

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №3
дисциплины «Программирование на Python»

Выполнил:
Данилецкий Дмитрий Витальевич
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем», очная
форма обучения

(подпись)

Руководитель практики:
Воронкин Р. А., канд. технических
наук, доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

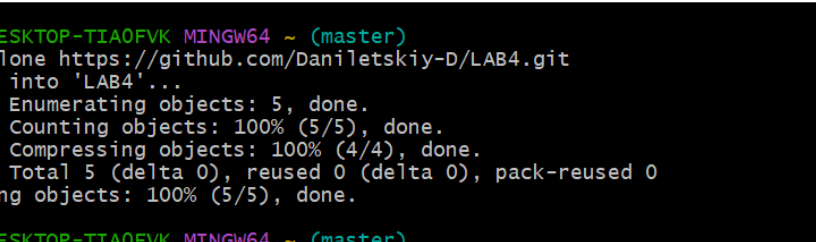
Тема: Основы языка Python

Цель: исследование процесса установки и базовых возможностей языка Python версии 3.x.

Краткие теоритические сведения

Ход работы

1. Создал общедоступный репозиторий на GitHub, в котором используется лицензия MIT и язык программирования Python. Выполнил клонирование созданного репозитория.



The screenshot shows a Windows terminal window with the title bar "MINGW64: c:/Users/slime/LAB4". The terminal content is as follows:

```
slime@DESKTOP-TIA0FVK MINGW64 ~ (master)
$ git clone https://github.com/Daniletskiy-D/LAB4.git
Cloning into 'LAB4'...
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.

slime@DESKTOP-TIA0FVK MINGW64 ~ (master)
$ cd LAB4

slime@DESKTOP-TIA0FVK MINGW64 ~/LAB4 (main)
$ |
```

Рисунок 1. Клонирование репозитория

2. Дополнил файл .gitignore необходимыми правилами.

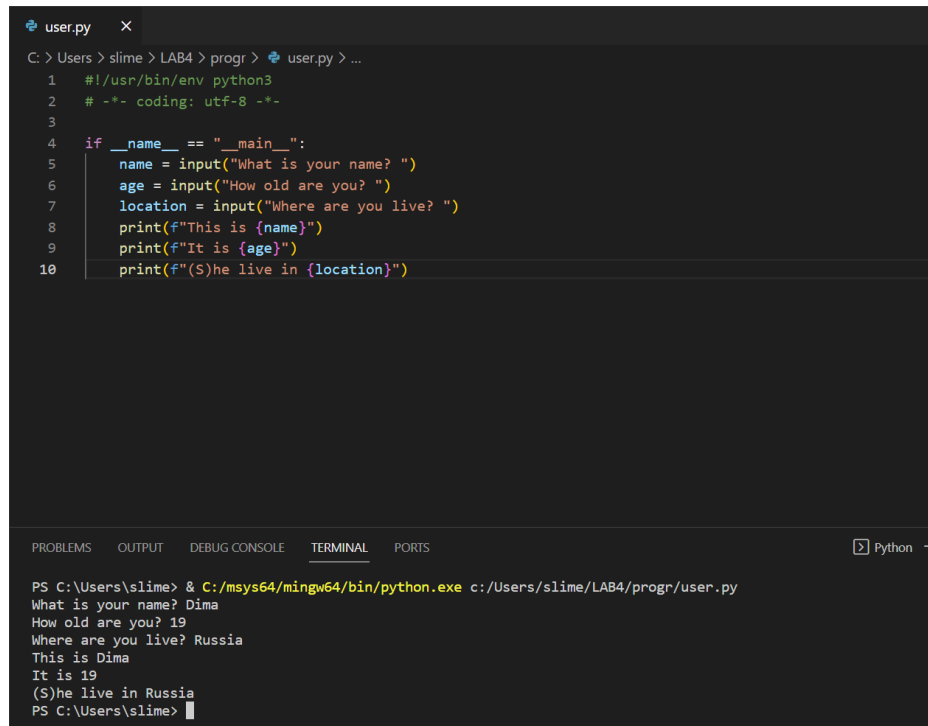
```

# * ignore - бинарии
#
# Sub: Папка Формат Вид Справка
# # Byte-compiled / optimized / DLL files
# _pycache_
# *.py[co]
# *.py.class
#
# # C extensions
# *.so
#
# # Distribution / packaging
# .Python
# build/
# develop-eggs/
# dist/
# downloads/
# eggs/
# .eggs/
# lib/
# lib64/
# parts/
# sdist/
# var/
# wheels/
# share/python-wheels/
# *.egg-info/
# .installed.cfg
# *.egg
# MANIFEST
#
# # PyInstaller
# # Usually these files are written by a python script from a template
# # before PyInstaller builds the exe, so as to inject date/other infos into it.
# *.manifest
# *.spec
#
# # Installer logs
# pip-log.txt
# pip-delete-this-directory.txt
#
# # Unit test / coverage reports
# htmlcov/
# .tox/
# .nox/
# .coverage
# .coverage.*
# .cache
# nosetests.xml

