

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ  
ФЕДЕРАЦИИ**  
**Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**

**Институт цифрового развития**

**ОТЧЁТ**

**по лабораторной работе №3**

Дисциплина: «Основы кроссплатформенного программирования»

Тема: «Условные операторы и циклы в языке Python»

Выполнил студент группы

ИВТ-б-о-21-1

Криворот Владимир Геннадьевич

« »\_\_\_\_\_20\_\_г.

Подпись студента\_\_\_\_\_

Работа защищена « »\_\_\_\_\_20\_\_г.

Проверил доцент

Кафедры инфокоммуникаций, старший  
преподаватель

Воронкин Р.А.

\_\_\_\_\_  
(подпись)

Ставрополь 2022

## Ход работы:

```
131 # Created by https://www.toptal.com/developers/gitignore/api/python,pycharm
132 # Edit at https://www.toptal.com/developers/gitignore?templates=python,pycharm
133
134 ### PyCharm ###
135 # Covers JetBrains IDEs: IntelliJ, RubyMine, PhpStorm, AppCode, PyCharm, CLion, Android Studio, WebStorm and Rider
136 # Reference: https://intellij-support.jetbrains.com/hc/en-us/articles/206544839
137
138 # User-specific stuff
139 .idea/**/workspace.xml
140 .idea/**/tasks.xml
141 .idea/**/usage.statistics.xml
142 .idea/**/dictionaries
143 .idea/**/shelf
144
145 # AWS User-specific
146 .idea/**/aws.xml
147
148 # Generated files
149 .idea/**/contentModel.xml
150
151 # Sensitive or high-churn files
152 .idea/**/dataSources/
153 .idea/**/dataSources.ids
154 .idea/**/dataSources.local.xml
155 .idea/**/sqlDataSources.xml
156 .idea/**/dynamic.xml
157 .idea/**/uiDesigner.xml
158 .idea/**/dbnavigator.xml
159
160 # Gradle
161 .idea/**/gradle.xml
162 .idea/**/libraries
163
164 # Gradle and Maven with auto-import
165 # When using Gradle or Maven with auto-import, you should exclude module files,
166 # since they will be recreated, and may cause churn. Uncomment if using
```

Рисунок 1. Добавление правил в .gitignore

```
D:\git\Laba2_1>git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main] main
Branch name for "next release" development: [develop] developer

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [D:/git/Laba2_1/.git/hooks]

D:\git\Laba2_1>
```

Рисунок 2. Организация репозитория согласно модели ветвления git-flow

```
name = str(input("Whats your name - "))
age = int(input("Whats your age - "))
place = str(input("Where are you from - "))
print("This is ", name)
print("It is ", age)
print("(S)he live in ", place)
```

Рисунок 3. Задача 1

```
str = '4 * 100 - 54'
print(str)
print("Now solve it - ")
answer = int(input())
print("Correct answer - ", 4 * 100 - 54, "and your answer - ", answer)
```

Рисунок 4. Задача 2

```
print("Print 4 numbers - ")
a = int(input())
b = int(input())
c = int(input())
d = int(input())
first = a + b
second = c + d
f = first / second
g = round(f, 2)
print(g)
```

Рисунок 5. Задача 3

```
# Известно значение температуры по шкале Цельсия. Найти соответствующее значение
# температуры по шкале:
# Фаренгейта;
# Кельвина.
# Для пересчета по шкале Фаренгейта необходимо исходное значение температуры умножить
# на 1,8 и к результату прибавить 32, а по шкале Кельвина абсолютное значение нуля
# соответствует -273,15 градуса по шкале Цельсия.

cels = int(input("Print cels - "))
Far = (cels * 1.8) + 32
Kel = -273.15 + cels
print(cels, Far, Kel)
```

Рисунок 6. Индивидуальная задача

```

D:\git\Laba2_1>git merge develop
merge: develop - not something we can merge

D:\git\Laba2_1>git merge developer
Updating 5cac2d5..8e4ce4a
Fast-forward
 .idea/.gitignore           | 3 +++
 .idea/Laba2_1.iml          | 10 ++++++++
 .idea/inspectionProfiles/profiles_settings.xml | 6 +++++
 .idea/misc.xml             | 4 ++++
 .idea/modules.xml          | 8 ++++++
 .idea/vcs.xml              | 6 +++++
 arithmetic.py              | 5 +++++
 individual.py              | 12 ++++++++
 numbers.py                 | 10 ++++++++
 user.py                   | 6 +++++
10 files changed, 70 insertions(+)
create mode 100644 .idea/.gitignore
create mode 100644 .idea/Laba2_1.iml
create mode 100644 .idea/inspectionProfiles/profiles_settings.xml
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/vcs.xml
create mode 100644 arithmetic.py
create mode 100644 individual.py
create mode 100644 numbers.py
create mode 100644 user.py

