VJEŽBA 1: RAD S GIT SUSTAVOM ZA KONTROLU VERZIJE DOKUMENATA. UPOZNAVANJE S PROGRAMSKIM JEZIKOM PYTHON.

I. Cilj vježbe:

Upoznati se s korištenjem i prednostima git sustava za kontrolu verzije dokumenata. Upoznati se s programskim jezikom Python i Visual Studio Code IDE.

II. Opis vježbe:

II.1. Git

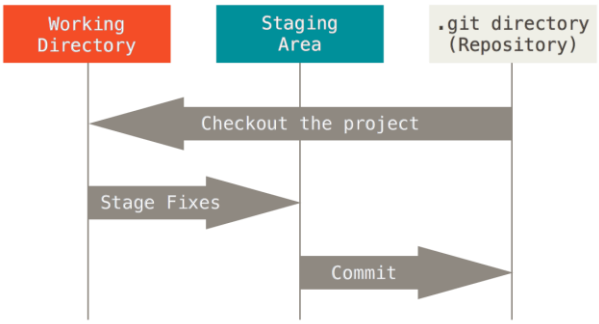
Sustavi za kontrolu verzije dokumenata postali su standard prilikom pisanja različitih dokumenata. U načelu to je softverska podrška za upravljanje promjenama na dokumentima, izvornom kodu, web stranicama i slično. Predstavljaju važnu komponentu prilikom razvoja softvera, pogotovo kada rade timovi programera na istom zadatku (kodu). Postoje centralizirane i distribuirane inačice. U ovoj vježbi radi se s *git* sustavom koji je besplatan distribuirani sustav za kontrolu verzije dokumenata.

Git sustav prati dani direktorij (projekt) na način da kreira „sliku“ direktorija svaki puta kada se određena promjena registrira u *git* sustavu. U *git* sustavu svaka datoteka može biti u jednom od tri moguća stanja:

- Committed – datoteka je uspješno pohranjena u lokalnu bazu
- Modified – datoteka je promijenjena, ali promjene nisu zapisane u bazu
- Staged – datoteka koja je promijenjena je označena za pohranjivanje u bazu

Na taj način postoje tri osnovne sekcije *git* sustava (Sl. 1.1.):

- Git directory – ovdje se pohranjuju meta podaci i baza projekta
- Working directory – ovo je samo jedna verzija projekta (aktualni podaci u direktoriju)
- Staging area – ovo je datoteka koja sadrži promjene datoteka koje su označene za pohranjivanje u bazu (često se naziva i index)



Sl. 1.1. Osnovne sekcije *git* sustava.

Uobičajeno, kako bi se spremio trenutan izgled projekta u bazu potrebno je najprije označiti datoteke čije se promjene žele pohraniti u bazu podataka (stage), a onda upravo i izvršiti pohranjivanje (commit).

Tipičan slijed naredbi bi bio:

```
#inicijalizacija git sustava unutar nekog direktorija koji se želi pratiti
git init

#označavanje datoteka čije promjene želimo pohraniti u bazu
git add  <filename>

#spremanje promjena u bazu
git commit -m "commit message"
```

Ako imate registriran račun na nekoj cloud usluga za pohranu git projekata, onda je potrebno najprije dodati adresu servera:

```
# zapisivanje adrese servera za trenutni projekt; potrebno izvršiti samo jednom
git remote add origin <server>
```

a zatim pohraniti promjene u udaljeni repozitorij:

```
# pohranjivanje promjena u udaljeni repozitorij
git push –u origin master
```

Ako radite s udaljenim repozitorijem, onda je često potrebno osvježiti lokalni projekt (npr. vaš kolega je napravio promjene na udaljenom repozitoriju ili ste vi s nekog drugog računala napravili promjene):

```
git pull
```

te ponoviti postupak za spremanje promjena u bazu. Također moguće je potpuno kopirati sadržaj udaljenog repozitorija na način:

```
git clone <link>
```

pri čemu je `<link>` odgovarajuća adresa. Grananje u *git* sustavu predstavlja kreiranje zasebne grane od glavne (*master*) grane. Kreiranje nove grane naziva `new_branch` i automatsko prebacivanje u granu `new_branch` moguće je izvršiti pomoću naredbe:

```
git branch –b new_branch
```

Prebacivanje iz grane u granu moguće je učiniti pomoću naredbe `checkout`. Na primjer, vraćanje u glavnu granu moguće je učiniti naredbom:

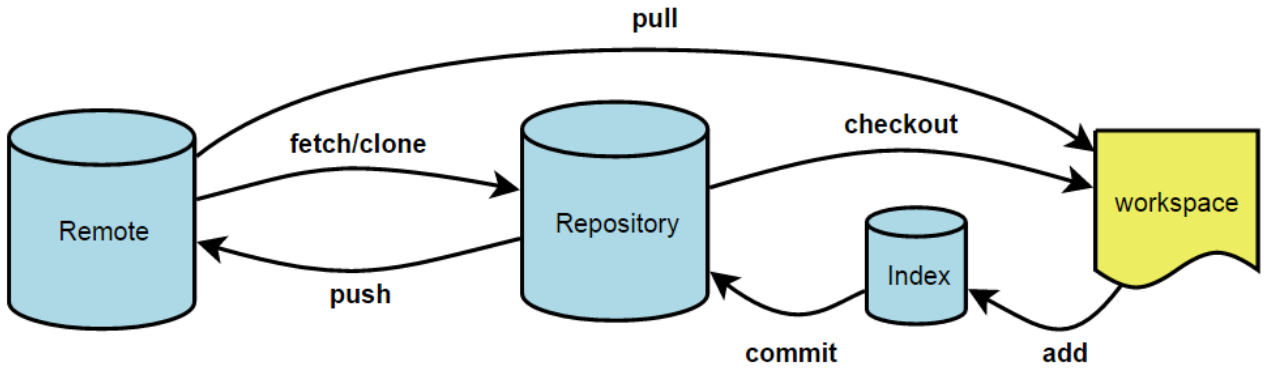
```
git checkout master
```

Kako bi se spojila neke grane s master granom, najprije se treba prebaciti u glavnu granu, a zatim primijeniti naredbu `merge`:

```
git merge new_branch
```

Status repozitorija i povijest promjena možete provjeriti naredbama:

```
git status
git log
```



Sl. 1.2. Rad s *git* sustavom.

Princip rada s *git* sustavom ilustriran je na slici 1.2. Detaljne upute za rad s *git* sustavom moguće je pronaći u materijalima s predavanja

II.2. Python

Programski jezik Python je popularni jezik visoke razine opće namjene. Python interpreter dostupan je za instalaciju na različitim operativnim sustavima. Trenutno aktualne verzije su Python 2.7 i Python 3.9. U ovom predlošku primjeri su napisani za verziju 3.7.
Programiranje u programskom jeziku Python svodi se na pisanje tekstualnih datoteka koje sadrže odgovarajuće programske naredbe, a datoteka ima ekstenziju `.py`. Pokretanje programa moguće je izvesti iz komandnog prozora na način:

```
python ime_skripte.py
```

Varijable u Pythonu mogu biti tipa: `string`, `integer` ili `float`. Ime varijable je proizvoljno pri čemu mogu sadržavati, brojke, slova i underscore (_), ali ne mogu započeti sa brojkom. Osim toga Pythonove ključne riječi (kao npr. `class`) ne mogu se koristiti kao imena varijabli. Dodjeljivanje vrijednosti provodi se pomoću znaka =. Na raspolaganju su i drugi operandi (za zbrajanje, množenje, i sl.).

Primjer 1.1

```
x = 23
print(x)
x = x + 7
print(x)
```

Python podržava standardne Boolove izraze (jednako, različito, veće, manje, …) čije izvršavanje daje vrijednosti `True` ili `False`.

Primjer 1.2

```
x = 23
y = x > 10
print(y)
```

Nadalje, na raspolaganju su funkcije za kontrolu toka programa poput naredbe grananja `if else`.

Primjer 1.3

```
x = 23
if x < 10:
  print("x je manji od 10")
else:
  print("x je veći ili jednak od 10")
```

U Pythonu su na raspolaganju naredbe za petlje `for` i `while`:

Primjer 1.4

```
i = 5
while i > 0:
  print(i)
  i = i - 1
print("Petlja gotova")
```

Primjer 1.5

```
friends = ['Joseph', 'Glenn', 'Sally']
for friend in friends:
  print('Happy New Year:', friend)
print('Done!')
```

Primjer 1.6

```
total = 0
for itervar in [3, 41, 12, 9, 74, 15]:
  total = total + itervar
print('Total: ', total)
```

U Pythonu postoji veliki broj ugrađenih matematičkih funkcija:

Primjer 1.7

```
print(max('Hello world'))
print(len('Hello world'))
```

Primjer 1.8

```
import random
import math

for i in range(10):
  x = random.random()
  y = math.sin(x)
  print('Broj:', x, ' Sin(broj):', y)
```

Prilikom pisanja programa korisnik može i sam definirati funkcije:

Primjer 1.9

```
def print_lyrics():
  print("I'm a lumberjack, and I'm okay.")
  print('I sleep all night and I work all day.')

print_lyrics()
```

Na raspolaganju je niz funkcija za rad sa stringovima odnosno nizom znakova koji se predstavljaju kao polja:

Primjer 1.10

```
fruit = 'banana'
index = 0
while index < len(fruit):
  letter = fruit[index]
  print(letter)
  index = index + 1
```

Primjer 1.11

```
s = 'Monty Python'
print s[6:12]

fruit = 'banana'
print(fruit[0:3])
print(fruit[0:3])
print(fruit[0:-1])

word = 'banana'
count = 0
for letter in word:
  if letter == 'a':
```

```
      count = count + 1
  print(count)
```

Ostale korisne naredbe za rad s stringovima:

Primjer 1.12

```
line = 'Please have a nice day'
if(line.startswith('Please')):
  print 'starts with Please'
elif(line.startswith('please')):
  print 'starts with please'

line.lower()
print line  #stringovi su immutable

data = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
atpos = data.find('@')
print atpos
```

Naredba za rad s datotekama je funkcija `open` koja vraća file handle.

Primjer 1.13.

```
fhand = open('mbox-short.txt')      # www.py4inf.com/code/mbox-short.txt
for line in fhand:
  line = line.rstrip()
  if line.startswith('From:'):
      print line
```

Liste su posebna struktura u Pythonu. Kao što su stringovi znakovni niz, tako je i lista niz veličina koje mogu biti različitog tipa. Liste su za razliku od stringova „mutable“

Primjer 1.14.

```
cheeses = ['Cheddar', 'Edam', 'Gouda']
'Edam' in cheeses

for cheese in cheeses:
  print cheese

a = [1, 2, 3]
b = [4, 5, 6]
c = a + b
print(c)
print(max(c))

t = ['a', 'b', 'c', 'd', 'e', 'f']
print(t[1:3])
print(max(t))

t.append('g')
print(t)

t = ['d', 'c', 'e', 'b', 'a']
t.sort()
print(t)

t.pop(2)
print(t)
```

```
s = 'ovo je neka recenica koju zelimo rastaviti'
t = s.split()
```

Dictionary je također posebna struktura u Pythonu gdje je svaki element key-value par.

Primjer 1.15.

```
eng2sp = dict()
eng2sp['one'] = 'uno'

eng2sp = {'one': 'uno', 'two': 'dos', 'three': 'tres'}
print(eng2sp)
print(eng2sp['two'])
```

III. Priprema za vježbu:

1. Upoznajte se s osnovnim naredbama *git* sustava i programskim jezikom Python prema II. Opis vježbe. Po potrebi koristite i dodatnu literaturu.

IV. Rad na vježbi:

1. Isprobajte Python primjere iz II.2. Python u Visual Studio Code IDE. Razmislite o svakoj liniji programskog koda i što je njen rezultat.

2. Podesite git na osobnom računalu sljedećih pokretanjem naredbi iz git komandnog prozora.

   ```
   git config --global user.name "ime prezime"
   git config --global user.email "email@example.com"
   ```

3. Kreirajte direktorij naziva `PSU_LV`. U ovaj direktorij stavljati ćete sav programski kod koji izradite tijekom vježbi. Inicijalizirajte praćenje ovog direktorija putem `git` sustava:

   ```
   git init
   ```

2. Otvorite korisnički račun na gitlab usluzi za pohranu repozitorija https://about.gitlab.com/. Kako biste mogli raditi push projekata na ovu uslugu potrebno je unijeti public ključ pod postavkama vašeg profila. Pošaljite gitlab odabrano korisničko ime na email predavača.

3. Izradite novi projekt/repozitorij na gitlabu naziva `PSU_LV`. Klonirajte na PC stvoreni projekt naredbom (zamijenite username sa vašim korisničkim imenom):

   ```
   git clone https://gitlab.com/username/PSU_LV
   ```

4. Pozicionirajte se u direktorij `PSU_LV`. Ovdje ćete pohranjivati rješenja s laboratorijskih vježbi. Dodajte `README.md` datoteku u ovaj direktorij. Unutar `README.md` stavite opis repozitorija (npr. Laboratorijske vježbe iz kolegija Primijenjeno strojno učenje, ak.god. 2020./2021.). Provedite postupak pohranjivanje promjena u bazu i u udaljeni gitlab repozitorij (koristite git naredbe `add`, `commit`, `push`). Provjerite na gitlabu imate li novu datoteku u repozitoriju `PSU_LV`.

