

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ**

**Отчет о лабораторной работе №14 по дисциплине основы программной
инженерии**

Выполнила:

Емельянова Яна

Александровна, 2 курс,

группа ПИЖ-б-о-20-1,

Проверил:

Доцент кафедры инфокоммуникаций,

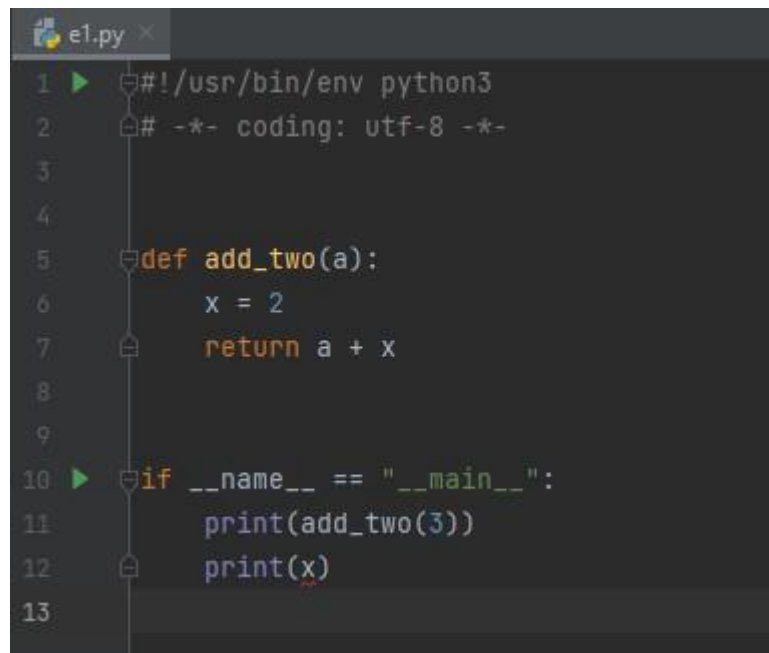
Воронкин Р.А.

Ставрополь, 2021 г.

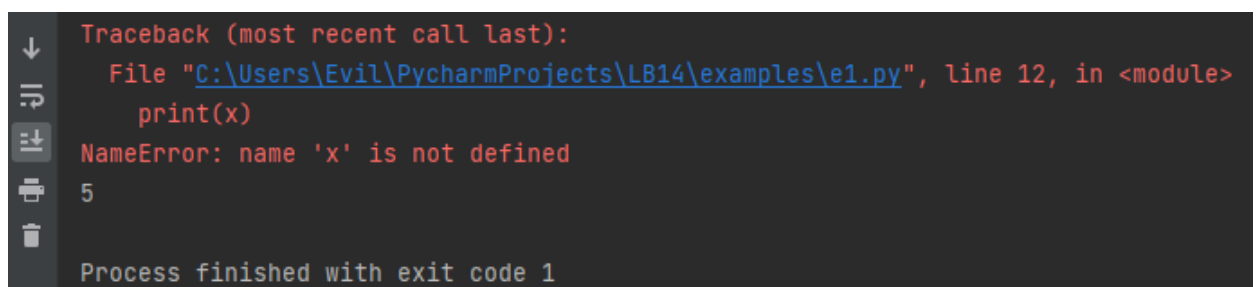
1. Замыкания в языке Python

Примеры из методических указаний

Область видимости Local



```
e1.py x
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4
5  def add_two(a):
6      x = 2
7      return a + x
8
9
10 ▶  if __name__ == "__main__":
11     print(add_two(3))
12     print(x)
13
```



```
Traceback (most recent call last):
  File "C:\Users\Evil\PycharmProjects\LB14\examples\e1.py", line 12, in <module>
    print(x)
NameError: name 'x' is not defined
5
Process finished with exit code 1
```

Область видимости Enclosing

```
e2.py x
1 ▶ #!/usr/bin/env python3
2   # -*- coding: utf-8 -*-
3
4
5   def add_four(a):
6       x = 2
7
8       def add_some():
9           print("x = " + str(x))
10          return a + x
11
12      return add_some()
13
14
15 ▶ if __name__ == "__main__":
16     print(add_four(5))
17
```

↓
x = 2
7
Process finished with exit code 0

Область видимости Global

```
e3.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4     x = 4
5
6
7     def fun():
8         print(x + 3)
9
10
11 ▶ if __name__ == "__main__":
12     fun()
13
```

```
7
Process finished with exit code 0
```