```

Рисунок 2. Дополнение .gitignore

3. Написал программу (файл user.py), которая запрашивает у пользователя его имя, возраст и место жительства, а после этого выводит данные, введенные пользователем.



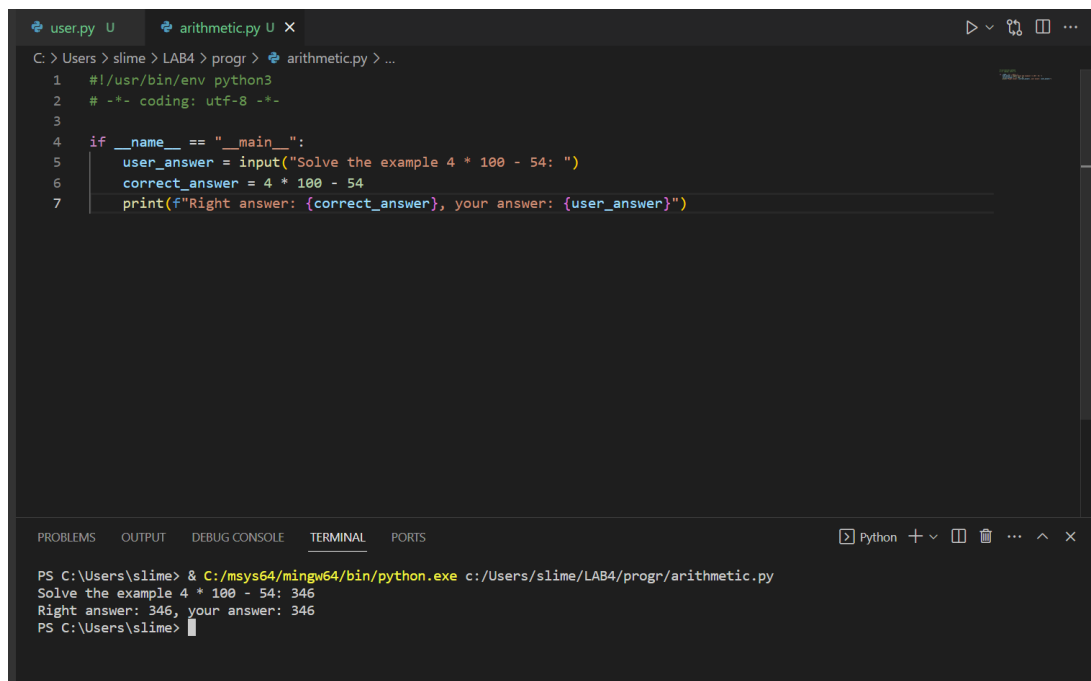
```
user.py X
C: > Users > slime > LAB4 > progr > user.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == "__main__":
5      name = input("What is your name? ")
6      age = input("How old are you? ")
7      location = input("Where are you live? ")
8      print(f"This is {name}")
9      print(f"It is {age}")
10     print(f"(S)he live in {location}")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python +

PS C:\Users\slime> & C:/msys64/mingw64/bin/python.exe c:/Users/slme/LAB4/progr/user.py
What is your name? Dima
How old are you? 19
Where are you live? Russia
This is Dima
It is 19
(S)he live in Russia
PS C:\Users\slime>
```

Рисунок 3. Результат работы программы user.py

4. Написал программу (файл arithmetic.py), которая предлагает пользователю решить пример $4 * 100 - 54$. Потом выводит на экран правильный ответ и ответ пользователя.



