**Московский государственный технический университет им. Н.Э.Баумана**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Отчет по лабораторной работе №4

«Python. Функциональные возможности»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Преподаватель: |
| Катков Александр, ИУ5-52 |  | Гапанюк Ю.Е. |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |  |

2017 г.

1. **Задание лабораторной работы.**

Разработать приложение для построения графиков тригонометрических функций на языке Javascript с HTML интерфейсом.

Ход работы:

1. Ознакомиться с теоретической частью

2. Создайте новый проект PyCharm тип проекта: Pure Python (мы не будем использовать Python в этой работе, просто это позволяет создать абсолютно пустой проект без зависимостей)

3. Добавьте в проект 2 файла:

a. index.html

b. index.js

4. Сверстайте страницу со следующими элементами:

a. два поля ввода для области определения аргумента (<input>)

b. поле для ввода функции (<input>)

c. кнопка “Построить график” (<button>)

d. поле вывода графика (<div>)

5. При помощи css укажите размеры блока графика, отличные от нуля

6. Присвойте каждому полю уникальный class (например, from, to, fun, output и т.д.)

7. Убедитесь, что ваша страница отображается в браузере нормально

8. Подключите jQuery, flot и ваш скрипт в index.html, используя теги <script>

<script src= "https://code.jquery.com/jquery-2.2.4.min.js" ></script>

<script src= "https://cdnjs.cloudflare.com/ajax/libs/flot/0.8.3/jquery.flot.js" ></script>

9. Переходим к разработке скрипта

10. Дождитесь загрузки страницы

$(function() {

// …

})

11. Найдите все элементы управления на вашей странице var $from = $(‘.from);

12. Подпишитесь на событие нажатия кнопки

$button.click(onClick);

13. Отмените действие по-умолчанию (отправку формы)

e.preventDefault()

14. Получите значения из полей ввода

$from.val()

15. Не забудьте преобразовать числовые значения из строк в числа parseFloat, parseInt

16. Создайте массив пар значений

const points = [[x1, y1], …, [xn, yn]];

17. Для того, чтобы получить значение функции, заданной в виде строки, используйте функцию eval() **const** x = **0.1** ; **const** fun = 'Math.sin(x)' ;

**const** y = eval (fun);

18. Постройте график по точкам

$. plot ( $ output, [ points ], {});

19. Проверьте правильность работы приложения, в случае проблем, воспользуйтесь отладчиком Chrome DevTools

20. Проверьте построение графиков функций:

a. Math.sin(x)

b. Math.random()

c. Math.exp(x)

21. Выведите название построенной функции в легенду:

<http://www.flotcharts.org/flot/examples/basic-options/index.html>

22. Дополнительное задание: сделайте анимацию графика функции как на осциллографе для этого по таймеру setInterval() / clearInterval() перестраивайте график функции, прибавляя к ***x*** изменяющийся коэффициент ***dx.***

1. **Листинг**

**ex\_1.py**

**from** Gens **import** field  
**from** Gens **import** gen\_random  
  
goods = [  
 {**'title'**: **'Ковер'**, **'price'**: 2000, **'color'**: **'green'**},  
 {**'title'**: **'Диван для отдыха'**, **'price'**: 5300, **'color'**: **'black'**},  
 {**'title'**: **'Стелаж'**, **'price'**: 7000, **'color'**: **'white'**},  
 {**'title'**: **'Вешалка для одежды'**, **'price'**: 800, **'color'**: **'white'**}  
]  
  
*# Реализация задания 1*print(list(field(goods,**'title'**)))  
print(list(field(goods,**'title'**,**'price'**)))  
print(list(gen\_random(1,3,5)))

**ex\_2.py**

**from** Gens **import** gen\_random  
**from** iterators **import** Unique  
  
data1 = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]  
data2 = gen\_random(1, 3, 10)  
data3 = [**'a'**, **'A'**, **'b'**, **'B'**]  
  
*# Реализация задания 2***for** element **in** Unique(data1):  
 print(element, end=**' '**)  
  
print(**'\n'**)  
  
