

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

Институт цифрового развития
Кафедра инфокоммуникаций

«Лабораторная работа 2.1 Основы языка Python»

ОТЧЕТ
по лабораторной работе №4
дисциплины
«Основы программной инженерии»

Выполнил:

Луценко Дмитрий Андреевич
2 курс, группа ПИЖ-б-о-21-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Проверил:

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2022 г.

Лабораторная работа 2.1 Основы языка Python.

Цель работы: исследование процесса установки и базовых возможностей языка Python

```
Командная строка
To https://github.com/DfoorS/Varick.git
* [new branch] develop -> develop

E:\Github\Varick>git push origin feature
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature' on GitHub by visiting:
remote:   https://github.com/DfoorS/Varick/pull/new/feature
remote:
To https://github.com/DfoorS/Varick.git
* [new branch] feature -> feature

E:\Github\Varick>git checkout develop
Switched to branch 'develop'

E:\Github\Varick>git status
On branch develop
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .idea/

nothing added to commit but untracked files present (use "git add" to track)

E:\Github\Varick>git status
On branch develop
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    arithmetic.py
    individual.py
    individualplus.py
    numbers.py
    user.py

nothing added to commit but untracked files present (use "git add" to track)

E:\Github\Varick>git add .

E:\Github\Varick>git commit -m "сделана все файлы"
[develop 1cdc2c4] сделана все файлы
5 files changed, 30 insertions(+)
create mode 100644 arithmetic.py
create mode 100644 individual.py
create mode 100644 individualplus.py
create mode 100644 numbers.py
create mode 100644 user.py

E:\Github\Varick>git push
fatal: The current branch develop has no upstream branch.
To push the current branch and set the remote as upstream, use
```

Рисунок X – коммит файлов на ветке develop

```
Командная строка
To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.

E:\Github\Varick>git push --set-upstream origin develop
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 1.21 KiB | 622.00 KiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/DfoorS/Varick.git
  8cf9aff..1cdc2c4 develop -> develop
branch 'develop' set up to track 'origin/develop'.

E:\Github\Varick>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

E:\Github\Varick>git merge develop
Updating 8cf9aff..1cdc2c4
Fast-forward
 arithmetic.py | 4 +++
 individual.py | 4 +++
 individualplus.py | 7 +++++
 numbers.py | 7 +++++
 user.py | 8 ++++++
 5 files changed, 30 insertions(+)
 create mode 100644 arithmetic.py
 create mode 100644 individual.py
 create mode 100644 individualplus.py
 create mode 100644 numbers.py
 create mode 100644 user.py

E:\Github\Varick>git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

E:\Github\Varick>git push
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/DfoorS/Varick.git
  8cf9aff..1cdc2c4 main -> main

E:\Github\Varick>
```

Рисунок X – слияние веток main и develop

Вывод: были исследованы процессы установки и базовых возможностей языка Python

Ответы на вопросы:

1. Основные этапы установки Python на Windows: скачать дистрибутив, запустить установщик, выбрать способ установки, отметить необходимые опции(при ручной установке), выбрать каталог установки. Основные этапы установки Python на Linux: обычно он уже установлен, это можно проверить через консоль. Если он не установлен, то можно либо собрать его из исходников, либо скачать из репозитория, введя команду в терминале
2. Anaconda позволяет изолировать окружение проекта от системной версии Python, который критически необходим для работы системы. Также conda позволяет без проблем переносить окружение с одной машины на другую. Кроме того, если вы что-то сломаете, то с Anaconda вы всегда сможете откатиться на более старую версию окружения. Конечно, если вы позаботитесь о регулярных бэкапах. С системной версией Python это гораздо сложнее и может потребовать переустановки системы.
3. Для выполнения проверки работоспособности Anaconda необходимо вначале запустить командный процессор с поддержкой виртуальных окружений Anaconda. В Windows это можно сделать выбрав следующий пункт главного меню системы Пуск Anaconda3 (64-bit)
Anaconda Prompt. В появившейся командной строке необходимо ввести jupyter notebook
4. Интерпретатор можно задать при создании проекта
5. Нажать на зеленый треугольник в верхней панели
6. В интерактивном режиме команды вводятся в консоль и интерпретатор её выполнит. В пакетном режиме интерпретируются файлы с исходным кодом