```

Рисунок 7. Слияние веток









 .idea	Решённые задачи	19 minutes ago
 .gitignore	Update .gitignore	1 hour ago
 LICENSE	Initial commit	1 hour ago
 README.md	Initial commit	1 hour ago
 arithmetic.py	Решённые задачи	19 minutes ago
 individual.py	Решённые задачи	19 minutes ago
 numbers.py	Решённые задачи	19 minutes ago
 user.py	Решённые задачи	19 minutes ago

Рисунок 8. Изменения в удаленном репозитории

```
# 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. Условные  
# операторы и операторы цикла не использовать.  
  
import random  
a = int(input("Print a -"))  
b = int(input("Print b - "))  
if a % b == 0 or b % a == 0:  
    print(1)  
else:  
    print(random.randint(-100, 100))
```

else

hard ×

D:\git\Laba2\_1\venv\Scripts\python.exe D:/git/Laba2\_1/hard.py  
Print a -2  
Print b - 4  
1  
  
Process finished with exit code 0

Рисунок 9. Задание повышенной сложности

### Контрольные вопросы:

#### 1. Для чего нужны диаграммы деятельности UML?

Позволяет наглядно визуализировать алгоритм программы.

#### 2. Что такое состояние действия и состояние деятельности?

Состояние действия - частный вид состояния деятельности, а конкретнее – такое состояние, которое не может быть подвергнуто дальнейшей декомпозиции.

Состояние деятельности можно представлять себе как составное состояние, поток управления которого включает только другие состояния

деятельности и действий.

**3. Какие нотации существуют для обозначения переходов и ветвлений в диаграммах деятельности?**

Переходы, ветвление, алгоритм разветвляющейся структуры, алгоритм циклической структуры.

#### **4. Какой алгоритм является алгоритмом разветвляющейся структуры?**

Алгоритм разветвляющейся структуры - это алгоритм, в котором вычислительный процесс осуществляется либо по одной, либо по другой ветви, в зависимости от выполнения некоторого условия.

#### **5. Чем отличается разветвляющийся алгоритм от линейного?**

Линейный алгоритм - алгоритм, все этапы которого выполняются однократно и строго последовательно.

Разветвляющийся алгоритм - алгоритм, содержащий хотя бы одно условие, в результате проверки которого ЭВМ обеспечивает переход на один из нескольких возможных шагов.

#### **6. Что такое условный оператор? Какие существуют его формы?**

Оператор, конструкция языка программирования, обеспечивающая выполнение определённой команды (набора команд) только при условии истинности некоторого логического выражения, либо выполнение одной из нескольких команд.

Условный оператор имеет полную и краткую формы.

#### **7. Какие операторы сравнения используются в Python?**

If, elif, else

#### **8. Что называется простым условием? Приведите примеры.**

Простым условием называется выражение, составленное из двух арифметических выражений или двух текстовых величин.

Пример: `a == b`

#### **9. Что такое составное условие? Приведите примеры.**

Составное условие – логическое выражение, содержащее несколько простых условий объединенных логическими операциями. Это операции `not`, `and`, `or`.



Пример: (a == b or a == c)

**10. Какие логические операторы допускаются при составлении сложных условий?**

not, and, or.

**11. Может ли оператор ветвления содержать внутри себя другие ветвления?**

Может.

**12. Какой алгоритм является алгоритмом циклической структуры?**

Циклический алгоритм — это вид алгоритма, в процессе выполнения которого одно или несколько действий нужно повторить.

**13. Типы циклов в языке Python.**

В Python есть 2 типа циклов: - цикл while, - цикл for.

**14. Назовите назначение и способы применения функции range.**

Функция range генерирует серию целых чисел, от значения start до stop, указанного пользователем. Мы можем использовать его для цикла for и обходить весь диапазон как список.

**15. Как с помощью функции range организовать перебор значений от 15 до 0 с шагом 2?**

range(15, 0, 2)

**16. Могут ли быть циклы вложенными?**

Могут.

**17. Как образуется бесконечный цикл и как выйти из него?**

Бесконечный цикл в программировании — цикл, написанный таким образом, что условие выхода из него никогда не выполняется.

**18. Для чего нужен оператор break?**

Используется для выхода из цикла.

**19. Где употребляется оператор continue и для чего он используется?**

Оператор `continue` используется только в циклах. В операторах `for` , `while` , `do while` , оператор `continue` выполняет пропуск оставшейся части кода тела цикла и переходит к следующей итерации цикла.

## **20. Для чего нужны стандартные потоки `stdout` и `stderr`?**

Ввод и вывод распределяется между тремя стандартными потоками: `stdin` — стандартный ввод (клавиатура), `stdout` — стандартный вывод (экран), `stderr` — стандартная ошибка (вывод ошибок на экран)

## **21. Как в Python организовать вывод в стандартный поток `stderr`?**

Указать в `print(..., file=sys.stderr)`.

## **22. Каково назначение функции `exit`?**

Функция `exit()` модуля `sys` - выход из Python.

**Вывод:** исследовали процесс установки и базовые возможности языка Python версии 3.