

Лабораторная работа 4

ТЕМА 4. Функции и модули

Лабораторные задания:

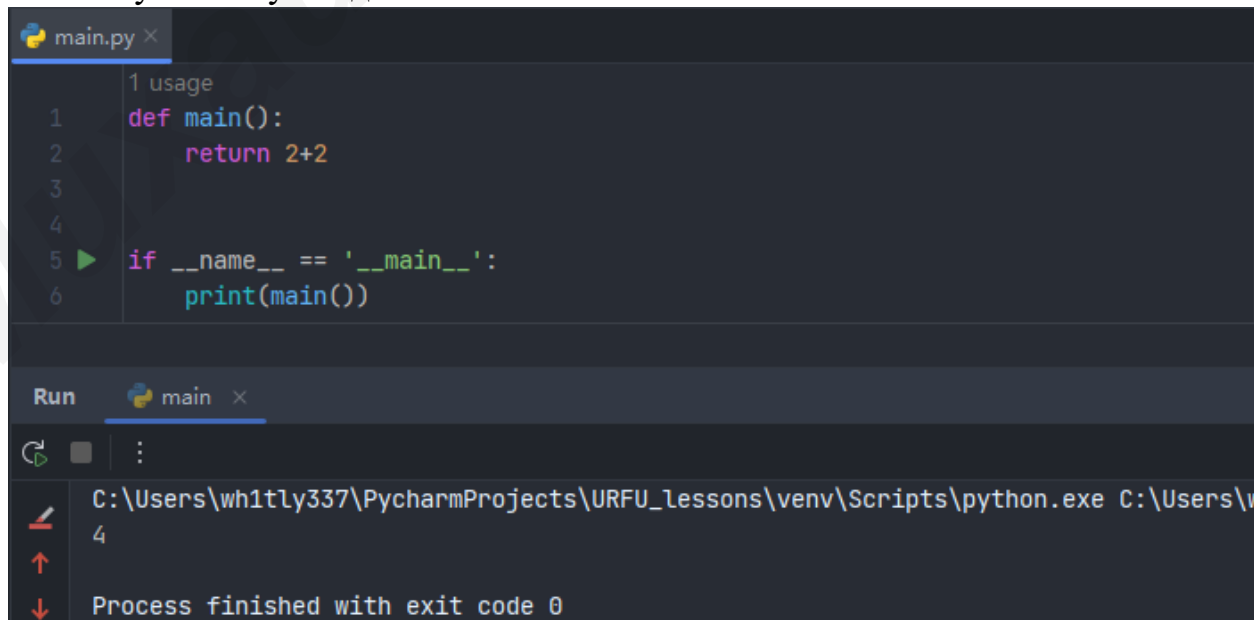
- 1) Напишите функцию, которая выполняет любые арифметические действия и выводит результат в консоль. Вызовите функцию используя “точку входа”.



```
main.py x
1 usage
2 def main():
3     print(2+2)
4
5 if __name__ == '__main__':
6     main()

Run main x
C:\Users\wh1tly337\PycharmProjects\URFU_lessons\venv\Scripts\python.exe C:\Users\w
4
Process finished with exit code 0
```

- 2) Напишите функцию, которая выполняет любые арифметические действия, возвращает при помощи return значение в место, откуда вызывали функцию. Выведите результат в консоль. Вызовите функцию используя “точку входа”.



```
main.py x
1 usage
2 def main():
3     return 2+2
4
5 if __name__ == '__main__':
6     print(main())

Run main x
C:\Users\wh1tly337\PycharmProjects\URFU_lessons\venv\Scripts\python.exe C:\Users\w
4
Process finished with exit code 0
```

Ниже представлена точно такая же программа, как и выше, только написана более развернуто. В этой программе стоит заметить что

результат работы функции `main()` мы помещаем в переменную `“answer”`, в дальнейшем можно как-то работать с ним, не вызывая функцию повторно, что хорошо сказывается, например, на скорости работы программы.



The screenshot shows a Python IDE with a file named `main.py`. The code in the editor is as follows:

```
1 usage
2 def main():
3     result = 2 + 2
4     return result
5
6 if __name__ == '__main__':
7     answer = main()
8     print(answer)
```

Below the editor, the 'Run' console shows the execution of the script. The command executed is `C:\Users\wh1tly337\PycharmProjects\URFU_lessons\venv\Scripts\python.exe C:\Users\w...`. The output is the number `4`. At the bottom, it states 'Process finished with exit code 0'.