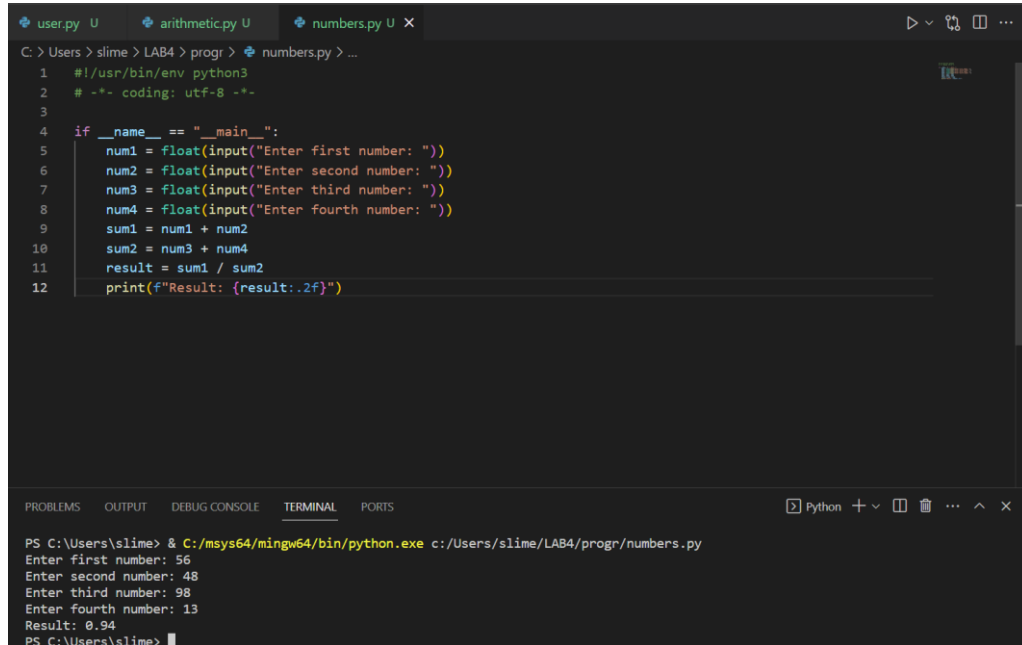
```
user.py U arithmetic.py X
C: > Users > slime > LAB4 > progr > arithmetic.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == "__main__":
5      user_answer = input("Solve the example 4 * 100 - 54: ")
6      correct_answer = 4 * 100 - 54
7      print(f"Right answer: {correct_answer}, your answer: {user_answer}")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + - - - - -

PS C:\Users\slime> & C:/msys64/mingw64/bin/python.exe c:/Users/slme/LAB4/progr/arithmetic.py
Solve the example 4 * 100 - 54: 346
Right answer: 346, your answer: 346
PS C:\Users\slime>
```

Рисунок 4. Результат работы программы arithmetic.py

5. Написал программу, которая запрашивает у пользователя четыре числа (файл numbers.py), отдельно складывает первые два и отдельно вторые два, далее делит первую сумму на вторую и выводит результат на экран так, чтобы ответ содержал две цифры после запятой.

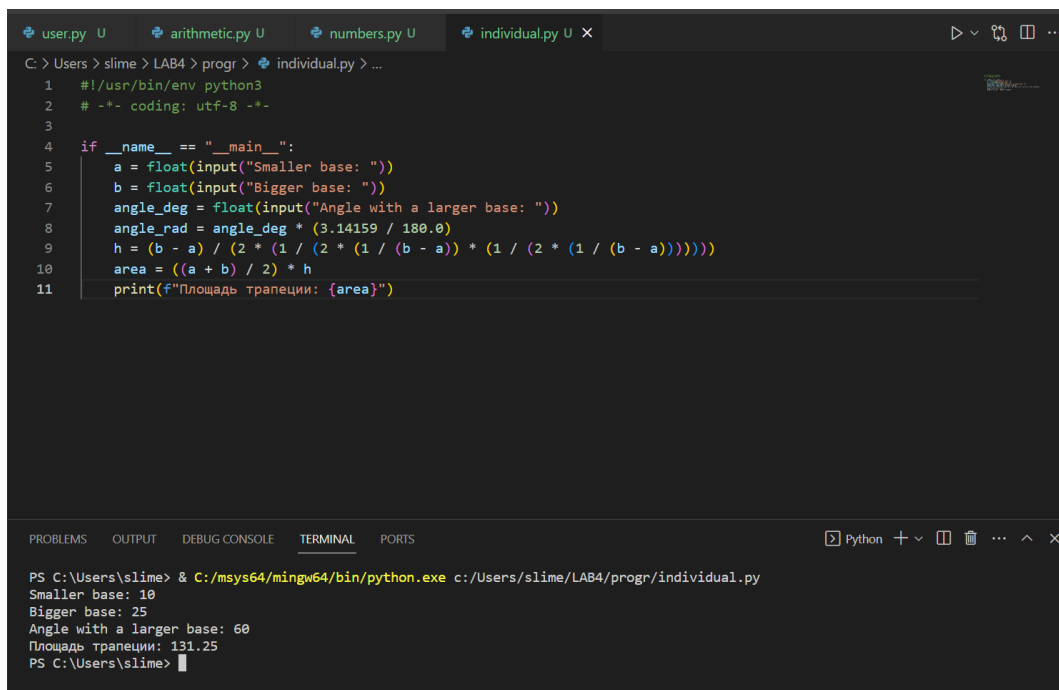


```
user.py U arithmetic.py U numbers.py X
C:\Users\slime> LAB4> progr> numbers.py ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == "__main__":
5      num1 = float(input("Enter first number: "))
6      num2 = float(input("Enter second number: "))
7      num3 = float(input("Enter third number: "))
8      num4 = float(input("Enter fourth number: "))
9      sum1 = num1 + num2
10     sum2 = num3 + num4
11     result = sum1 / sum2
12     print(f"Result: {result:.2f}")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python + - - - - -
PS C:\Users\slime> & C:/msys64/mingw64/bin/python.exe c:/Users/slme/LAB4/progr/numbers.py
Enter first number: 56
Enter second number: 48
Enter third number: 98
Enter fourth number: 13
Result: 0.94
PS C:\Users\slime>
```

