

## Лабораторная работа №6

Урусов Максим Андреевич

ИВТ-б-о-21-1

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

### 1. Создал общедоступный репозиторий на GitHub с MIT

**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

---

**Owner \*** **Repository name \***

MaksimUrusov /

Great repository names are short, simple, and unique. The repository `laba-2.12` already exists on this account. [glowing-giggle?](#)

**Description** (optional)

---

☒ **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.

---

**Initialize this repository with:**  
Skip this step if you're importing an existing repository.

☒ **Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)

**Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

**Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)

---

You are creating a public repository in your personal account.

---

2. Выполнил клонирование созданного репозитория.

```
D:\Program Files\Git>git clone https://github.com/MaksimUrusov/Laba-2.12.git
Cloning into 'Laba-2.12'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
```

3. Дополнил файл .gitignore необходимыми правилами для работы с IDE PyCharm.

```
# Created by https://www.toptal.com/developers/gitignore/api/python,pycharm
# Edit at https://www.toptal.com/developers/gitignore?templates=python,pycharm

## PyCharm ##
# Covers JetBrains IDEs: IntelliJ, RubyMine, PhpStorm, AppCode, PyCharm, CLion, Android Studio, WebStorm and Rider
# Reference: https://intellij-support.jetbrains.com/ru/en-us/articles/206544839

# User-specific stuff
.idea/**/workspace.xml
.idea/**/tasks.xml
.idea/**/usage.statistics.xml
.idea/**/dictionaries
.idea/**/shelf

# AWS User-specific
.idea/**/aws.xml

# Generated files
.idea/**/contentModel.xml

# Sensitive or high-churn files
.idea/**/dataSources/
.idea/**/dataSources.ids
.idea/**/dataSources.local.xml
.idea/**/sqlDataSources.xml
.idea/**/dynamic.xml
.idea/**/uiDesigner.xml
.idea/**/dbnavigator.xml

# Gradle
.idea/**/gradle.xml
.idea/**/libraries

# Gradle and Maven with auto-import
# When using Gradle or Maven with auto-import, you should exclude module files,
# since they will be recreated, and may cause churn. Uncomment if using
# auto-import
```

4. Организовал репозиторий в соответствии с моделью ветвления git-flow.

```
D:\Program Files\Git>cd laba-2.12
D:\Program Files\Git\Laba-2.12>git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]
fatal: '?[C' is not a valid branch name
error: pathspec '?[C' did not match any file(s) known to git
Fatal: Could not check out branch ' '.
```

5. Создал проект пайчарм

Имя	Дата изменения	Тип	Размер
.idea	18.10.2022 1:22	Папка с файлами	
doc	03.11.2022 4:19	Папка с файлами	
.gitignore	18.10.2022 1:22	Файл "GITIGNORE"	6 КБ
LICENSE	03.11.2022 4:01	Файл	2 КБ
README.md	03.11.2022 4:01	Файл "MD"	1 КБ

## 6. Проработал примеры ла

```
import math

new *
def decorator(func):
    new *
    def wrapper(r):
        print('Площадь круга равна = {:.3f}'.format(func(r)))
    return wrapper

new *
@decorator
def circle(r):
    s = math.pi * pow(r, 2)
    return s

if __name__ == "__main__":
    circle(11)
```

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  new *
5  def hello_world():
6      print('Hello world!')
7
8  new *
9  def higher_order(func):
10     print('Получена функция {} в качестве аргумента'.format(func))
11     func()
12     return func
13
14 if __name__ == '__main__':
15     higher_order(hello_world)
```

if \_\_name\_\_ == '\_\_main\_\_'

prim3 x

C:\ProgramData\Anaconda3\python.exe "D:\Program Files\Git\Laba-2.12\prim3.py"

Получена функция <function hello\_world at 0x000001AA5951F0D0> в качестве аргумента

Hello world!

Process finished with exit code 0

бораторной работы.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

new *
def hello_world():
    print("Hello world!")

if __name__ == '__main__':

    a = type(hello_world)
    print(a)
```

\_\_name\_\_ == '\_\_main\_\_'

prim1 x

C:\ProgramData\Anaconda3\python.exe "D:\Program Files\Git\Laba-2.12\prim1.py"

<class 'function'>

Process finished with exit code 0

```
new *
1 def benchmark(func):
2     import time
3
4     new *
5     def wrapper(*args, **kwargs):
6         start = time.time()
7         return_value = func(*args, **kwargs)
8         end = time.time()
9         print('[*] Время выполнения: {} секунд.'.format(end-start))
10        return return_value
11    return wrapper
12
13    new *
14    @benchmark
15    def fetch_webpage(url):
16        import requests
17        webpage = requests.get(url)
18        return webpage.text
19
20    webpage = fetch_webpage('https://google.com')
21    print(webpage)
```

prim4 x

C:\ProgramData\Anaconda3\python.exe "D:\Program Files\Git\Laba-2.12\prim4.py"

[\*] Время выполнения: 1.768993616104126 секунд.

