

دانشگاه صنعتی شریف



دانشکده مهندسی کامپیوتر

پروژه درس مبانی برنامه‌سازی پایتون

گروه ۷ - پاییز ۱۴۰۳

عنوان پروژه: سامانه آموزشی (LMS)

استاد: علی ابریشمی

طراحان پروژه:

مانی ابراهیمی
نیما پشبادی
محمدامین حیدری
سید امیرمحمد جزایری

تاریخ تحویل: ۱۵ بهمن ۱۴۰۳

فهرست مطالب

۳	پیشنهادهای ما و آموزش‌ها	۱
۳	۱.۱ گیت (Git)	۱.۱
۳	۱.۲ شی‌گرایی	۱.۲
۴	۱.۳ تست واحد (Unit test)	۱.۳
۴	۱.۴ رسم نمودار	۱.۴
۴	۱.۵ رابط کاربری گرافیکی	۱.۵
۵	۱.۶ انتشار پروژه روی GitHub	۱.۶
۵	۱.۶.۱ روش اول: استفاده از محیط ترمینال	۱.۶.۱
۶	۱.۶.۲ روش دوم: آپلود دستی فایل‌ها	۱.۶.۲

پیشنهادهای ما و آموزش‌ها

۱.۱ ♦ گیت (Git)

همانطور که از تمرین صفر به یاد دارید، گیت‌هاب محلی بود که در آن مخازن کد نسخه‌گذاری شده با سیستم کنترل نسخه^۱ گیت قرار می‌گرفتند. برای یادگیری کار با Git و GitHub توصیه می‌کنیم [این ویدیو](#) و [این ویدیو](#) را مشاهده بفرمایید.

۱.۲ ♦ شی‌گرایی

مفهوم شی‌گرایی در کلاس درس برایتان مورد بررسی قرار گرفت. اما برای مرور چند نکته برایتان در ادامه خواهیم آورد.

به طور کلی، از رویکرد شی‌گرا و برنامه‌نویسی شی‌گرا^۲ زمانی استفاده می‌کنیم که می‌خواهیم یک شی از دنیای واقعی را درون یک سیستم یا برنامه مدل کنیم. برای مثال با هم یک کتاب را مدل می‌کنیم. فرض کنید مشخصه‌های یک کتاب به صورت زیر باشد:

- تعداد صفحات
 - نام مولف
 - باز یا بسته بودن (فرض کنید در ابتدا کتاب بسته است).
 - صفحه‌ی فعلی (فرض کنید در ابتدای کار کتاب در صفحه‌ی اول خود باشد).
 - و همچنین عملیات زیر می‌تواند روی یک کتاب انجام شود:
 - باز کردن کتاب
 - بستن کتاب
 - رفتن به صفحه‌ی **p** در کتاب
- در این صورت کد زیر را خواهیم داشت:

```
class Book:
    def __init__(self, number_of_pages, author):
        self.number_of_pages = number_of_pages
        self.author = author
        self.is_open = False
        self.current_page = 1

    def open_book(self):
        self.is_open = True

    def close_book(self):
        self.is_open = False

    def go_to_page_p(self, p):
        self.current_page = p
```

^۱Version Control System (VCS)
^۲Object-oriented programming (OOP)

۱.۳ ♦ تست واحد (Unit test)

به طور کلی تست واحد را برای آزمون عملکرد صحیح و منطقی بخش‌هایی از برنامه‌هایمان می‌نویسیم. فرض کنید یک فایل به نام `my_module.py` نوشته‌ایم که محتوای آن تابع زیر است:

```
Code

# my_module.py

def add_numbers(a, b):
    """
    Adds two numbers and returns the result.
    """
    return a + b
```