Рисунок 5. Результат работы программы numbers.py

6. Написал программу (файл individual.py) для решения индивидуального задания. Вариант 8: даны основания равнобедренной трапеции и угол при большем основании, найти площадь трапеции.

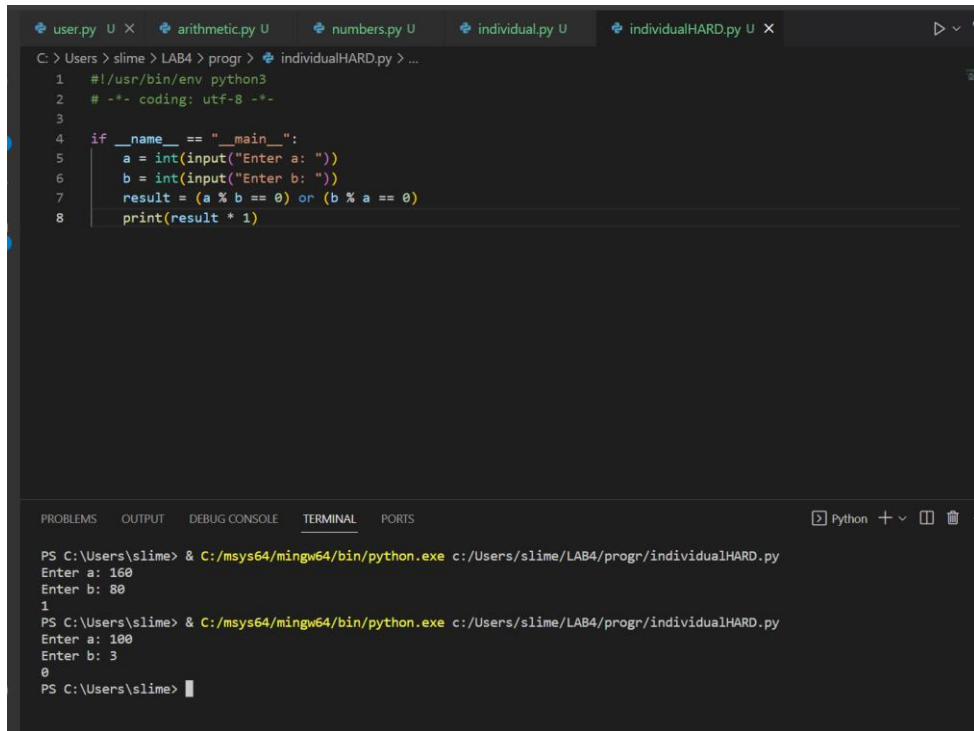


```
user.py U arithmetic.py U numbers.py U individual.py X
C:\Users\slime> LAB4> progr> individual.py ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == "__main__":
5      a = float(input("Smaller base: "))
6      b = float(input("Bigger base: "))
7      angle_deg = float(input("Angle with a larger base: "))
8      angle_rad = angle_deg * (3.14159 / 180.0)
9      h = (b - a) / (2 * (1 / (2 * (1 / (b - a)) * (1 / (2 * (1 / (b - a)))))))
10     area = ((a + b) / 2) * h
11     print(f"Площадь трапеции: {area}")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python + - - - - -
PS C:\Users\slime> & C:/msys64/mingw64/bin/python.exe c:/Users/slme/LAB4/progr/individual.py
Smaller base: 10
Bigger base: 25
Angle with a larger base: 60
Площадь трапеции: 131.25
PS C:\Users\slime>
```

