**Домашнее задание к занятию "4.2. Использование Python для решения типовых DevOps задач". Корнеев Роман**

**Обязательная задача 1**

Есть скрипт:

#!/usr/bin/env python3

a = 1

b = '2'

c = a + b

**Вопросы:**

| **Вопрос** | **Ответ** |
| --- | --- |
| Какое значение будет присвоено переменной c? | ??? |
| Как получить для переменной c значение 12? | ??? |
| Как получить для переменной c значение 3? | ??? |

**Ответ:**

1. произойдет ошибка, ввиду того разные типы данных (int и str) и эти типы не подходят для операции.

2. поменять a к типу строка (str): c=str(a)+b

3. поменять b к типу целое число (int): c=a+int(b)

**Обязательная задача 2**

Мы устроились на работу в компанию, где раньше уже был DevOps Engineer. Он написал скрипт, позволяющий узнать, какие файлы модифицированы в репозитории, относительно локальных изменений. Этим скриптом недовольно начальство, потому что в его выводе есть не все изменённые файлы, а также непонятен полный путь к директории, где они находятся. Как можно доработать скрипт ниже, чтобы он исполнял требования вашего руководителя?

#!/usr/bin/env python3

import os

bash\_command = ["cd ~/netology/sysadm-homeworks", "git status"]

result\_os = os.popen(' && '.join(bash\_command)).read()

is\_change = False

for result in result\_os.split('\n'):

if result.find('modified') != -1:

prepare\_result = result.replace('\tmodified: ', '')

print(prepare\_result)

break

**Ответ:**

#!/usr/bin/env python3

import os

bash\_command = ["cd ~/netology/sysadm-homeworks", "git status"]

result\_os = os.popen(' && '.join(bash\_command)).read()

#is\_change = False

for result in result\_os.split('\n'):

if result.find('modified') != -1:

prepare\_result = result.replace('\tmodified: ', '')

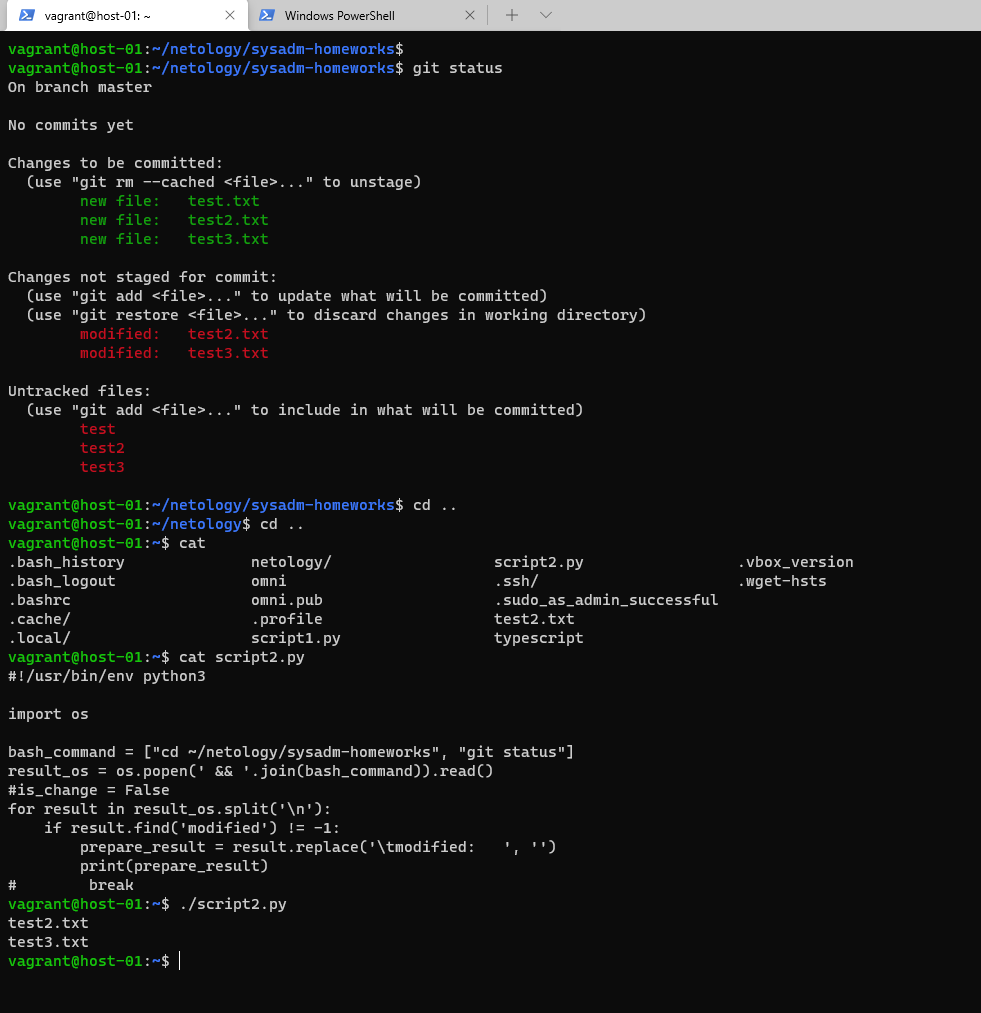
print(prepare\_result)

