

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 №9  
дисциплины «Программирование на языке Python»

Вариант\_15\_

Выполнила:  
Маньшина Дарья Алексеевна  
2 курс, группа ИТС-б-о-22-1,  
11.03.02 «Инфокоммуникационные  
технологии и системы связи»,  
направленность (профиль)  
«Инфокоммуникационные системы и  
сети», очная форма обучения

---

(подпись)

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

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

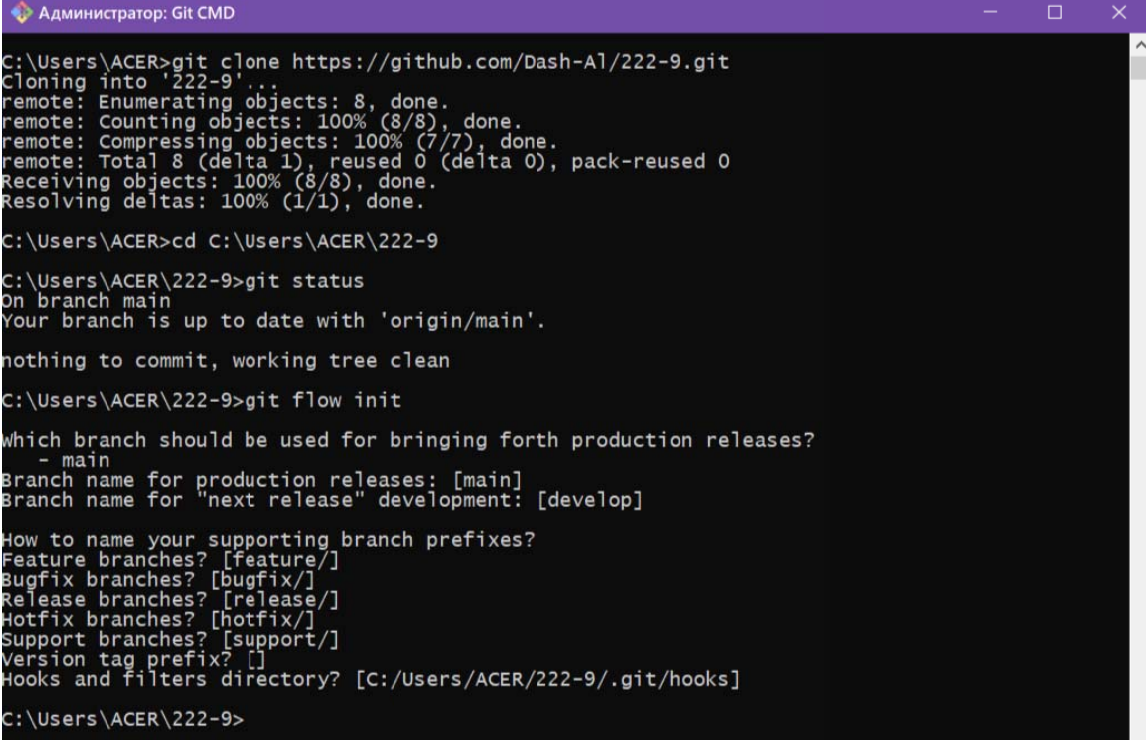
Ставрополь, 2023 г.

Тема: тестирование в Python [unittest]

Цель: приобретение навыков написания автоматизированных тестов на языке программирования Python версии 3.x.

Ход работы:

1. Подготовка к выполнению работы. Клонирование репозитория и добавление пакетов black, isort, flake8.



```
Администратор: Git CMD

C:\Users\ACER>git clone https://github.com/Dash-A1/222-9.git
Cloning into '222-9'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.
Resolving deltas: 100% (1/1), done.

C:\Users\ACER>cd C:\Users\ACER\222-9

C:\Users\ACER\222-9>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

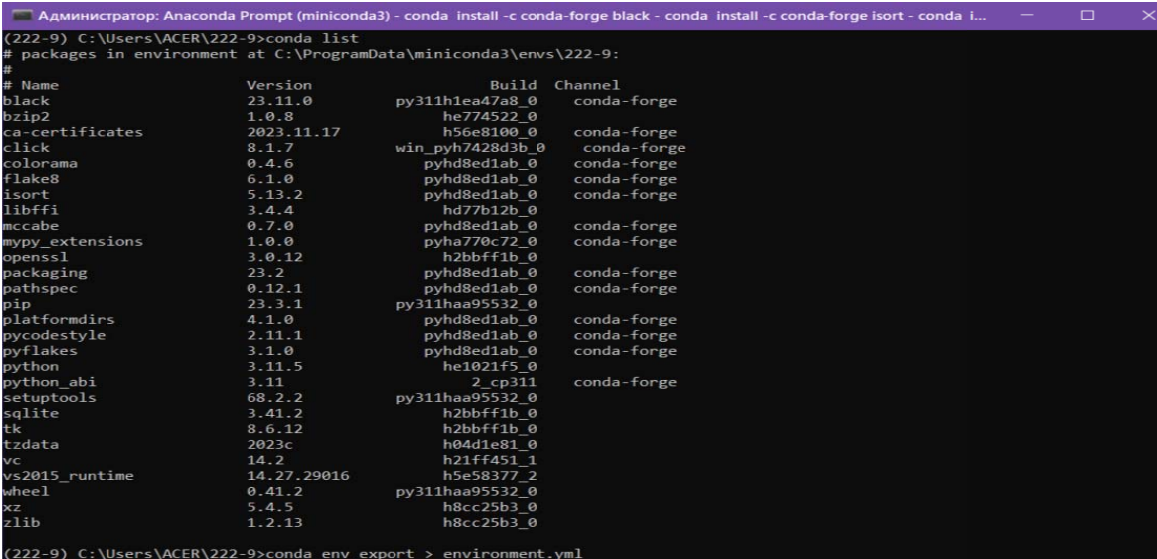
C:\Users\ACER\222-9>git flow init

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

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? [C:/Users/ACER/222-9/.git/hooks]

C:\Users\ACER\222-9>
```

Рисунок 1. Клонирование репозитория



```
Администратор: Anaconda Prompt (miniconda3) - conda install -c conda-forge black - conda install -c conda-forge isort - conda i...

(222-9) C:\Users\ACER\222-9>conda list
# packages in environment at C:\ProgramData\miniconda3\envs\222-9:
#
# Name                    Version           Build    Channel
black                     23.11.0           py311h1ea47a8_0 conda-forge
bzip2                     1.0.8             he774522_0     conda-forge
ca-certificates           2023.11.17        h56e8100_0     conda-forge
click                     8.1.7             win_pyh7428d3b_0 conda-forge
colorama                  0.4.6             pyhd8ed1ab_0   conda-forge
flake8                    6.1.0             pyhd8ed1ab_0   conda-forge
isort                      5.13.2            pyhd8ed1ab_0   conda-forge
libffi                    3.4.4             hd77b12b_0     conda-forge
mccabe                    0.7.0             pyhd8ed1ab_0   conda-forge
mypy_extensions           1.0.0             pyha770c72_0   conda-forge
openssl                   3.0.12            h2bbff1b_0     conda-forge
packaging                  23.2              pyhd8ed1ab_0   conda-forge
pathspec                   0.12.1            pyhd8ed1ab_0   conda-forge
pip                       23.3.1            py311haa95532_0 conda-forge
platformdirs               4.1.0             pyhd8ed1ab_0   conda-forge
pycodestyle                2.11.1            pyhd8ed1ab_0   conda-forge
pyflakes                   3.1.0             pyhd8ed1ab_0   conda-forge
python                     3.11.5            he1021f5_0     conda-forge
python_abi                 3.11              2_cp311        conda-forge
setuptools                 68.2.2            py311haa95532_0 conda-forge
sqlite                     3.41.2            h2bbff1b_0     conda-forge
tk                          8.6.12            h2bbff1b_0     conda-forge
tzdata                     2023c             h04d1e81_0     conda-forge
vc                          14.2              h21ff451_1     conda-forge
vs2015_runtime             14.27.29016       h5e58377_2     conda-forge
wheel                      0.41.2            py311haa95532_0 conda-forge
xz                          5.4.5             h8cc25b3_0     conda-forge
zlib                       1.2.13            h8cc25b3_0     conda-forge

(222-9) C:\Users\ACER\222-9>conda env export > environment.yml
```

Рисунок 2. Установка пакетов

Проработаем примеры из методички. Для начала создадим модуль `calc.py`:

```
def add(a, b):
    return a + b
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
def sub(a, b):
    return a - b

def mul(a, b):
    return a * b

def div(a, b):
    if b == 0:
        raise ValueError("Cannot divide by zero")
    return a / b
```

Для того, чтобы протестировать эту библиотеку, мы можем создать отдельный файл с названием `test_calc.py` и поместить туда функции, которые проверяют корректность работы функций из `calc.py`.

