1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт компьютерных наук и кибербезопасности
5. **Кафедра «Информационная безопасность компьютерных систем»**

**ЛАБОРАТОРНАЯ РАБОТА № 1**

1. «Управление ФС/реестром посредством CLI приложения на ЯП Python»
2. по дисциплине «Цифровая культура»
3. Выполнил
4. студент гр. 5131001/20003 Черникова В.М.

<*подпись*>

1. Преподаватель
2. Писков А. А.

<*подпись*>

1. Санкт-Петербург
2. 2024
3. **Цель лабораторной работы** — Получение навыков управления файловой системой и реестром с использованием языка программирования Python.
4. **Алгоритма работы программы**

**Для начала надо импортировать все нужные библиотеки:**

import argparse

import os

import shutil

from winreg import \*

**Далее напишем инструкцию, с помощью которой программу можно будет запускать из программной строки:**

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

main()

**И главную функцию, где парсятся аргументы и выбирается команда:**

def main ():

p = argparse.ArgumentParser()

# file system args

p.add\_argument("-create\_file","--create", help="Creating of file")

p.add\_argument("-delete\_file","--delete", help="Delete file")

p.add\_argument("-write\_to\_file", "--write", nargs=2, help="To write anything into file")

p.add\_argument("-read\_from\_file", "--read", help="To read information from file")

p.add\_argument("-copy\_file\_to\_dir", "--copy", nargs=2, help="To copy file between directories")

p.add\_argument("-rename\_file", "--rename", nargs=2, help="To rename file")

p.add\_argument("-create\_key", "--cr\_key", nargs=2, help="To rename file")

p.add\_argument("-delete\_key", "--del\_key", nargs=2, help="To rename file")

p.add\_argument("-write\_key", "--w\_key", nargs=3, help="To rename file")

args = p.parse\_args()#массив с аргументами, который находится, например, в write

print(args)

**Здесь выбирается и выполняется команда:**

if args.write:

with open(args.write[1], "w") as file:

file.write(args.write[0])

if args.read:

with open(args.read, "r") as file:

mmm=file.read()

print(mmm)

if args.delete:

try:

os.remove(args.delete)

except FileNotFoundError:

print("No such file")

if args.create:

try:

with open(args.create, "x") as f:

print("File created")

except FileExistsError:

print("File is already exist")

if args.rename:

try:

os.rename(args.rename[0], args.rename[1])

except FileNotFoundError:

print("No such file")

if args.copy:

try:

shutil.copyfile(args.copy[0], args.copy[1]+'\\'+args.copy[0])

except FileNotFoundError:

print("No such file")

**Напишем функцию, котрая будет парсить аргументы и выбирать корни реестра**

def my\_choise ( string ) :

ss=list(string.split("\\"))[0]

#print(ss)

if ss=="HKEY\_CLASSES\_ROOT":

return HKEY\_CLASSES\_ROOT

if ss=="HKEY\_LOCAL\_MACHINE":

return HKEY\_LOCAL\_MACHINE

if ss=="HKEY\_USERS":

return HKEY\_USERS

if ss=="HKEY\_DYN\_DATA":

return HKEY\_DYN\_DATA

if ss=="HKEY\_CURRENT\_CONFIG":

return HKEY\_CURRENT\_CONFIG

if ss=="HKEY\_CURRENT\_USER":

return HKEY\_CURRENT\_USER

if ss=="HKEY\_PERFORMANCE\_DATA":

return HKEY\_PERFORMANCE\_DATA

**Далее нужно написать функции для работы с регистром**

if args.del\_key:

key\_val = args.del\_key[1].replace(list(args.del\_key[1].split("\\"))[0],"")[1:]+"\\"+args.del\_key[0]

path\_root=my\_choise(args.del\_key[1])

try:

key = OpenKey(path\_root, key\_val)

val1 = DeleteKey(path\_root, key\_val)

CloseKey(key)

print("Key deleted")

except FileNotFoundError:

print("No such key")

if args.cr\_key:

key\_val = args.cr\_key[1].replace(list(args.cr\_key[1].split("\\"))[0],"")[1:]+"\\"+args.cr\_key[0]

path\_root=my\_choise(args.cr\_key[1])

#print(key\_val)

try:

key = OpenKey(path\_root, key\_val)

print("Key has already been created")

CloseKey(key)

except FileNotFoundError:

key = CreateKey(path\_root, key\_val)