Рисунок 6. Результат работы программы individual.py

7. Написал программу (файл individualHARD.py) для решения индивидуального задания повышенной сложности. Вариант 8: Даны два целых числа a и b . Если a делится на b или b делится на a , то вывести 1, иначе любое другое число.



```
user.py U x arithmetic.py U numbers.py U individual.py U individualHARD.py U x
C:\Users\slime> LAB4> progr> individualHARD.py> ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == "__main__":
5      a = int(input("Enter a: "))
6      b = int(input("Enter b: "))
7      result = (a % b == 0) or (b % a == 0)
8      print(result + 1)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python + v [ ] [ ] [ ]

PS C:\Users\slime> & C:/msys64/mingw64/bin/python.exe c:/Users/slme/LAB4/progr/individualHARD.py
Enter a: 160
Enter b: 80
1
PS C:\Users\slime> & C:/msys64/mingw64/bin/python.exe c:/Users/slme/LAB4/progr/individualHARD.py
Enter a: 100
Enter b: 3
0
PS C:\Users\slime>
```

Рисунок 7. Результат работы программы individualHARD.py

8. Выполнил коммит файлов в репозиторий git в ветку для разработки, затем выполнил слияние ветки для разработки с веткой main и отправил сделанные изменения на сервер GitHub.



```
MINGW64/c/Users/slme/LAB4
slime@DESKTOP-TIA0FVK MINGW64 ~/LAB4 (main)
$ git push
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Daniletskiy-D/LAB4.git
f7a2806..4722fa3 main -> main

slime@DESKTOP-TIA0FVK MINGW64 ~/LAB4 (main)
$ git merge develop
Updating 4722fa3..e93f372
Fast-forward
 progr/arithmetic.py | 7 ++++++
 progr/individual.py | 11 ++++++++
 progr/individualHARD.py | 8 ++++++
 progr/numbers.py | 12 ++++++++
 progr/user.py | 10 ++++++++
 5 files changed, 48 insertions(+)
 create mode 100644 progr/arithmetic.py
 create mode 100644 progr/individual.py
 create mode 100644 progr/individualHARD.py
 create mode 100644 progr/numbers.py
 create mode 100644 progr/user.py

slime@DESKTOP-TIA0FVK MINGW64 ~/LAB4 (main)
$
```

Рисунок 8. Слияние веток

Контрольные вопросы

1. Опишите основные этапы установки Python в Windows и Linux.

В Windows: Скачать установочный файл Python с официального сайта (python.org). Запустить установщик и следовать инструкциям. Выбрать опцию "Add Python to PATH" (добавить Python в переменную среды PATH) для удобства использования Python из командной строки. Завершить процесс установки.

В Linux: Многие дистрибутивы Linux уже имеют Python предустановленным. В противном случае, можно установить его с помощью пакетного менеджера вашего дистрибутива, например, в Ubuntu: `sudo apt-get install python3`. После установки можно проверить версию с помощью команды `python3 --version`.

2. В чем отличие пакета Anaconda от пакета Python, скачиваемого с официального сайта?

Anaconda - это дистрибуция Python, предназначенная для научных вычислений и анализа данных. Основное отличие заключается в том, что Anaconda включает в себя множество предустановленных библиотек и инструментов, таких как NumPy, Pandas, Matplotlib, Jupyter и многие другие, что делает ее идеальным выбором для работы в области анализа данных и машинного обучения. Стандартный пакет Python с официального сайта включает только базовые библиотеки.

3. Как осуществить проверку работоспособности пакета Anaconda?

Для проверки работоспособности Anaconda можно запустить интерактивную оболочку IPython или Jupyter Notebook. Также можно создать новое окружение и установить несколько библиотек для проверки.

4. Как задать используемый интерпретатор языка Python в IDE PyCharm?

В PyCharm можно задать интерпретатор Python в настройках проекта или в глобальных настройках IDE. Для этого перейдите в "File" -> "Settings"

(или "Preferences" на macOS) -> "Project: [имя проекта]" -> "Python Interpreter" и выберите нужный интерпретатор Python.