**for** element **in** Unique(list(data2)):  
 print(element, end=**' '**)  
  
print(**'\n'**)  
  
**for** element **in** Unique(list(data3)):  
 print(element, end=**' '**)  
  
print(**'\n'**)  
  
**for** element **in** Unique(list(data3), ignore\_case=**True**):  
 print(element, end=**' '**)

**ex\_3.py**

data = [4, -30, 100, -100, 123, 1, 0, -1, -4]  
*# Реализация задания 3*print(sorted(data, key=**lambda** x: abs(x)))

**ex\_4.py**

**from** decoretors **import** print\_result  
  
*# Необходимо верно реализовать print\_result  
# и задание будет выполнено  
#*@print\_result  
**def** test\_1():  
 **return** 1  
  
@print\_result  
**def** test\_2():  
 **return 'iu'**@print\_result  
**def** test\_3():  
 **return** {**'a'**: 1, **'b'**: 2}  
  
@print\_result  
**def** test\_4():  
 **return** [1, 2]  
  
test\_1()  
test\_2()  
test\_3()  
test\_4()

**ex\_5.py**

**from** time **import** sleep  
**from** Ctxmnrgs **import** timer  
  
**with** timer():  
 sleep(5.5)

**ex\_6.py**

**import** json  
**import** sys  
**from** Gens **import** field, gen\_random  
**from** decoretors **import** print\_result  
**from** iterators **import** Unique **as** unique  
**from** Ctxmnrgs **import** timer  
  
path = **"data\_light\_cp1251.json"  
  
with** open(path) **as** f:  
 data = json.load(f)  
  
**def** f1(arg):  
 **return** (sorted([i **for** i **in** unique([j[**'job-name'**] **for** j **in** arg], ignore\_case=**True**)]))  
  
**def** f2(arg):  
 **return** (filter(**lambda** x: (x.lower().find(**'программист'**) == 0), arg))  
  
**def** f3(arg):  
 **return** list(map(**lambda** x: x + **" с опытом Python"**, arg))  
  
@print\_result  
**def** f4(arg):  
 **return** ([**"{}, {} {} {}"**.format(x, **"зарплата"**, y, **"руб."**) **for** x, y **in** zip(arg, list(gen\_random(100000, 200000, len(arg))))])  
  
**with** timer():  
 f4(f3(f2(f1(data))))

**Gens.ру**

**import** random  
  
**def** field(items, \*args):  
 **assert** len(args) > 0  
 **if** len(args) == 1:  
 **for** i **in** items:  
 **if** args[0] **in** i: **yield** i[args[0]]  
 **else**:  
 **for** i **in** items:  
 res = {}  
 **for** j **in** args:  
 **if** j **in** i:  
 res[j] = i[j]  
 **yield** res  
  
  
**def** gen\_random(begin, end, num\_count):  
 **for** i **in** range (num\_count):  
 **yield** random.randint(begin, end)

**iterators.py**  
**class** Unique(object):  
 **def** \_\_init\_\_(self, items, \*\*kwargs):  
 **if** (**'ignore\_case' in** kwargs.keys()) **and** (kwargs[**'ignore\_case'**]):  
 self.items = [str(i).lower() **for** i **in** items]  
 **else**:  
 self.items = items  
 self.index = 0  
 self.used = []  
  
 **def** \_\_next\_\_(self):  
 **while** self.items[self.index] **in** self.used:  
 **if** self.index == len(self.items) - 1:  
 **raise** StopIteration  
 self.index += 1  
  
 self.used.append(self.items[self.index])  
 **return** self.items[self.index]  
  
 **def** \_\_iter\_\_(self):  
 **return** self

**Ctxmnrgs.py**

**import** time  
  
**class** timer:  
 **def** \_\_enter\_\_(self):  
 self.t = time.time()  
  
 **def** \_\_exit\_\_(self, exp\_type, exp\_value, traceback):  
 print(time.time() - self.t)

1. **Результат:**