CloseKey(key)

print("Key created")

if args.w\_key:

key\_val = args.w\_key[1].replace(list(args.w\_key[1].split("\\"))[0],"")[1:]+"\\"+args.w\_key[0]

path\_root=my\_choise(args.w\_key[1])

try:

key = OpenKey(path\_root, key\_val)

SetValue(path\_root, key\_val, REG\_SZ, args.w\_key[2])

CloseKey(key)

except FileNotFoundError:

key = CreateKey(path\_root, key\_val)

SetValue(path\_root, key\_val, REG\_SZ, args.w\_key[2])

CloseKey(key)

1. **Ответы на контрольные вопросы**
2. ***Чем компилируемый язык программирования отличается от интерпретируемого? Приведите примеры языков программирования для обоих типов.***

Компилируемые языки сразу переводятся в машинный код, который может выполнить процессор. В результате они выполняются быстрее и эффективнее, чем интерпретируемые языки. Кроме того, в таких языках разработчик лучше контролирует аппаратные средства (управление памятью, использование ЦП и т.д.). Компилируемым языкам требуется дополнительный этап «сборки», при котором их сначала компилируют вручную. Каждый раз при внесении изменений вам нужно будет «пересобирать» программу. Примеры истинных компилируемых языков: C, **C++, Erlang, Haskell, Rust и Go**.

Интерпретаторы проходятся по каждой строке программы и выполняют все команды. Интерпретируемые языки гораздо медленнее компилируемых. Но с появлением JIT-компиляции (динамической компиляции) эта разница начинает сокращаться. Примеры популярных интерпретируемых языков: **PHP, Ruby, Python и JavaScript**.

1. ***Каким образом могут быть считаны входные данные для консольного приложения помимо использования модуля argparse?***

Например, с помощью библиотеки sys:

import sys

print("\n".join(sys.argv))

Или с помощью библиотеки docopt:

from docopt import docopt

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

arguments = docopt(\_\_doc\_\_, version='Naval Fate 2.0')

print(arguments)

1. ***В чем разница открытия файла конструкцией:***

fd = open(‘test.txt’, ‘r’)

…

fd.close()

***от конструкции:***

with open(‘test.txt’, ‘r’) as fd:

…

Конструкция with автоматически закрывает файл

1. ***Каким образом могут быть установлены модули для Python?***

С помощью pip или внутри виртуальной среды, например, в IDE PyCharm

1. **Приложение.**

Полный листинг программы. Дополнительные задания выделелены желтым.

import argparse  
import os  
import shutil  
from winreg import \*  
  
  
def my\_choise(string):  
 ss = list(string.split("\\"))[0]  
 # print(ss)  
 if ss == "HKEY\_CLASSES\_ROOT":  
 return HKEY\_CLASSES\_ROOT  
 if ss == "HKEY\_LOCAL\_MACHINE":  
 return HKEY\_LOCAL\_MACHINE  
 if ss == "HKEY\_USERS":  
 return HKEY\_USERS  
 if ss == "HKEY\_DYN\_DATA":  
 return HKEY\_DYN\_DATA  
 if ss == "HKEY\_CURRENT\_CONFIG":  
 return HKEY\_CURRENT\_CONFIG  
 if ss == "HKEY\_CURRENT\_USER":  
 return HKEY\_CURRENT\_USER  
 if ss == "HKEY\_PERFORMANCE\_DATA":  
 return HKEY\_PERFORMANCE\_DATA  
  
  
def search\_registry(path\_root, key\_val, meaning):  
 #print("in dir: ", path\_root, key\_val, meaning)  
 try:  
 key = OpenKey(path\_root, key\_val)  
 num\_keys = QueryInfoKey(key)[0]  
 #print("info =", QueryInfoKey(key), end=" ")  
 for i in range(QueryInfoKey(key)[1]):  
 try:  
 sub\_value = EnumValue(key, i)  
 #print("sub - ", sub\_value)  
 if meaning == str(sub\_value[1]):  
 print('!', key\_val, "\\", str(sub\_value[0]))  
 except OSError:  
 pass  
 for i in range(num\_keys):  
 sub\_key\_name = EnumKey(key, i)  
 sub\_key = OpenKey(key, sub\_key\_name)  
 search\_registry(path\_root, key\_val + "\\" + sub\_key\_name, meaning)  
 CloseKey(sub\_key)  
 CloseKey(key)  
 CloseKey(path\_root)  
 except PermissionError:  
 print("Permission denied")