5. Как осуществить запуск программы с помощью IDE PyCharm?

Для запуска программы в PyCharm можно нажать кнопку "Run" (Запустить) или "Debug" (Отладка) в верхней панели. Также можно использовать комбинации клавиш, например, Shift + F10 для запуска.

6. В чем суть интерактивного и пакетного режимов работы Python?

Интерактивный режим позволяет вводить команды Python построчно и немедленно видеть результат. Пакетный режим используется для выполнения скриптов и программ, которые выполняются целиком.

7. Почему язык программирования Python называется языком динамической типизации?

Язык Python называется динамическим из-за того, что типы данных переменных определяются автоматически во время выполнения программы, а не во время компиляции.

8. Какие существуют основные типы в языке программирования Python?

Основные типы данных в Python включают числа (int, float), строки (str), списки (list), кортежи (tuple), множества (set), словари (dict), булевы значения (bool), и многое другое.

9. Как создаются объекты в памяти? Каково их устройство? В чем заключается процесс объявления новых переменных и работа операции присваивания?

Переменные - это ссылки на объекты. Процесс объявления переменных заключается в присвоении им значений, и Python автоматически выделяет память для хранения объектов.

10. Как получить список ключевых слов в Python?

Получение списка ключевых слов в Python можно сделать с помощью модуля keyword. Используйте import keyword и keyword.kwlist для получения списка ключевых слов.

11. Каково назначение функций `id()` и `type()`?

Функция `id()` возвращает уникальный идентификатор объекта в памяти, а функция `type()` возвращает тип объекта.

12. Что такое изменяемые и неизменяемые типы в Python.

Изменяемые типы данных могут быть изменены после создания, например, списки. Неизменяемые типы данных, такие как кортежи и строки, не могут быть изменены после создания.

13. Чем отличаются операции деления и целочисленного деления?

Операция деления (`/`) возвращает результат в виде числа с плавающей точкой, а операция целочисленного деления (`//`) возвращает результат в виде целого числа.

14. Какие имеются средства в языке Python для работы с комплексными числами?

Для работы с комплексными числами в Python используется встроенный тип `complex`.

15. Каково назначение и основные функции библиотеки (модуля) `math`? По аналогии с модулем `math` изучите самостоятельно назначение и основные функции модуля `cmath`.

Библиотека (модуль) `math` в Python предоставляет функции и константы для выполнения математических операций и вычислений. Основное назначение и функции библиотеки `math` включают в себя:

Вычисления математических функций: `math` предоставляет функции для выполнения различных математических операций, таких как корень, логарифмы, тригонометрические функции и др.

Константы: `math` содержит константы, такие как число π (π) и экспонента (e), которые можно использовать в вычислениях.

Округление и модуль: `math` предоставляет функции для округления чисел, нахождения модуля числа, а также другие функции для работы с числами.

Тригонометрические функции: `math` включает в себя тригонометрические функции, такие как синус, косинус, тангенс и другие, которые позволяют выполнять вычисления связанные с углами.

Экспоненциальные и логарифмические функции: `math` предоставляет функции для работы с экспоненциальными и логарифмическими вычислениями, такие как возведение в степень, натуральный логарифм и другие.

Библиотека `math` является полезным инструментом для выполнения разнообразных математических операций в Python, и она широко используется при решении математических задач, научных вычислений и инженерных задач.

Модуль `cmath` предоставляет аналогичные функции для комплексных чисел.

16. Каково назначение именных параметров `sep` и `end` в функции `print()`?

Параметры `sep` и `end` в функции `print()` используются для настройки разделителей между значениями и окончания вывода.

17. Каково назначение метода `format()`? Какие еще существуют средства для форматирования строк в Python? Примечание: в дополнение к рассмотренным средствам изучите самостоятельно работу с f-строками в Python.

Метод `format()` используется для форматирования строк, позволяя вставлять значения в строку. Также в Python есть f-строки для удобного форматирования строк с использованием выражений.

18. Каким образом осуществить ввод с консоли значения целочисленной и вещественной переменной в языке Python?

Для ввода значений с консоли в Python используются функции `input()` для строк, `int(input())` для целых чисел и `float(input())` для вещественных чисел.

Вывод: В ходе выполнения лабораторной работы был успешно установлен и изучен на базовом уровне Python 3.x. Были приобретены базовые навыки программирования на этом языке, что дало возможность создавать и решать задачи с использованием Python. Кроме того была освоена модель git-flow.