7. Зафиксируйте сделанные изменения в репозитории. (после создания веток не запустил, поэтому не работало)

```
D:\REP6\LB2.11>git push --all
Enumerating objects: 19, done.
Counting objects: 100% (19/19), done.
Delta compression using up to 8 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (16/16), 92.88 kiB | 7.14 MiB/s, done.
Total 16 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/Arsen445/LB2.11.git
   30b5e5e..c836547  develop -> develop

D:\REP6\LB2.11>git status
On branch develop
nothing to commit, working tree clean

D:\REP6\LB2.11>_
```

8. Решите индивидуальное задание.

Используя замыкания функций, объявите внутреннюю функцию, которая на основе двух параметров вычисляет площадь фигуры. Какой именно фигуры: треугольника или прямоугольника, определяется параметром `type` внешней функции. Если `type` принимает значение 0, то вычисляется площадь треугольника, а иначе – прямоугольника. По умолчанию параметр `type` должен быть равен 0. Вычисленное значение должно возвращаться внутренней функцией. Вызовите внутреннюю функцию замыкания и отобразите на экране результат ее работы.

```

import math

new *
def decorator(func):
    new *
    def wrapper(r):
        print('Площадь круга равна = {:.3f}'.format(func(r)))
    return wrapper

new *
@decorator
def circle(r):
    s = math.pi * pow(r, 2)
    return s

if __name__ == "__main__":
    circle(11)

```

```

C:\ProgramData\Anaconda3\python.exe "D:\Program Files\Git\Laba-2.12\ind.py"
Площадь круга равна = 380.133

Process finished with exit code 0

```

## 9. Зафиксируйте сделанные изменения в репозитории.

```
nothing added to commit but untracked files present (use "git add" to track)
D:\REP6\LB2.11>git add --all
D:\REP6\LB2.11>git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   ind.py
D:\REP6\LB2.11>git commit -m "e1"
[develop f888dbd] e1
1 file changed, 31 insertions(+)
create mode 100644 ind.py
D:\REP6\LB2.11>git push --all
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 632 bytes | 632.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Arsen445/LB2.11.git
   c836547..f888dbd  develop -> develop
D:\REP6\LB2.11>
```

## 10. Выполните слияние ветки для разработки с веткой main/master.

```
D:\REP6\LB2.11>git checkout main
Unlink of file 'doc/лб6.docx' failed. Should I try again? (y/n) y
Unlink of file 'doc/лб6.docx' failed. Should I try again? (y/n) n
warning: unable to unlink 'doc/лб6.docx': invalid argument
Updating files: 100% (13/13), done.
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
D:\REP6\LB2.11>git status
On branch main
Your branch is up to date with 'origin/main'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        doc/
nothing added to commit but untracked files present (use "git add" to track)
D:\REP6\LB2.11>git add --all
D:\REP6\LB2.11>git status
On branch main
Your branch is up to date with 'origin/main'.
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   "doc/\320\233\320\2216.docx"
D:\REP6\LB2.11>git merge develop
Updating 71773ee..f888dbd
Fast-forward
 .idea/.name | 1 +
 .idea/LB2.8.iml | 8 ++++++
 .idea/inspectionProfiles/profiles_settings.xml | 6 +++++
 .idea/modules.xml | 8 ++++++
 .idea/vcs.xml | 6 +++++
 "doc/~$ \320\233\320\2216.docx" | Bin 0 -> 162 bytes
 "doc/\320\233\320\2216.docx" | Bin 0 -> 566932 bytes
 ind.py | 31 ++++++++++++++++++++++++++++++++++++++
 prim1.py | 15 ++++++
 prim2.py | 13 ++++++
 prim3.py | 9 ++++++

```

## 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?**

**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?**

Уровень Python интерпретатора. 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?**

```
>>> def fun1(a):  
    x = a * 3  
    def fun2(b):  
        nonlocal x  
        return b + x  
    return fun2  
  
>>> test_fun = fun1(4)  
  
>>> test_fun(7)  
19
```

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

```
>>> tp1 = lambda a, b: (a, b)
```

Если мы передадим в качестве аргументов числа, то, получим простой кортеж.

```
>>> a = tp1(1, 2)
>>> a
(1, 2)
```

Эту операцию можно производить не только над числами, но и над сущностями, ей же и порожденными.

```
>>> b = tp1(3, a)
>>> b
(3, (1, 2))

>>> c = tp1(a, b)
>>> c
((1, 2), (3, (1, 2)))
```