- 3) Напишите функцию, в которую передаются два аргумента, над ними производится арифметическое действие, результат возвращается туда, откуда эту функцию вызывали. Выведите результат в консоль. Вызовите функцию в любом небольшом цикле.

На скриншоте ниже приведен пример программы, в которой аргумент функции `“x”` превращается в параметр `“one”`, то же самое происходит с `“y”` и `“two”`

```
main.py x
1 usage
2 def main(one, two):
3     result = one + two
4     return result
5
6 for i in range(5):
7     x = 1
8     y = 10
9     answer = main(x, y) # x = one, y = two
10    print(answer)

Run main x
C:\Users\wh1tly337\PycharmProjects\URFU_lessons\venv\Scripts\python.exe C:\Users\w
11
11
11
11
11
Process finished with exit code 0
```


Ниже представлена точно такая же программа, как и выше, только аргументы передаются в вызове функции, а не как отдельные переменные.

```
main.py x
1 usage
2 def main(one, two):
3     return one + two
4
5 for i in range(5):
6     answer = main(one=1, two=10)
7     print(answer)

Run main x
C:\Users\wh1tly337\PycharmProjects\URFU_lessons\venv\Scripts\python.exe C:\Users\w
11
11
11
11
11
Process finished with exit code 0
```

- 4) Напишите функцию, на вход которой подается какое-то изначально неизвестное количество аргументов, над которыми будет производиться арифметические действия. Для выполнения задания необходимо использовать кортеж “*args”. На скриншоте ниже приведен пример такой программы с комментариями.

Для закрепления понимания работы с кортежами настоятельно рекомендуем поменять аргументы вызова функции, вручную посчитать результат, только потом запустить программу с новыми значениями и проверить себя, насколько вы поняли данный аспект программирования.



```
1 usage
2 def main(x, *args):
3     one = x # 10
4     two = sum(args) # 0, 1, 2, -1, 0, -1, 1, 2
5     three = float(len(args)) # длина кортежа args
6     # пример работы операций над параметрами функции
7     print(f"one={one}\ntwo={two}\nthree={three}")
8
9     return x + sum(args) / float(len(args))
10
11 if __name__ == '__main__':
12     result = main(10, 0, 1, 2, -1, 0, -1, 1, 2)
13     print(f"\nresult={result}")
```

Run main

```
C:\Users\wh1tly337\PycharmProjects\URFU_lessons\venv\Scripts\python.exe C:\Users\w
one=10
two=4
three=8.0
result=10.5
Process finished with exit code 0
```

- 5) Напишите функцию, которая на вход получает кортеж “**kwargs” и при помощи цикла выводит значения, поступившие в функцию. На скриншоте ниже указаны два варианта вызова функции с “**kwargs” и два варианта работы с данными, поступившими в эту функцию. Комментарии в коде и теоретическая часть помогут вам разобраться в этом нелегком аспекте. Вызовите функцию используя “точку входа”.

```
main.py ×
1  # оператор ** упаковывает аргументы, переданные по имени, в словарь.
2  # Имена параметров служат ключами.
3  2 usages
4  def main(**kwargs):
5      for i in kwargs.items():
6          print(i[0], i[1])
7
8      print() # добавляет пустую строку в консоль
9
10     for key in kwargs:
11         print(f"{key} = {kwargs[key]}")
12
13  ▶ if __name__ == '__main__':
14      # обычный вызов функции, принимающий кортеж **kwargs, как параметр
15      main(x=[1, 2, 3], y=[3, 3, 0], z=[2, 3, 0], q=[3, 3, 0], w=[3, 3, 0])
16      print() # добавляет пустую строку в консоль
17      # также возможно использовать в виде аргумента для функции уже готовый
18      # словарь, но приписав в начале **
19      main(**{'x': [1, 2, 3], 'y': [3, 3, 0]})

Run main ×
C:\Users\whitly337\PycharmProjects\URFU_lessons\venv\Scripts\python.exe C:\Users\w
x [1, 2, 3]
y [3, 3, 0]
z [2, 3, 0]
q [3, 3, 0]
w [3, 3, 0]

x = [1, 2, 3]
y = [3, 3, 0]
z = [2, 3, 0]
q = [3, 3, 0]
w = [3, 3, 0]

x [1, 2, 3]
y [3, 3, 0]

x = [1, 2, 3]
y = [3, 3, 0]

Process finished with exit code 0
```