5. Kreirajte direktorij `LV1` unutar direktorija `PSU_LV`. U ovaj direktorij kopirajte sve datoteke vezane za ovu vježbu, a koje se nalaze na loomen stranici predmeta pod `LV1`.

6. Unutar direktorija `PSU_LV/LV1/` nalazi se python skripta `counting_words.py` s pripadajućim podacima. Napravite odgovarajuće izmjene u python skripti te pohranite promjene u repozitorij. Na kraju pohranite promjene u gitlab repozitorij.

7. Načinite novu granu naziva `python` unutar repozitorija `PSU_LV`. Prebacite se u granu `python`. Unutar direktorija `PSU_LV/LV1/` napravite python skriptu `hello_world.py` koja sadrži jednu naredbu:

   ```
   print("Hello world!");
   ```

   Prebacite se u glavnu granu i napravite spajanje grane s glavnom granom. Pohranite promjene u repozitorij. Pohranite promjene u gitlab repozitorij. Provjerite promjene s naredbom:

   ```
   git log --oneline --decorate --graph -all
   ```

8. Riješite dane zadatke. Odgovarajuće python skripte (ili jedna) s rješenjima trebaju biti pohranjene u direktorij `PSU_LV/LV1/`. Svaki zadatak rješavajte u zasebnoj *git* grani koju spojite s glavnom granom kada riješite pojedini zadatak. Svaki puta kada načinite promjene koje se spremaju u *git* repozitorij napišite i odgovarajuću poruku prilikom izvršavanja `commit` naredbe. Na kraju pohranite promjene i u `PSU_LV` repozitorij na vašem korisničkom računu.

9. Dodajte tekstualnu datoteku `PSU_LV/LV1/Readme.md` s kratkim opisom vježbe i kratkim opisom rješenja vježbe te pohranite promjene u repozitorij.

Zadatak 1

Napišite python skriptu koja od korisnika zahtijeva unos radnih sati te koliko je plaćen po radnom satu. Nakon toga izračunajte koliko je korisnik zaradio i ispišite na ekran.

Primjer:

Radni sati: 35 h
kn/h: 20.5
Ukupno: 717.5 kn

Zadatak 2

Napišite program koji od korisnika zahtijeva upis jednog broja koji predstavlja nekakvu ocjenu i nalazi se između 0.0 i 1.0. Ispišite kojoj kategoriji pripada ocjena na temelju sljedećih uvjeta:

>= 0.9 A
>= 0.8 B
>= 0.7 C
>= 0.6 D
< 0.6 F

Ako korisnik nije utipkao broj, ispišite na ekran poruku o grešci (koristite `try` i `except` naredbe). Također, ako je broj izvan intervala [0.0 i 1.0] potrebno je ispisati odgovarajuću poruku.

Zadatak 3

Prepravite zadatak 1 na način da ukupni iznos izračunavate u zasebnoj funkciji naziva `total_kn`.

Zadatak 4

Napišite program koji od korisnika zahtijeva unos brojeva u beskonačnoj petlji sve dok korisnik ne upiše „`Done`“ (bez navodnika). Nakon toga potrebno je ispisati koliko brojeva je korisnik unio, njihovu srednju, minimalnu i maksimalnu vrijednost. Osigurajte program od krivog unosa (npr. slovo umjesto brojke) na način da program zanemari taj unos i ispiše odgovarajuću poruku.

Zadatak 5

Napišite program koji od korisnika zahtijeva unos imena tekstualne datoteke. Program nakon toga treba tražiti linije oblika:

```
X-DSPAM-Confidence: <neki_broj>
```

koje predstavljaju pouzdanost korištenog spam filtra. Potrebno je izračunati srednju vrijednost pouzdanosti. Koristite datoteke `mbox.txt` i `mbox-short.txt`

Primjer

```
Ime datoteke: mbox.txt
Average X-DSPAM-Confidence: 0.894128046745
Ime datoteke: mbox-short.txt
Average X-DSPAM-Confidence: 0.750718518519
```

Zadatak 6

Napišite Python skriptu koja će učitati tekstualnu datoteku te iz redova koji počinju s „`From`“ izdvojiti mail adresu te ju spremiti u listu. Nadalje potrebno je napraviti dictionary koji sadrži hostname (dio emaila iza znaka @) te koliko puta se pojavljuje svaki hostname u učitanoj datoteci. Koristite datoteku `mbox-short.txt`. Na ekran ispišite samo nekoliko email adresa te nekoliko zapisa u dictionary-u.

V. Izvještaj s vježbe

Kao izvještaj s vježbe prihvaća se web link na repozitorij pod nazivom `PSU_LV`.