Как использовать замыкания в Python?

```
e4.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4
5 def mul(a):
6     def helper(b):
7         return a * b
8     return helper
9
10
11 ▶ if __name__ == "__main__":
12     print(mul(5)(2))
13     new_mul5 = mul(5)
14     print(new_mul5)
15     print(new_mul5(2))
16     print(new_mul5(7))
17
```

```
↓
10
⌵
<function mul.<locals>.helper at 0x000002CE330CD280>
10
⌵
35
⌵
Process finished with exit code 0
```

```
e5.py x
1  ▶  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  def fun1(a):
6      x = a * 3
7
8  def fun2(b):
9      nonlocal x
10     return b + x
11
12     return fun2
13
14
15  ▶  if __name__ == "__main__":
16      test_fun = fun1(4)
17      print(test_fun(7))
18
```

```
19
↓
Process finished with exit code 0
```

Свойство замыкания – средство для построения
иерархических данных

```
e6.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 ▶ if __name__ == "__main__":
5     tpl = lambda a, b: (a, b)
6     a = tpl(1, 2)
7     print(a)
8     b = tpl(3, a)
9     print(b)
10    c = tpl(a, b)
11    print(c)
12
```

```
↓ (1, 2)
⏮ (3, (1, 2))
⏮ ((1, 2), (3, (1, 2)))
⏮
🖨 Process finished with exit code 0
🗑
```

1.1 Индивидуальное задание (рис 1-3).

Вариант 9

```
ind_zad.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4
5 def func(type="max"):
6     def calc(coll):
7         if type == "max":
8             elem = max(coll)
9         if type == "min":
10            elem = min(coll)
11        return elem
12
13    return calc
14
15
16 ▶ if __name__ == "__main__":
17     type = input("Введите тип значения - min или max: ")
18     print("Введите кортеж элементов: ")
19     a = (int(i) for i in input().split())
20     print(f"Результат работы функции: {func(type)(a)}")
21
```

Рисунок 1 – Код программы

```
Введите тип значения - min или max: min
Введите кортеж элементов:
1 2 3 4 4 -78 242 211
Результат работы функции: -78

Process finished with exit code 0
```

Рисунок 2 – Результат выполнения программы

```
Введите тип значения - min или max: max
Введите кортеж элементов:
1 2 3 4 99 43 34434 566 2
Результат работы функции: 34434

Process finished with exit code 0
```

Рисунок 3 – Результат выполнения программы

2. Ответы на контрольные вопросы

1. Что такое замыкание?

Замыкание (closure) в программировании — это функция, в теле которой присутствуют ссылки на переменные, объявленные вне тела этой функции в окружающем коде и не являющиеся ее параметрами.

2. Как реализованы замыкания в языке программирования Python?

Необходимо объявить вложенную функцию в объемлющей функции. Эта вложенная функция должна ссылаться на значение переменных, объявленных в объемлющей функции, необходимо, чтобы объемлющая функция возвращала значение вложенной функции.

3. Что подразумевает под собой область видимости Local?

Эту область видимости имеют переменные, которые создаются и используются внутри функций.

4. Что подразумевает под собой область видимости Enclosing?

Суть данной области видимости в том, что внутри функции могут быть вложенные функции и локальные переменные, так вот локальная переменная функции для ее вложенной функции находится в enclosing области видимости.

5. Что подразумевает под собой область видимости Global?

Переменные области видимости global – это глобальные переменные уровня модуля (модуль – это файл с расширением .py).

6. Что подразумевает под собой область видимости Build-in?

В рамках этой области видимости находятся функции open, len и т. п.,

также туда входят исключения. Эти сущности доступны в любом модуле Python и не требуют предварительного импорта. Built-in – это максимально широкая область видимости.

7. Как использовать замыкания в языке программирования Python?

Пример:

```
def mul(a):  
    def helper(b):  
        return a * b  
    return helper
```

Использование: `mul(5)(2)` или `new_mul5 = mul(5)`, `new_mul5(2)`

8. Как замыкания могут быть использованы для построения иерархических данных?

В общем случае, операция комбинирования объектов данных обладает свойством замыкания в том случае, если результаты соединения объектов с помощью этой операции сами могут соединяться этой же операцией. Это свойство позволяет строить иерархические структуры данных. Покажем это на примере кортежей в Python:

```
tpl = lambda a, b: (a, b)  
a = tpl(1, 2)  
(1, 2)  
b = tpl(3, a)  
(3, (1, 2))  
c = tpl(a, b)  
((1, 2), (3, (1, 2)))
```