- 6) Напишите две функции. Первая – получает в виде параметра “**kwargs”. Вторая считает среднее арифметическое из значений первой функции. Вызовите первую функцию используя “точку входа” и минимум 4 аргумента.

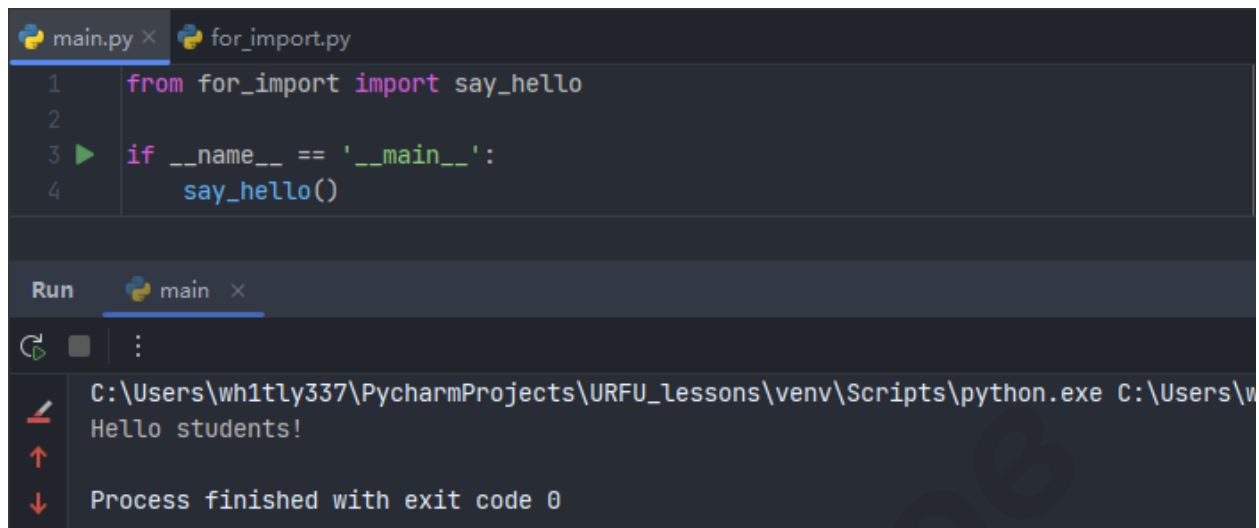
```
main.py x
1 usage
2 def main(**kwargs):
3     """Функция работающая с **kwargs."""
4     for i, j in kwargs.items():
5         print(f"{i}. Mean = {mean(j)}")
6
7
8 1 usage
9 def mean(data):
10     """Функция подсчета среднего арифметического."""
11     return sum(data) / float(len(data))
12
13
14 if __name__ == '__main__':
15     main(x=[1, 2, 3], y=[3, 3, 0])

Run main x
C:\Users\wh1tly337\PycharmProjects\URFU_lessons\venv\Scripts\python.exe C:\Users\w
x. Mean = 2.0
y. Mean = 2.0
Process finished with exit code 0
```

