

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**

**Языки программирования  
Отчет по лабораторной работе №6**

Декораторы  
функций в языке Python

Выполнил студент группы

ИТС-б-о-20-1

Скрыпник А.С. « » \_\_\_\_\_ 20\_\_ г.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

Проверил доцент  
Кафедры инфокоммуникаций, старший  
преподаватель  
Воронкин Р.А.

\_\_\_\_\_  
(подпись)

Ставрополь 2021

**Цель работы:** приобретение навыков по работе с декораторами функций при написании программ с помощью языка программирования Python версии 3.x.

**Порядок выполнения работы:**

- 1) Создал публичный репозиторий LR-5.
- 2) Проработал пример 1.

```
[*] Время выполнения: 2.9096245765686035 секунд.  
Process finished with exit code 0
```

Рисунок 1. Пример 1.

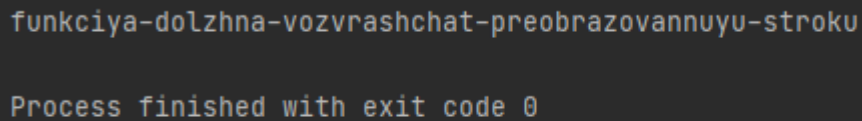
- 3) Проработал пример 2.

```
[*] Время выполнения: 1.0790035724639893 секунд.  
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="ru"><head><meta content="&#1055;&#1086;&#1080;&#1089;&#1082; &#1080;&#1085;&#1092;  
var f=this||self;var h,k=[];function l(a){for(var b;a&&(!a.getAttribute)||!(b=a.getAttribute("eid")));a=a.parentNode;return b||h}function m(a){for(var b=null;a  
function n(a,b,c,d){var e="";c||-1==b.search("&#1055;")||(e="&#1055;"+l(d),-1==b.search("&#1055;"))&&(d=m(d))&&(e+="&#1055;"+d);d="";!c&&f._cshid&&-1==b.search("&#1055;  
google.y={};google.sy=[];google.x=function(a,b){if(a)var c=a.id;else{do c=Math.random();while(google.y[c])}google.y[c]=[a,b];return 1};google.sx=function(a){go  
document.documentElement.addEventListener("submit",function(b){var a;if(a=b.target){var c=a.getAttribute("data-submitfalse");a="1"===c||"q"===c&&!a.elements.q.  
</style><style>body,td,a,p,.h{font-family:arial,sans-serif}body{margin:0;overflow-y:scroll}#gog{padding:3px 8px 0}td{line-height:.8em}.gac_m td{line-height:1.7p  
var f=this||self;var g,h,k=null!==(g=f.mei)&&void 0!==(g?g:1,l=null!==(h=f.sdo)&&void 0!==(h?h:!0,p=0,q,r=google.end,u=r.jsr;google.ml=function(a,b,d,m,c){c=void  
e);var n=a.fileName;n&&(b+="&script="+c(n),e&&n===window.location.href&&(e=document.documentElement.outerHTML.split("\n")[e],b+="&cad="+c(e?e.substring(0,300)).  
if (!iesg){document.f&&document.f.q.focus();document.gbqf&&document.gbqf.q.focus();}  
})();</script><div id="mngb"><div id=gbar><noabr><b class=gbl>&#1055;&#1086;&#1080;&#1089;&#1082;</b> <a class=gbl href="https://www.google.ru/imghp?hl=ru&tab=w  
else top.location='/doodles/';});});</script><input value="ALS-WAMAAAAAYcDnL0-h3Hwa0NTAVwispjZPLKUEWn0d" name="iflsig" type="hidden"></span></span></td><td cl  
var a,b="1";if(document&&document.getElementById)if("undefined"!=typeof XMLHttpRequest)b="2";else if("undefined"!=typeof XMLHttpRequest){var c,d,e=["MSXML2.XMLH  
var a=window.innerWidth,b=window.innerHeight;if(!a||!b){var c=window.document,d="CSS1Compat"==c.compatMode?c.documentElement:c.body;a=d.clientWidth;b=d.clientH  
var e=this||self,f=function(a){return a};var g;var l=function(a,b){this.g=b=="h?":"";l.prototype.toString=function(){return this.g+""};var h={};  
function m(){var a=u;google.lx=function(){n(a);google.lx=function(){};google.bx||google.lx()}  
function n(a){google.timers&&google.timers.load&&google.tick&&google.tick("load","xjsls");var b=document;var c="SCRIPT";"application/xhtml+xml"===b.contentType  
function _F_installCss(c){  
(function(){google.jl={attn:false,blt:'none',chnk:0,dw:false,dwu:true,emt:0,end:0,ine:false,lls:'default',pdt:0,rep:0,snet:true,strt:0,ubm:false,uwp:true});})()  
Process finished with exit code 0
```

Рисунок 2. Пример 2.

4) Объявите функцию с именем `to_lat`, которая принимает строку на кириллице преобразовывает ее в латиницу, используя следующий словарь для замены русских букв на соответствующее латинское написание: Функция должна возвращать преобразованную строку. Замены делать без учета регистра (исходную строку перевести в нижний регистр – малые буквы). Все небуквенные символы `"! ? ; , _"` превращать в символ `'-'` (дефиса). Определите декоратор для этой функции, который несколько подряд идущих дефисов, превращает в один дефис. Полученная строка должна возвращаться при

вызове декоратора. Примените декоратор к функции `to_lat` и вызовите ее. Результат работы декорированной функции отобразите на экране.



```
funkciya-dolzha-vozvrashchat-preobrazovannuyu-stroku  
Process finished with exit code 0
```

Рисунок 3. Индивидуальное задание 1

### Ответы на контрольные вопросы:

1) Что такое декоратор?

Ответ: Декоратор — это функция, которая позволяет обернуть другую функцию для расширения её функциональности без непосредственного изменения её кода. Какие аргументы называются именованными в Python?

2) Почему функции являются объектами первого класса?

Ответ: Объектами первого класса в контексте конкретного языка программирования называются элементы, с которыми можно делать всё то же, что и с любым другим объектом: передавать как параметр, возвращать из функции и присваивать переменной.

3) Каково назначение функций высших порядков?

Ответы: Функции высших порядков — это такие функции, которые могут принимать в качестве аргументов и возвращать другие функции.

4) Как работают декораторы?

Ответ: Раз мы знаем, как работают функции высших порядков, теперь мы можем понять как работают декораторы. Сначала посмотрим на пример декоратора:

```
def decorator_function(func):
    def wrapper():
        print('Функция-обёртка!')
        print('Оборачиваемая функция: {}'.format(func))
        print('Выполняем обёрнутую функцию...')
        func()
        print('Выходим из обёртки')
    return wrapper
```

Здесь `decorator_function()` является функцией-декоратором. Как вы могли заметить, она является функцией высшего порядка, так как принимает функцию в качестве аргумента, а также возвращает функцию. Внутри `decorator_function()` мы определили другую функцию, обёртку, так сказать, которая обёртывает функцию-аргумент и затем изменяет её поведение. Декоратор возвращает эту обёртку. Теперь посмотрим на декоратор в действии:

```
>>> @decorator_function
... def hello_world():
...     print('Hello world!')
...
>>> hello_world()
Оборачиваемая функция: <function hello_world at 0x032B26A8>
Выполняем обёрнутую функцию...
Hello world!
Выходим из обёртки
```

## 5) Как работают декораторы?

Ответ: Ответ в предыдущем вопросе.

**Вывод:** в ходе лабораторной работы, были приобретены навыки по работе с декораторами функций при написании программ с помощью языка программирования Python версии 3.x.