# break

===========

логическая переменная is\_change является лишней, также лишняя команда break которая прерывает обработку при первом же найденном вхождении. Скриншот ниже.

**Вывод скрипта при запуске при тестировании:**



**Обязательная задача 3**

1. Доработать скрипт выше так, чтобы он мог проверять не только локальный репозиторий в текущей директории, а также умел воспринимать путь к репозиторию, который мы передаём как входной параметр. Мы точно знаем, что начальство коварное и будет проверять работу этого скрипта в директориях, которые не являются локальными репозиториями.

**Ответ:**

#!/usr/bin/env python3

import os

import sys

cmd = os.getcwd()

if len(sys.argv)>=2:

cmd = sys.argv[1]

bash\_command = ["cd "+cmd, "git status 2>&1"]

print('\033[31m')

result\_os = os.popen(' && '.join(bash\_command)).read()

#is\_change = False

for result in result\_os.split('\n'):

if result.find('fatal') != -1:

print('\033[31m Каталог \033[1m '+cmd+'\033[0m\033[31m не является GIT репозиторием\033[0m')

if result.find('modified') != -1:

prepare\_result = result.replace('\tmodified: ', '')

# для удобства вывода добавлена замена всех пробелов

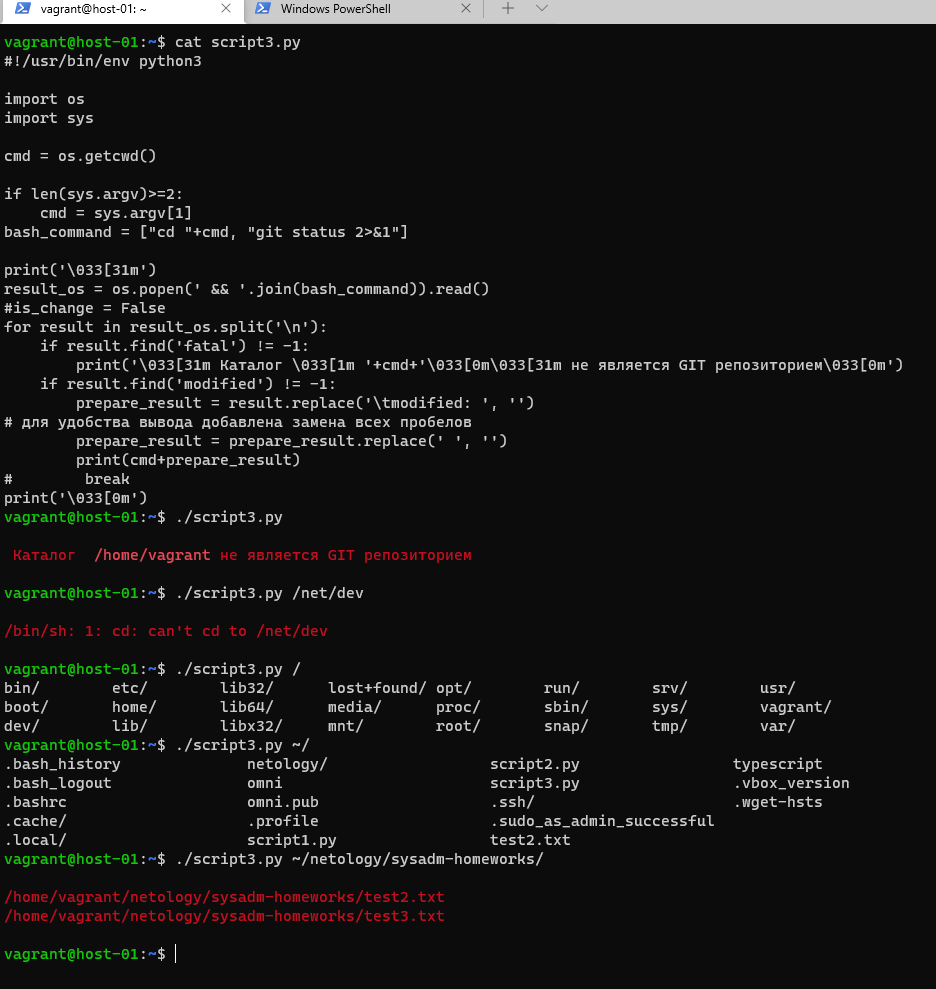
prepare\_result = prepare\_result.replace(' ', '')

print(cmd+prepare\_result)