def main():  
 p = argparse.ArgumentParser()  
 # file system args  
 p.add\_argument("-create\_file", "--create", help="Creating of file")  
 p.add\_argument("-delete\_file", "--delete", help="Delete file")  
 p.add\_argument("-find\_string", "--find", help="Find file")  
 p.add\_argument("-write\_to\_file", "--write", nargs=2, help="To write anything into file")  
 p.add\_argument("-read\_from\_file", "--read", help="To read information from file")  
 p.add\_argument("-copy\_file\_to\_dir", "--copy", nargs=2, help="To copy file between directories")  
 p.add\_argument("-rename\_file", "--rename", nargs=2, help="To rename the file")  
 p.add\_argument("-create\_key", "--cr\_key", nargs=2, help="To create the key")  
 p.add\_argument("-delete\_key", "--del\_key", nargs=2, help="To delete the key")  
 p.add\_argument("-write\_key", "--w\_key", nargs=3, help="To write into key")  
 p.add\_argument("-find\_key", "--f\_key", nargs=2, help="To find the meaning in keys")  
 args = p.parse\_args() # массив с аргументами, который находится, например, в write  
 # print(args)  
 if args.write:  
 with open(args.write[1], "w") as file:  
 file.write(args.write[0])  
 if args.read:  
 with open(args.read, "r") as file:  
 mmm = file.read()  
 print(mmm)  
 if args.delete:  
 try:  
 os.remove(args.delete)  
 except FileNotFoundError:  
 print("No such file")  
 if args.create:  
 try:  
 with open(args.create, "x") as f:  
 print("File created")  
 except FileExistsError:  
 print("File is already exist")  
 if args.rename:  
 try:  
 os.rename(args.rename[0], args.rename[1])  
 except FileNotFoundError:  
 print("No such file")  
 if args.copy:  
 try:  
 shutil.copyfile(args.copy[0], args.copy[1] + '\\' + args.copy[0])  
 except FileNotFoundError:  
 print("No such file")  
 if args.find:  
 for root, dirs, files in os.walk(args.search[1]):  
 for file in files:  
 file\_path = os.path.join(root, file)  
 try:  
 with open(file\_path, 'r') as f:  
 contents = f.read()  
 if args.search[0] in contents:  
 print(file\_path)  
 except Exception as e:  
 continue  
 if args.del\_key:  
 key\_val = args.del\_key[1].replace(list(args.del\_key[1].split("\\"))[0], "")[1:] + "\\" + args.del\_key[0]  
 path\_root = my\_choise(args.del\_key[1])  
 try:  
 key = OpenKey(path\_root, key\_val)  
 DeleteKey(path\_root, key\_val)  
 CloseKey(key)  
 print("Key deleted")  
 except FileNotFoundError:  
 print("No such key")  
 if args.cr\_key:  
 key\_val = args.cr\_key[1].replace(list(args.cr\_key[1].split("\\"))[0], "")[1:] + "\\" + args.cr\_key[0]  
 path\_root = my\_choise(args.cr\_key[1])  
 print(key\_val)  
 try:  
 key = OpenKey(path\_root, key\_val)  
 print("Key has already been created")  
 CloseKey(key)  
 except FileNotFoundError:  
 key = CreateKey(path\_root, key\_val)  
 CloseKey(key)  
 print("Key created")  
 if args.w\_key:  
 key\_val = args.w\_key[1].replace(list(args.w\_key[1].split("\\"))[0], "")[1:] + "\\" + args.w\_key[0]  
 path\_root = my\_choise(args.w\_key[1])  
 try:  
 key = OpenKey(path\_root, key\_val)  
 SetValue(path\_root, key\_val, REG\_SZ, args.w\_key[2])  
 CloseKey(key)  
 except FileNotFoundError:  
 key = CreateKey(path\_root, key\_val)  
 SetValue(path\_root, key\_val, REG\_SZ, args.w\_key[2])  
 CloseKey(key)  
 if args.f\_key:  
 key\_val = args.f\_key[1].replace(list(args.f\_key[1].split("\\"))[0], "")[1:]  
 path\_root = my\_choise(args.f\_key[1])  
 search\_registry(path\_root, key\_val, args.f\_key[0])  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()