- 7) Создайте дополнительный файл .py. Напишите в нем любую функцию, которая будет что угодно выводить в консоль, но не вызывайте ее в нем. Откройте файл main.py, импортируйте в него функцию из нового файла и при помощи “точки входа” вызовите эту функцию.

```
Project v
v URFU_lessons C:\Users\wh
> venv library root
for_import.py
main.py
> External Libraries
> Scratches and Consoles

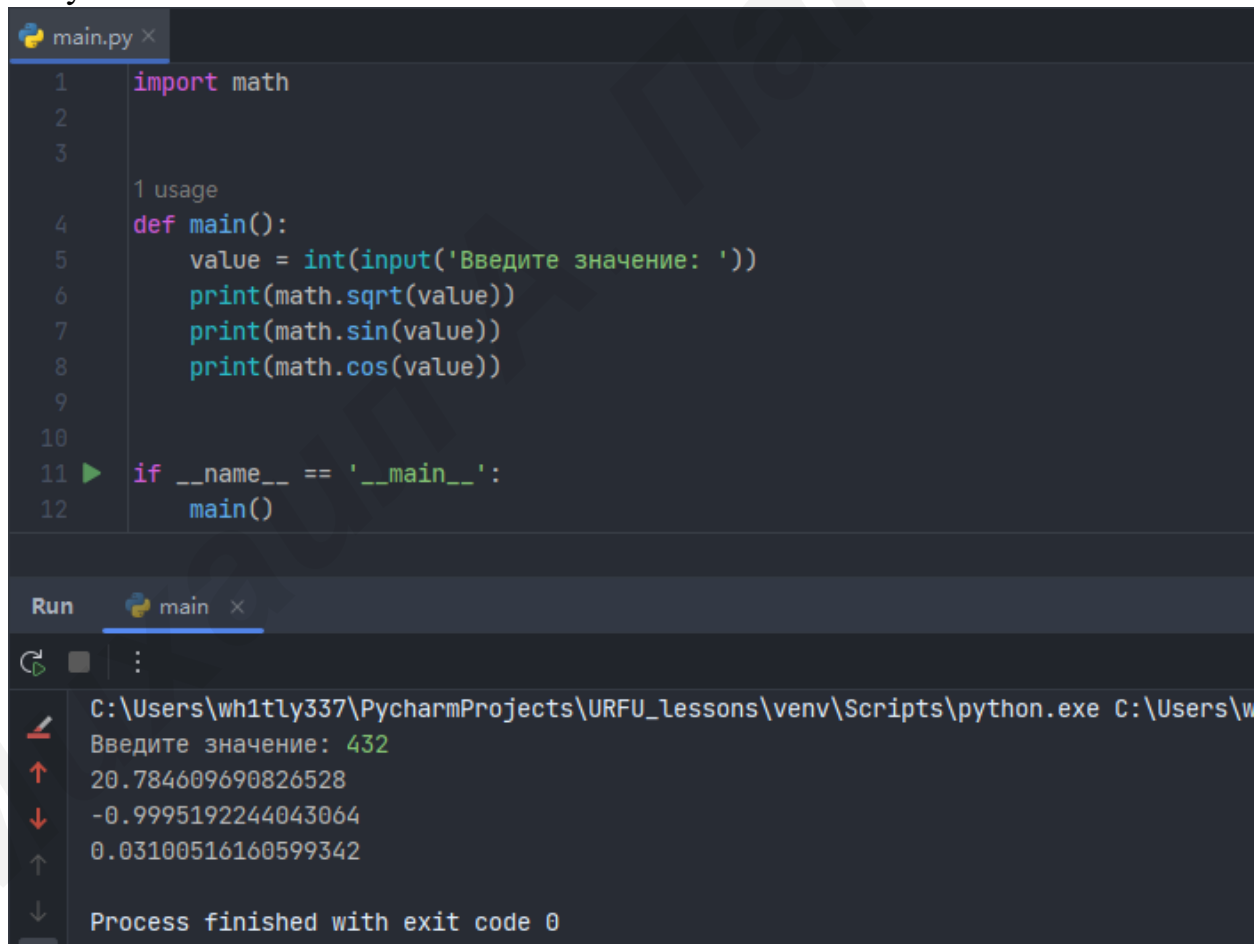
main.py for_import.py x
2 usages
1 def say_hello():
2     print('Hello students!')
3
```



```
main.py x for_import.py
1 from for_import import say_hello
2
3 if __name__ == '__main__':
4     say_hello()

Run main x
C:\Users\wh1tly337\PycharmProjects\URFU_lessons\venv\Scripts\python.exe C:\Users\w
Hello students!
Process finished with exit code 0
```