# break

print('\033[0m')

**Вывод скрипта при запуске при тестировании:**



**Обязательная задача 4**

1. Наша команда разрабатывает несколько веб-сервисов, доступных по http. Мы точно знаем, что на их стенде нет никакой балансировки, кластеризации, за DNS прячется конкретный IP сервера, где установлен сервис. Проблема в том, что отдел, занимающийся нашей инфраструктурой очень часто меняет нам сервера, поэтому IP меняются примерно раз в неделю, при этом сервисы сохраняют за собой DNS имена. Это бы совсем никого не беспокоило, если бы несколько раз сервера не уезжали в такой сегмент сети нашей компании, который недоступен для разработчиков. Мы хотим написать скрипт, который опрашивает веб-сервисы, получает их IP, выводит информацию в стандартный вывод в виде: <URL сервиса> - <его IP>. Также, должна быть реализована возможность проверки текущего IP сервиса c его IP из предыдущей проверки. Если проверка будет провалена - оповестить об этом в стандартный вывод сообщением: [ERROR] <URL сервиса> IP mismatch: <старый IP> <Новый IP>. Будем считать, что наша разработка реализовала сервисы: drive.google.com, mail.google.com, google.com.

**Ответ:**

#!/usr/bin/env python3

import socket as sok

import time as tm

import datetime as dt

# set variables

i = 1

wait = 2 # интервал проверок в секундах

servers = {'drive.google.com':'0.0.0.0', 'mail.google.com':'0.0.0.0', 'google.com':'0.0.0.0'}

init=0

print('start check')

print(servers)

print('wait process')

while 1==1 : #отладочное число проверок

for host in servers:

ip = sok.gethostbyname(host)

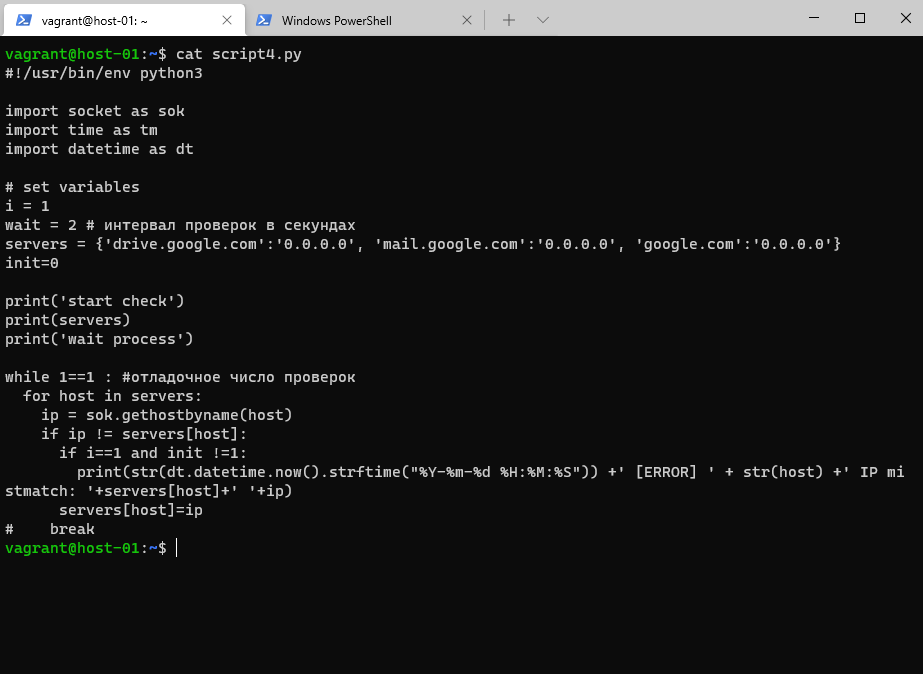
if ip != servers[host]:

if i==1 and init !=1:

print(str(dt.datetime.now().strftime("%Y-%m-%d %H:%M:%S")) +' [ERROR] ' + str(host) +' IP mistmatch: '+servers[host]+' '+ip)

servers[host]=ip

# break



**Вывод скрипта при запуске при тестировании:**

