\d+"
4
5 mod = lambda x: str(3*int(x.group(0))**2 + 5)
6
7 log = lambda x: print(decorator(x))
8 inp = lambda x: input("[?] %s:\n > " % x)
9
10
11 def decorator(text):
12     if type(text) is tuple:
13         if text[0] == "ok":
14             return "[+] %s" % text[1]
15         if text[0] == "err":
16             return "[-] %s" % text[1]
17     else:
18         return "[!] %s" % text
19
```

```

20 def modificate(text):
21     try:
22         return ("ok", sub(REGEX, mod, text))
23     except Exception as e:
24         print(e)
25         return ("err", "Something went wrong!")
26
27 def main():
28     while True:
29         text = inp("Please enter source text")
30         fixed_text = modificate(text)
31         if fixed_text[0] == "ok":
32             log(("ok", "Fixed text: %s" % fixed_text[1]))
33         else:
34             log(fixed_text)
35
36 if __name__ == "__main__":
37     try:
38         main()
39     except KeyboardInterrupt:
40         print("\nBye!")

```

### 3.4.2. Пример вывода



```

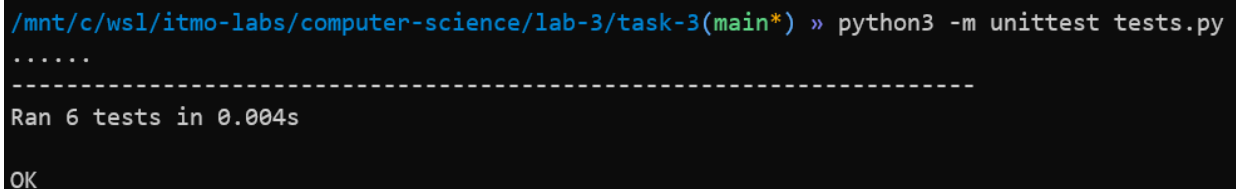
/mnt/c/wsl/itmo-labs/computer-science/lab-3/task-3(main*) » python3 script.py
[?] Please enter source text:
> 20 + 22 = 42
[+] Fixed text: 1205 + 1457 = 5297
[?] Please enter source text:
> ^C
Bye!

```

Рис. : Скриншот фрагмента сессии в ZSH

### 3.4.3. Тесты

Ознакомиться с исходным кодом тестов можно на моем GitHub: <https://github.com/anim3boy/itmo-labs/blob/main/computer-science/lab-3/task-3/tests.py>.



```

/mnt/c/wsl/itmo-labs/computer-science/lab-3/task-3(main*) » python3 -m unittest tests.py
.....
-----
Ran 6 tests in 0.004s
OK

```

Рис. : Скриншот фрагмента сессии в ZSH

### 3.4.4. Дополнение

Также я написал небольшую утилиту для шифрования и дешифрования сообщений. Принцип шифрования основывается на принципе из доп. задания №2, с тем отличием, что преобразование затрагивает не натуральные числа,



а символы. Данный тип шифрования данных имеет небольшой практический смысл, однако может наглядно иллюстрировать общие принципы. Ознакомиться с исходным кодом можно на моем GitHub: <https://github.com/anim3boy/itmo-labs/blob/main/computer-science/lab-3/task-3/crypt.py>.

```
/mnt/c/wsl/itmo-labs/computer-science/lab-3/task-3(main*) » ./crypt.py encode "Welcome to ITMO, buddy" riot@anim3boy
e5a2b8e79e90e8a2b5e78ba0e981a8e8ad80e79e90e0b085e9b6b5e981a8e0b085e3b9b8e58ab5e49680e4a4a8e19ab5e0b085e78291ea81b0e794b5e
794b5eaae98
-----
/mnt/c/wsl/itmo-labs/computer-science/lab-3/task-3(main*) » ./crypt.py decode e5a2b8e79e90e8a2b5e78ba0e981a8e8ad80e79e90e
0b085e9b6b5e981a8e0b085e3b9b8e58ab5e49680e4a4a8e19ab5e0b085e78291ea81b0e794b5e794b5eaae98
Welcome to ITMO, buddy
-----
/mnt/c/wsl/itmo-labs/computer-science/lab-3/task-3(main*) » _ riot@anim3boy
```

Рис. : Скриншот фрагмента сессии в ZSH

### 3.5. Выводы

В ходе выполнения данной лабораторной работы я узнал несколько непопулярных возможностей модуля `re`, например возможность передать в `re.sub` функцию вместо шаблона для замены. Написал несколько скриптов, впервые попробовал себя в написании Unit-тестов, а также создал небольшую утилиту для "шифрования" текстовых сообщений.

# Литература

- [1] Regular Expression HOWTO [Электронный ресурс] // Python Documentation — <https://docs.python.org/3/howto/regex.html> — (дата обращения: 10.10.2021).
- [2] Регулярные выражения в Python от простого к сложному. Подробности, примеры, картинки, упражнения. [Электронный ресурс] // Хабр — <https://habr.com/ru/post/349860/> — (дата обращения: 10.10.2021).