- 8) Напишите программу, которая будет выводить корень, синус, косинус полученного от пользователя числа.



```
main.py x
1 import math
2
3
4 1 usage
5 def main():
6     value = int(input('Введите значение: '))
7     print(math.sqrt(value))
8     print(math.sin(value))
9     print(math.cos(value))
10
11 if __name__ == '__main__':
12     main()

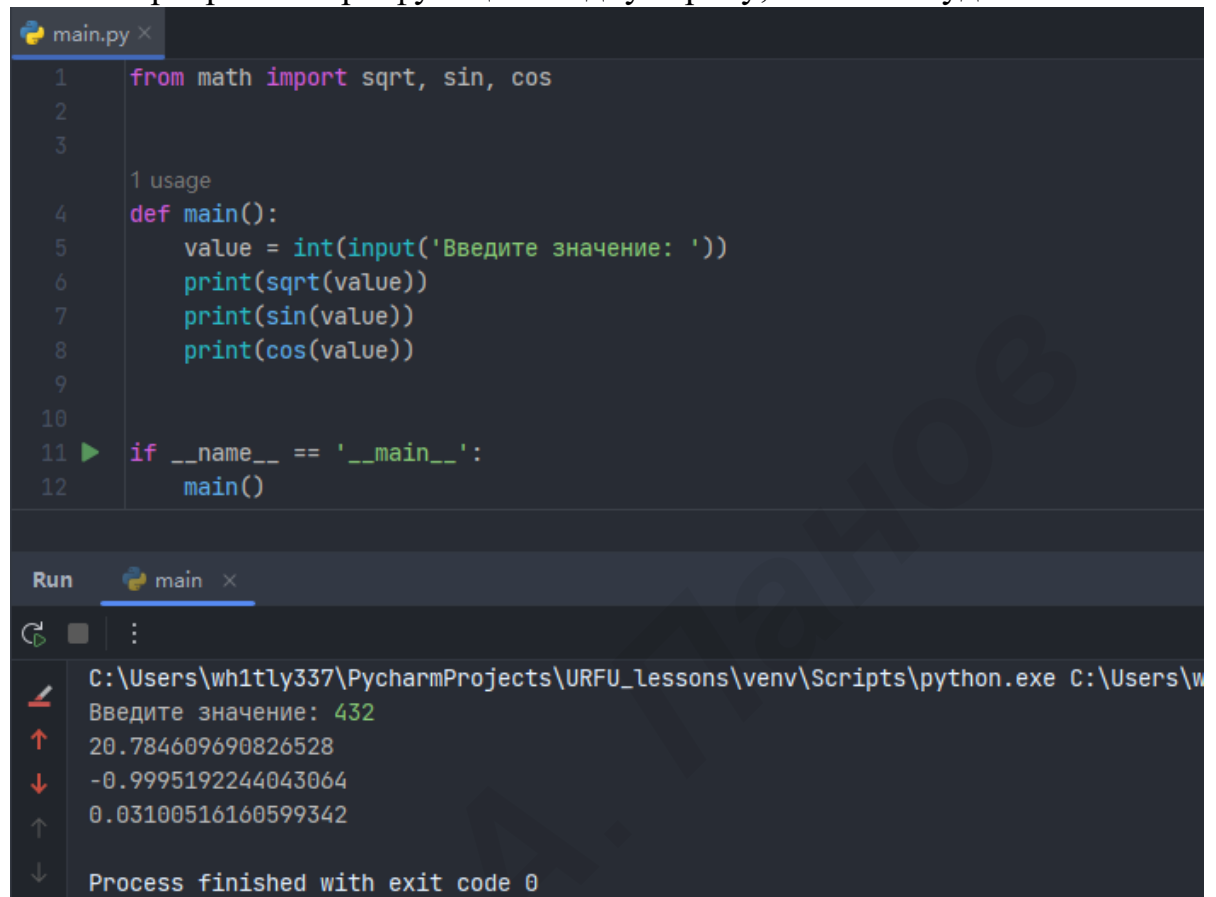
Run main x
C:\Users\wh1tly337\PycharmProjects\URFU_lessons\venv\Scripts\python.exe C:\Users\w
Введите значение: 432
20.784609690826528
-0.9995192244043064
0.03100516160599342
Process finished with exit code 0
```

На первом скриншоте мы просто импортировали модуль `math` целиком и вызвали его длинным способом через `math.название_функции`.