Программа `test_calc.py`:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import calc
if __name__ == '__main__':
    def test_add():
        if calc.add(1, 2) == 3:
            print("Test add(a, b) is OK")
        else:
            print("Test add(a, b) is Fail")
    def test_sub():
        if calc.sub(4, 2) == 2:
            print("Test sub(a, b) is OK")
        else:
            print("Test sub(a, b) is Fail")
    def test_mul():
        if calc.mul(2, 5) == 10:
            print("Test mul(a, b) is OK")
        else:
            print("Test mul(a, b) is Fail")
    def test_div():
        if calc.div(8, 4) == 2:
            print("Test div(a, b) is OK")
        else:
            print("Test div(a, b) is Fail")
    test_add()
    test_sub()
    test_mul()
    test_div()
```

Результат:

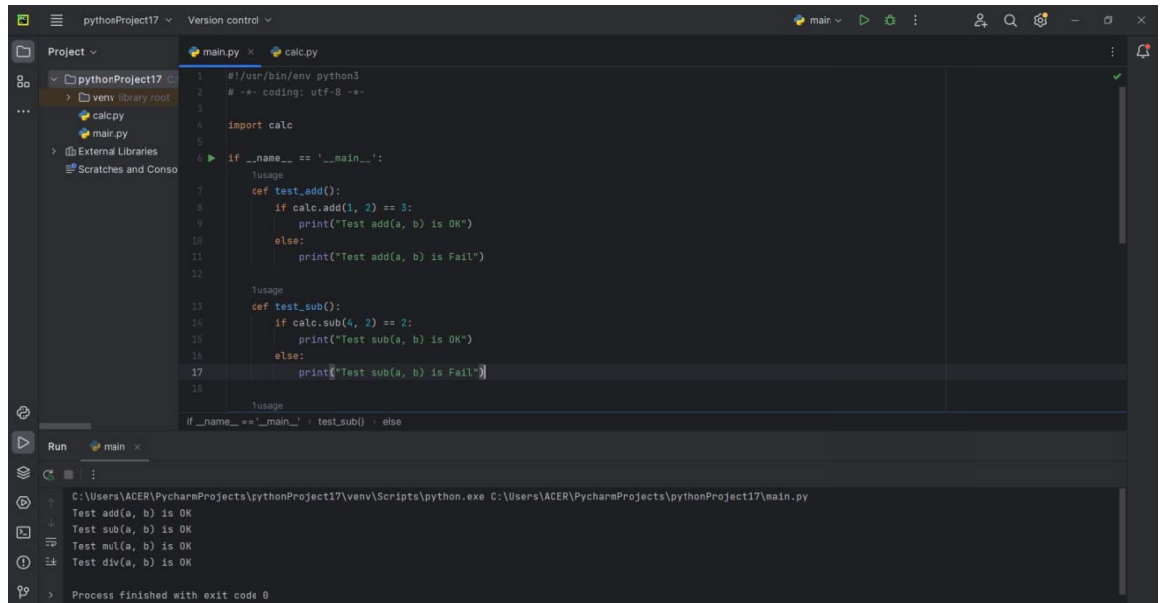


Рисунок 3. Результат программы test\_calc.py

Теперь посмотрим как можно было бы протестировать набор функций из calc.py с помощью unittest.

Для этого сделаем следующие действия:

1. Создадим файл с именем utest\_calc.py
2. Добавим в него следующий код:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
```

```
import unittest
import calc
```

```
class CalcTest(unittest.TestCase):
    """Calc tests"""
    @classmethod
    def setUpClass(cls):
        """Set up for class"""
        print("setUpClass")
        print("=====")
```

```
    @classmethod
    def tearDownClass(cls):
        """Tear down for class"""
        print("=====")
        print("tearDownClass")
```

```
    def setUp(self):
        """Set up for test"""
```

```

        print("Set up for [" + self.shortDescription() + "]")

def tearDown(self):
    """Tear down for test"""
    print("Tear down for [" + self.shortDescription() + "]")
    print("")

def test_add(self):
    """Add operation test"""
    print("id: " + self.id())
    self.assertEqual(calc.add(1, 2), 3)

def test_sub(self):
    """Sub operation test"""
    print("id: " + self.id())
    self.assertEqual(calc.sub(4, 2), 2)

def test_mul(self):
    """Mul operation test"""
    print("id: " + self.id())
    self.assertEqual(calc.mul(2, 5), 10)

def test_div(self):
    """Div operation test"""
    print("id: " + self.id())
    self.assertEqual(calc.div(8, 4), 2)

if __name__ == '__main__':
    unittest.main()