7. Python называется языком динамической типизации, потому что тип переменной определяется на этапе выполнения программы
8. Основные типы данных в языке Python:
- None (неопределенное значение переменной)
 - Логические переменные (Boolean Type)
 - Числа (Numeric Type)
 - int – целое число
 - float – число с плавающей точкой
 - complex – комплексное число
 - Списки (Sequence Type)
 - list – список
 - tuple – кортеж
 - range – диапазон
 - Строки (Text Sequence Type)
 - Str
 - Бинарные списки (Binary Sequence Types)
 - bytes – байты
 - bytearray – массивы байт
 - memoryview – специальные объекты для доступа к внутренним данным объекта через protocol buffer
 - Множества (Set Types)
 - set – множество
 - frozenset – неизменяемое множество
 - Словари (Mapping Types)
 - dict – словарь
9. При инициализации переменной, на уровне интерпретатора, происходит следующее:
- создается целочисленный объект 5 (можно представить, что в этот момент создается ячейка и 5 кладется в эту ячейку);
 - данный объект имеет некоторый идентификатор, значение: 5, и тип: целое число;
 - посредством оператора “=” создается ссылка между переменной b и целочисленным объектом 5 (переменная b ссылается на объект 5).
10. Для получения списка ключевых слов нужно подключить модуль keyword и воспользоваться командой keyword.kwlist.
- ```
import keyword
print("Python keywords: ", keyword.kwlist)
```

11. Функция `id()` нужна для того чтобы посмотреть на объект с каким идентификатором ссылается переменная. Функция `type()` нужна для определения типа переменной
12. К неизменяемым типам относятся: целые числа (`int`), числа с плавающей точкой (`float`), комплексные числа (`complex`), логические переменные (`bool`), кортежи (`tuple`), строки (`str`) и неизменяемые множества (`frozen set`). К изменяемым (`mutable`) типам относятся: списки (`list`), множества (`set`), словари (`dict`).
- Неизменяемость означает, что объект больше не изменится.
13. Деление – обычная математическая операция, обозначаемая знаком «/», а целочисленное обозначается «//» и в результат будет записана только целая часть.
14. Для работы с комплексными числами используется модуль `cmath`
15. Модуль `math` предоставляет функционал для работы с числами.

`math.ceil(X)` – округление до ближайшего большего числа.

`math.factorial(X)` - факториал числа  $X$ .

`math.floor(X)` - округление вниз.

`math.isinf(X)` - является ли  $X$  бесконечностью.

`math.modf(X)` - возвращает дробную и целую часть числа  $X$ . Оба числа имеют тот же знак, что и  $X$ .

`math.trunc(X)` - усекает значение  $X$  до целого.

`math.exp(X)` -  $e^X$ .

`math.expm1(X)` -  $e^X - 1$ . При  $X \rightarrow 0$  точнее, чем `math.exp(X)-1`.

`math.log(X, [base])` - логарифм  $X$  по основанию `base`. Если `base` не указан, вычисляется натуральный логарифм.

`math.log1p(X)` - натуральный логарифм  $(1 + X)$ . При  $X \rightarrow 0$  точнее, чем `math.log(1+X)`.

`math.log10(X)` - логарифм  $X$  по основанию 10.

`math.log2(X)` - логарифм  $X$  по основанию 2.

`math.pow(X, Y)` -  $X^Y$ .

`math.sqrt(X)` - квадратный корень из  $X$ .

`math.acos(X)` - арккосинус  $X$ . В радианах.

`math.asin(X)` - арксинус  $X$ . В радианах.

`math.atan(X)` - арктангенс  $X$ . В радианах.

`math.cos(X)` - косинус  $X$  ( $X$  указывается в радианах).

`math.sin(X)` - синус  $X$  ( $X$  указывается в радианах).

`math.tan(X)` - тангенс  $X$  ( $X$  указывается в радианах).

`math.pi` -  $\pi = 3,1415926\dots$

`math.e` -  $e = 2,718281\dots$

Для работы с комплексными числами модуль `cmath` предоставляет следующие функции:

`cmath.phase(x)` - возвращает фазу комплексного числа (её ещё называют аргументом). Эквивалентно `math.atan2(x.imag, x.real)`. Результат лежит в промежутке  $[-\pi, \pi]$ .

Получить модуль комплексного числа можно с помощью встроенной функции `abs()`.

`cmath.polar(x)` - преобразование к полярным координатам. Возвращает пару  $(r, \phi)$ .

`cmath.rect(r, phi)` - преобразование из полярных координат.

`cmath.exp(x)` -  $e^x$ .

`cmath.log(x[, base])` - логарифм  $x$  по основанию `base`. Если `base` не указан, возвращается натуральный логарифм.

`cmath.log10(x)` - десятичный логарифм.

`cmath.sqrt(x)` - квадратный корень из  $x$ .

`cmath.acos(x)` - арккосинус  $x$ .

`cmath.asin(x)` - арксинус  $x$ .

`cmath.atan(x)` - арктангенс  $x$ .

`cmath.cos(x)` - косинус  $x$ .

`cmath.sin(x)` - синус  $x$ .

`cmath.tan(x)` - тангенс  $x$ .

`cmath.pi` -  $\pi$ .

`cmath.e` -  $e$ .

16. `sep` – разделитель между объектами, `end` – символ в конце строки, по умолчанию `\n`
17. Метод `format()` нужен для форматирования строк. Форматирование так же можно сделать с помощью оператора `%`. F-строки – это форматированная строка с префиксом `'f'`, которая содержит выражения внутри фигурных скобок `{ }`
18. Для целочисленной `int(input())`; для вещественной `float(input())`