Также импорт стандартного модуля в python возможно осуществить и другими способами, которые будут выполнять ту же самую функцию, но синтаксис будет немного отличаться.

На втором скриншоте из модуля `math` мы загрузили в программу только 3 необходимые функции и обращались к ним так, будто они

находятся у нас в файле просто через их название. Также замечу что мы импортировали три функции в одну строку, что очень удобно.



The screenshot displays the PyCharm IDE interface. The top pane shows a file named `main.py` with the following Python code:

```
1 from math import sqrt, sin, cos
2
3
4 1 usage
5 def main():
6     value = int(input('Введите значение: '))
7     print(sqrt(value))
8     print(sin(value))
9     print(cos(value))
10
11 if __name__ == '__main__':
12     main()
```

The bottom pane shows the 'Run' output for the `main` script. It displays the command executed, the user input, and the resulting output values for the square root, sine, and cosine functions.

```
Run main x
C:\Users\whitly337\PycharmProjects\URFU_lessons\venv\Scripts\python.exe C:\Users\w
Введите значение: 432
20.784609690826528
-0.9995192244043064
0.03100516160599342
Process finished with exit code 0
```

На третьем скриншоте мы импортировали модуль `math` и при помощи оператора `*` загрузили все его функции. По большому счету мы сделали то же самое что и на первом скриншоте, но у нас только поменялся синтаксис вызова этих функций, он стал похож на вызов со второго скриншота.


```
main.py x
1  from math import *
2
3
4  1 usage
5  def main():
6      value = int(input('Введите значение: '))
7      print(sqrt(value))
8      print(sin(value))
9      print(cos(value))
10
11  if __name__ == '__main__':
12      main()

Run  main x
C:\Users\whitly337\PycharmProjects\URFU_lessons\venv\Scripts\python.exe C:\Users\w
Введите значение: 432
20.784609690826528
-0.9995192244043064
0.03100516160599342
Process finished with exit code 0
```

- 9) Напишите программу, которая будет рассчитывать какой день недели будет через n-ное количество дней, которые укажет пользователь.

```
main.py ×
1 from datetime import datetime as dt # dt теперь равно datetime
2 from datetime import timedelta as td # td теперь равно timedelta
3 # Мы воспользовались возможностью Python создавать псевдонимы
4
5
6 1 usage
7 def main():
8     print(
9         f"Сегодня {dt.today().date()}. "
10        f"День недели - {dt.today().isoweekday()}"
11    )
12    n = int(input('Введите количество дней: '))
13    today = dt.today()
14    result = today + td(days=n)
15    print(
16        f"Через {n} дней будет {result.date()}. "
17        f"День недели - {result.isoweekday()}"
18    )
19
20 if __name__ == '__main__':
21     main()
```

Run main ×



С:\Users\wh1tly337\PycharmProjects\URFU_lessons\venv\Scripts\python.exe C:\Users\w
Сегодня 2023-05-12. День недели - 5
Введите количество дней: 543
Через 543 дней будет 2024-11-05. День недели - 2
Process finished with exit code 0





В результате день недели указан в виде цифры, где 1 = понедельник, 2 = вторник, 3 = среда и так далее.

- 10) Напишите программу с использованием глобальных переменных, которая будет считать площадь треугольника или прямоугольника в зависимости от того, что выберет пользователь. Получение всей необходимой информации реализовать через `input()`, а подсчет площадей выполнить при помощи функций. Результатом программы будет число, равное площади, необходимой фигуры.

```
main.py x
1  global result
2
3
4  1 usage
5  def rectangle():
6      a = float(input("Ширина: "))
7      b = float(input("Высота: "))
8      global result
9      result = a * b
10
11  1 usage
12  def triangle():
13      a = float(input("Основание: "))
14      h = float(input("Высота: "))
15      global result
16      result = 0.5 * a * h
17
18  figure = input("1-прямоугольник, 2-треугольник: ")
19
20  if figure == '1':
21      rectangle()
22  elif figure == '2':
23      triangle()
24
25  print(f"Площадь: {result}")
```

Run main x

 /usr/local/bin/python3.11 /Users/user/PycharmProjects/URFU_lessons/main.py
1-прямоугольник, 2-треугольник: 2
 Основание: 5
 Высота: 6
 Площадь: 15.0