```

Результат:

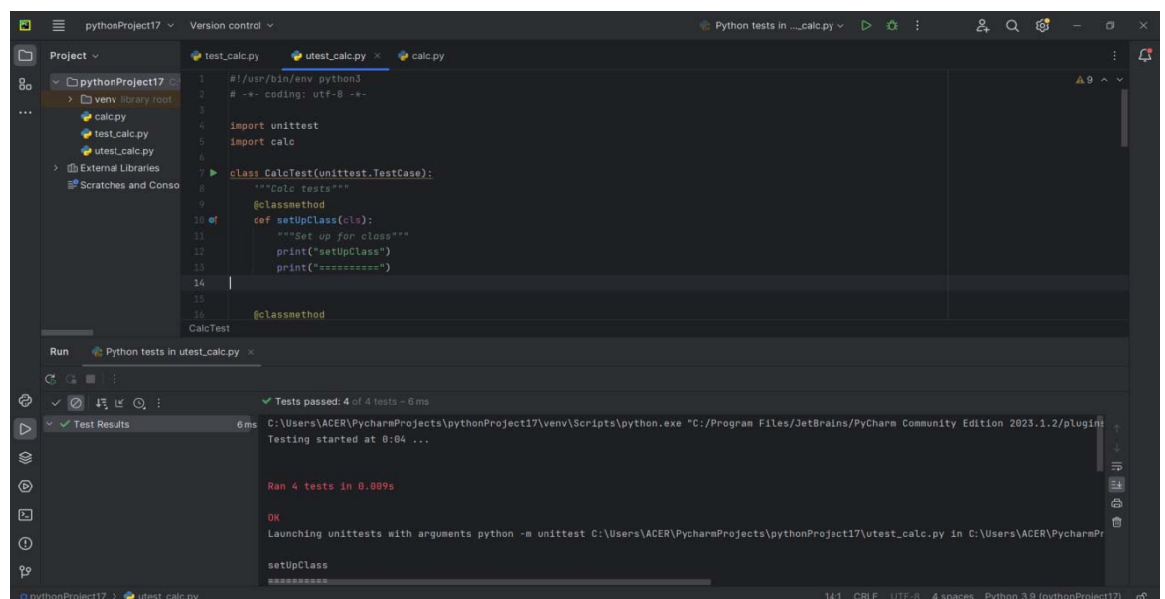


Рисунок 4. Результат работы программы `utest_calc.py`

Выполним ИДЗ:

Условие: # использовать словарь, содержащий следующие ключи: фамилия, имя; знак Зодиака; дата рождения (список из трёх чисел). Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть упорядочены по датам рождения; вывод на экран информацию о людях, родившихся под знаком, название которого введено с клавиатуры; если таких нет, выдать на дисплей соответствующее сообщение. Оформив каждую команду в виде отдельной функции. Добавьте тесты с использованием модуля unittest, проверяющие операции по работе с базой данных.

Программа:

#Использовать словарь, содержащий следующие ключи: фамилия, имя; знак Зодиака; дата рождения (список из трёх чисел).

# Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры;

# записи должны быть упорядочены по датам рождения; вывод на экран информацию о людях, родившихся под знаком, название которого введено с клавиатуры;

# если таких нет, выдать на дисплей соответствующее сообщение. Оформив каждую команду в виде отдельной функции.

# Добавьте возможность получения имени файла данных, используя соответствующую переменную окружения.

```
#!/usr/bin/env python3
```

```
# -*- coding: utf-8 -*-
```

```
import os
```

```
import unittest
```

```
def input_data():
```

```
    data = []
```

```
    while True:
```

```
        surname = input("Введите фамилию: ")
```

```
        name = input("Введите имя: ")
```

```
        zodiac = input("Введите знак Зодиака: ")
```

```
        birthday = input("Введите дату рождения (через пробел): ").split()
```

```
        if len(birthday) != 3:
```

```
            print("Неверный формат даты. Повторите ввод.")
```

```
            continue
```

```
        try:
```

```
            birthday = [int(x) for x in birthday]
```

```
        except ValueError:
```

```
            print("Неверный формат даты. Повторите ввод.")
```

```
            continue
```

```
        data.append({
```

```
            "фамилия": surname,
```

```
            "имя": name,
```

```

        "знак Зодиака": zodiac,
        "дата рождения": birthday
    })
    if input("Желаете добавить еще запись? (y/n): ") != 'y':
        break
data.sort(key=lambda x: x["дата рождения"])
return data

def find_people_by_zodiac(data, zodiac):
    people = []
    for person in data:
        if person["знак Зодиака"] == zodiac:
            people.append(person)
    return people

def print_people(people):
    if len(people) == 0:
        print("Нет людей с таким знаком Зодиака.")
    else:
        for person in people:
            print("Фамилия: {}".format(person["фамилия"]))
            print("Имя: {}".format(person["имя"]))
            print("Знак Зодиака: {}".format(person["знак Зодиака"]))
            print("Дата рождения: {}/{}/{}".format(person["дата рождения"][0], person["дата
рождения"][1],
                                                    person["дата рождения"][2]))
            print()

def main():
    filename = os.environ.get("DATA_FILE")
    if filename:
        with open(filename, "r") as file:
            data = eval(file.read())
    else:
        data = input_data()

    zodiac = input("Введите знак Зодиака для поиска: ")
    people = find_people_by_zodiac(data, zodiac)
    print_people(people)

class TestDatabase(unittest.TestCase):
    def setUp(self):
        self.db = []

    def test_input_data(self):
        input_data(self.db)
        self.assertEqual(len(self.db), 2)
        self.assertEqual(self.db[0]["фамилия"], "Иванов")
        self.assertEqual(self.db[1]["фамилия"], "Петров")

    def test_find_persons_by_zodiac(self):
        self.db = [

```

```

{"фамилия": "Иванов", "имя": "Иван", "знак Зодиака": "Овен", "дата рождения": [1,
4, 2000]}},
{"фамилия": "Петров", "имя": "Петр", "знак Зодиака": "Лев", "дата рождения": [5, 6,
1999]}},
{"фамилия": "Сидорова", "имя": "Мария", "знак Зодиака": "Овен", "дата рождения":
[7, 8, 2001]}
]

# Тест существующего знака Зодиака
result = find_persons_by_zodiac(self.db, "Овен")
self.assertEqual(result, "Иванов Иван: [1, 4, 2000]\nСидорова Мария: [7, 8, 2001]")

# Тест несуществующего знака Зодиака
result = find_persons_by_zodiac(self.db, "Рыбы")
self.assertEqual(result, "В базе данных нет людей, родившихся под этим знаком
Зодиака")

```

```

if __name__ == "__main__":
    unittest.main()

```

Результат:

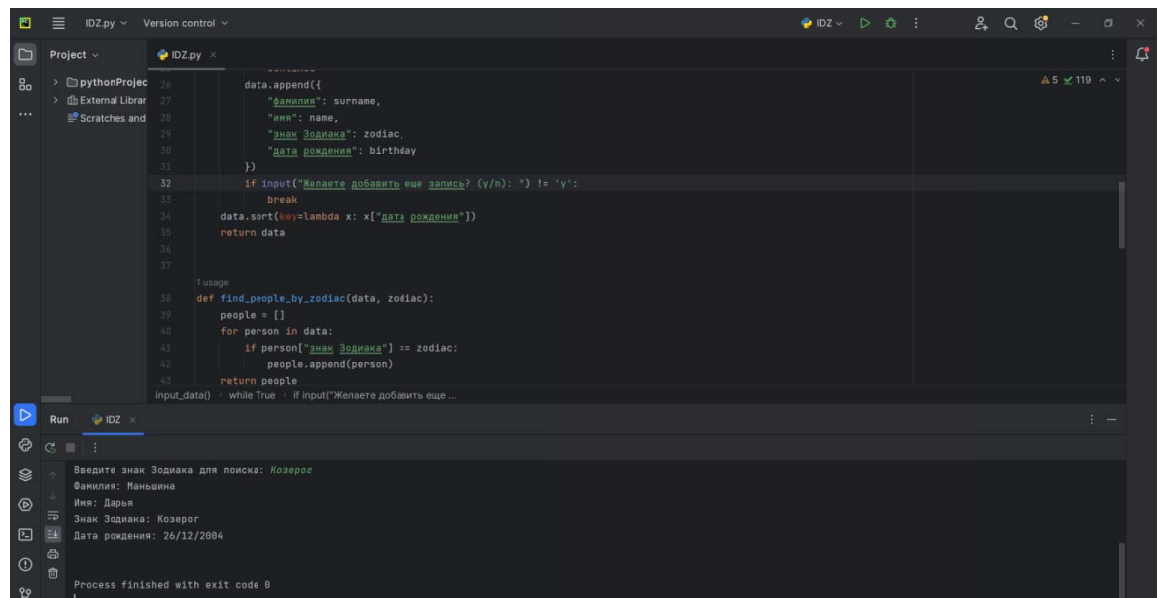


Рисунок 5. Результат индивидуального задания

Вывод: в ходе лабораторной работы ознакомилась с написанием автоматизированных тестов на языке программирования Python версии 3.x.

### Контрольные вопросы

1. Для чего используется автономное тестирование?

Автономное тестирование используется для проверки функциональности и корректности работы отдельных модулей или компонентов программы без необходимости запуска всей системы в целом.



2. Какие фреймворки Python получили наибольшее распространение для решения задач автономного тестирования?

Для автономного тестирования на Python наиболее распространены фреймворки unittest, pytest, nose и другие.

3. Какие существуют основные структурные единицы модуля unittest?

Основными структурными единицами модуля unittest являются класс TestCase, представляющий один тест, и класс TestSuite, объединяющий несколько тестов.

4. Какие существуют способы запуска тестов unittest?

Тесты unittest можно запускать вручную, с помощью командной строки или интегрированной среды разработки.

5. Каково назначение класса TestCase?

Класс TestCase содержит методы setUp, tearDown, которые выполняются перед и после выполнения теста соответственно, а также метод test, который выполняет сам тест.

6. Какие методы класса TestCase выполняются при запуске и завершении работы тестов?

Методы класса TestCase также включают assertEquals для проверки равенства значений, assertNotEqual для проверки неравенства значений, и assertRaises для проверки генерации исключений.

7. Какие методы класса TestCase используются для проверки условий и генерации ошибок?

Методы сбора информации о тесте включают метод `setUpClass`, который выполняется один раз для всех тестов в классе, и метод `tearDownClass`, который выполняется после завершения всех тестов.

8. Какие методы класса `TestCase` позволяют собирать информацию о самом тесте?

Класс `TestSuite` используется для объединения нескольких тестов в один набор, а класс `TestResult` используется для получения результатов выполнения тестов.

9. Каково назначение класса `TestSuite`? Как осуществляется загрузка тестов?

Класс `TestSuite` предназначен для объединения группы тестов в единый тестовый набор. Загрузка тестов осуществляется через импорт соответствующих тестовых модулей или использование встроенной функции `unittest.TestLoader.loadTestsFromNames`.

10. Каково назначение класса `TestResult`?

Класс `TestResult` предназначен для получения результатов тестирования. Он содержит информацию о количестве успешных и неудачных тестов, времени выполнения тестов и т.д.

11. Для чего может понадобиться пропуск отдельных тестов?

Пропуск отдельных тестов может понадобиться в случае, если нужно временно исключить некоторые тесты из набора или если нужно провести тестирование только определенных тестов.

12. Как выполняется безусловный и условных пропуск тестов? Как выполнить пропуск класса тестов?

Безусловный пропуск теста выполняется с помощью метода `TestSuite.skipTest`, который пропускает указанный тест без выполнения. Условный пропуск теста может быть выполнен с помощью метода `TestCase.skipIf`, который пропускает тест, если заданное условие выполняется. Пропуск класса тестов может быть выполнен через использование декоратора `@unittest.skipUnless`.