برای تست آن، از تست واحد (درون فایل `test_my_module.py`) زیر استفاده می‌کنیم.

```
Code

# test_my_module.py

import unittest
from my_module import add_numbers

class TestAddNumbers(unittest.TestCase):
    def test_add_positive_numbers(self):
        # Test adding two positive numbers
        self.assertEqual(add_numbers(2, 3), 5)

    def test_add_negative_numbers(self):
        # Test adding two negative numbers
        self.assertEqual(add_numbers(-2, -3), -5)

    def test_add_mixed_numbers(self):
        # Test adding a positive and a negative number
        self.assertEqual(add_numbers(-2, 3), 1)

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

و برای تست، فایل تستمان را اجرا می‌کنیم که قبول^۳ شدن آن به معنای عملکرد صحیح و به مشکل خوردن آن^۴ به معنای عملکرد نادرست است.

۱.۴ ♦ رسم نمودار

برای یادگیری مازول‌های `matplotlib` و `seaborn` استفاده از دوره‌های آموزشی [Kaggle](#) یا مشاهده‌ی [این ویدیو](#) پیشنهاد می‌شود.

۱.۵ ♦ رابط کاربری گرافیکی

برای انجام بخش‌های مربوط به رابط کاربری گرافیکی^۵ توصیه می‌شود از مازول `Tkinter` یا `PyQT` استفاده کنید. در صورتی که از `Tkinter` استفاده می‌کنید [این ویدیو](#) و در صورتی که از `PyQT` استفاده می‌کنید [این ویدیو](#) یا [این ویدیو](#) را مشاهده بفرمایید.

pass^۳
fail^۴
GUI^۵

ابتدا نیاز است به وبسایت GitHub بروید و در آنجا اکانت خود را بسازید. پس از ساختن اکانت، وارد آن شوید و مراحل زیر را انجام دهید:

۱.۶.۱ روش اول: استفاده از محیط ترمینال

اگر بخش Git را انجام می‌دهید، توصیه می‌کنیم از دستورات گفته شده در این روش استفاده نمایید.

۱. ساختن مخزن

(آ) روی دکمه New Repository کلیک کنید.

(ب) نامی برای مخزن انتخاب کنید.

(ج) مخزن را عمومی یا خصوصی تنظیم کنید.

(د) روی Create Repository کلیک کنید.

۲. اتصال مخزن محلی به GitHub:

برای اتصال مخزن Git محلی به مخزن ایجادشده در GitHub، دستور زیر را اجرا کنید:

```
git remote add origin https://github.com/username/repository.git
```

که `repository` نام مخزن گیت‌هاب شما و `username` نام کاربری شما در گیت‌هاب است.

۳. ارسال تغییرات به GitHub:

برای ارسال تغییرات ثبت‌شده به GitHub، دستورات زیر را اجرا کنید

```
git push -u origin main
```

۴. کلون کردن مخزن از GitHub:

برای دریافت یک مخزن موجود از GitHub روی سیستم خود، دستور زیر را اجرا کنید:

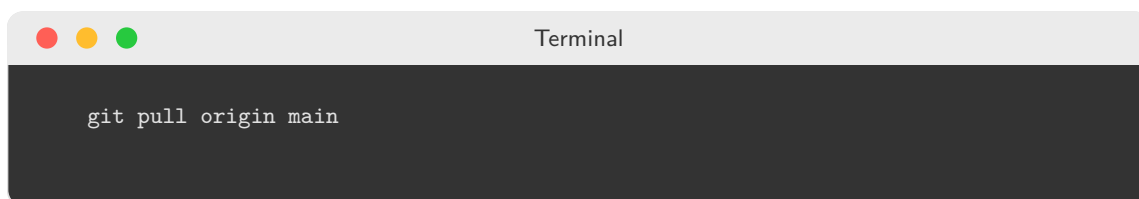
```
git clone https://github.com/username/repository.git
```

۵. مدیریت تغییرات در GitHub:

برای مشاهده سوابق تغییرات، دستور زیر را اجرا کنید:

```
git log
```

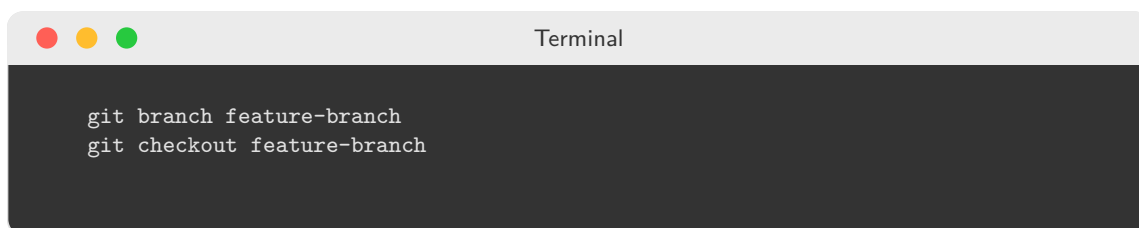
۶. برای همگام‌سازی تغییرات جدید از مخزن GitHub، از دستور زیر استفاده کنید::



```
git pull origin main
```

۷. ایجاد شاخه جدید برای ویژگی‌ها:

برای توسعه ویژگی‌های جدید بدون تأثیر بر شاخه اصلی:



```
git branch feature-branch  
git checkout feature-branch
```

۱۰۶۰۲ ■ روش دوم: آپلود دستی فایل‌ها

پس از ایجاد مخزن کد، فایل‌های پروژه‌تان را به صورت دستی (مثلاً با drag and drop) به مخزن اضافه کنید.