SVEUČILIŠTE U RIJECI
FAKULTET INFORMATIKE I DIGITALNIH TEHNOLOGIJA

Preddiplomski studij informatike

Projektni zadatak iz kolegija
Programske paradigme i jezici

Izrada jednostavnog uređivača teksta koristeći Python i Perl

Autor: Dino Ahmičić
Mentori: doc. dr. sc. Miran Pobar
Karlo Babić

U Rijeci, Siječanj 2024.

Sadržaj

1. Uvod

Fokusirajući se na pojedinosti programskih jezika, ovaj rad uspoređuje dvije jednostavne implementacije uređivača teksta – jednu koja koristi programski jezik Perl, i drugu koja koristi Python.

Ova analiza obuhvaća brojne dimenzije, pokrivajući paradigme programiranja, složenosti povezane sa sintaksom i operacijama datoteka kojima se rukuje kroz strategije korisničkog unosa, mehanizme rukovanja iznimkama, strukturu funkcija koje uključuju uzorke koda primjenjene kroz korištenje globalnih varijabli zajedno s uvjetnim konstrukcijama koje pokreću izvršenje glavnog programa.

2. Programske paradigme

Perl, koji se temelji na proceduralnoj i imperativnoj paradigmi, ime tendecije naglašavanja eksplicitnih uputa i postupaka. Jezik promiče modularnost korištenjem potprograma koji omogućuju rješavanje problema u strukturiranom i fleksibilnom okruženju. U našem radu korištena paradigma se može opisati kao proceduralna, sa imperativnim/strukturiranim stilom programiranja.

Prema Larry Wall-u, Perl ima dva slogana. Prvi, „There's more than one way to do it.“ insinuirajući jednostavnost pisanja sažetih izjava. Druga jest, „Easy things should be easy, hard things should be possible.“

Python jezik je vrlo prilagodljiv i uspijeva spojiti imperativnu kao i objektno orijentiranu paradigmu. Unatoč zadržavanju imperativnih struktura, python se fokusira na prakse temeljene na objektima kroz klase i funkcije koje pojačavaju modularnost u ekspanzivnim bibliotekama koda. Što se tiče našeg rada, korištena paradigma se može opisati kao objektno orijentirana,te pomiješana s proceduralnom, sa imperativnim, objektno orijentiranim stilom programiranja.

3. Sintaksa

Perl uvodi koncept prefiksa – posebnih znakova koji označavaju kojem tipu varijabla pripada. Primjerice, „$“ jest skalarna varijabla, „@“ je niz, „%“ asocijativno polje. Ova značajka omogućuje Perl-u da se bavi različitim tipovima podataka na dinamičan način. Sintaksa je kratka i kreativna, sklona je korištenju interpunkcijskih znakova ili simbola.

Perl sintaksa savršeno je prilagođena stvaranju kompaktnog stila kodiranja, koji može izvesti složene operacije u samo nekoliko redaka. Iako ova konciznost može biti izazov za programere koji nisu upoznati s jezikom. Perl koristi regularne izraze koji naglašavaju sposobnosti rukovanja nizovima ovog jezika.

Prefiksi se koriste uz deklaracije varijabli; oni također imaju ulogu pristupa elementima nizova i hash-ova. Ovo specifično svojstvo čini Perl izvrsnim skriptnim jezikom, ali zahtijeva poznavanje radi boljeg korištenja.

```
sub main {
    my $filename = "";
    my @lines;
```

```
chomp $new_line;
last if $new_line eq 'END';
push @$lines, $new_line;
```

S druge strane, Python se uvelike oslanja na čitljivost i jednostavnu sintaksu. Nedostatak prefiksa olakšava deklaraciju varijabli i daje kodu intuitivnu prirodu za nove korisnike. Korištenjem uvlačenja za određivanje blokova koda, Python uklanja upotrebu zagrada ili ključnih riječi. Ovaj sintaktički zahtjev čini Python kod puno čitljivijim od Perl-a.

Liste su glavna struktura podataka za uređene zbirke u Pythonu. Također, modul „OS“ nam omogućava lagano upravljanje datotekama. Izravna upotreba uglatih zagrada „[]“ u definiranju popisa slijedi obvezu jezika prema transparentnosti. Pythonova sintaksa omogućuje korištenje netehničkih riječi u imenovanju varijabli i funkcija, čineći kod sam po sebi razumljivim.

Štoviše, Python slijedi načelo "eksplicitno je bolje od implicitnog", zalažući se za jasnu i nedvosmislenu sintaksu. Ova se filozofija primjenjuje na poruke o pogrešci, što pomaže programerima da razumiju i riješe probleme.

```
global filename, lines
filename = ""
lines = []
```

```
while additional_text.upper() != 'END':
    lines.append(additional_text + '\n')
    additional_text = input()
```

4. Operacije nad datotekama

Perl slijedi standardnu obradu datoteka i funkciju otvaranja zajedno s konceptom korištenja „my $file“ handler-a. Čitanje redaka iz datoteke lako se izvodi preko <STDIN>, čime se pokazuje jednostavan, ali učinkovit način za to.

Slično kao i čitanje iz datoteke, Perl prihvaća korisnički unos kroz <STDIN>. Funkcija chomp omogućuje uklanjanje znakova na kraju novog retka koji pomažu u osiguravanju integriteta podataka.

```
if (open my $file, '<', $filename) {
    @lines = <$file>;
    close $file;
    print "Datoteka uspjesno otvorena.\n";
```

```
if ($filename) {
    print "Unesi broj retka koji mijenjam: ";
    my $line_number = <STDIN>;
    chomp $line_number;
```

Sa funkcijom open(), Python unapređuje operacije s datotekama kako bi pružio upravitelje konteksta, koji su usmjereni na bolje upravljanje resursima. File.readlines() pojednostavio je proces čitanja redaka iz datoteke, prateći većinu aktualnih trendova u programiranju.

```
full_path = os.path.join(script_directory, filename)
with open(full_path, 'r') as file:
    lines = file.readlines()
print("Datoteka uspješno otvorena.")
```

Python pojednostavljuje korisnički unos korištenjem funkcije input() za izravno hvatanje kao niz. Nedostatak eksplicitnog brisanja novog retka jedan je od čimbenika koji čine Python jednostavnim.

5. Rukovanje iznimkama i funkcije

Rukovanje iznimkama/pogreškama u Perlu uključuje if-else konstrukcije koje ovise o uvjetnim izjavama za otporno upravljanje iznimkama. Iako je praktičan, može dovesti do opširnosti u kompliciranim situacijama.

```
if (open my $file, '<', $filename) {
    @lines = <$file>;
    close $file;
    print "Datoteka uspjesno otvorena.\n";
} else {
    print "Error: Datoteku nemoguce otvoriti.\n";
```

Python, pak, implementira strukturiranu metodologiju s blokovima „try-except“ koji omogućuju bolju čitljivost koda i eksplicitnije rukovanje pogreškama. Stoga je usredotočenost na formalno rukovanje pogreškama u skladu s Pythonovim nastojanjima da se održava.

```
try:
    full_path = os.path.join(script_directory, filename)
    with open(full_path, 'r') as file:
        lines = file.readlines()
    print("Datoteka uspješno otvorena.")
except FileNotFoundError:
    print("Error: Nemoguce otvoriti datoteku.")
```

Što se tiče funkcija, Perl deklarira funkcije koristeći termin „sub“, naglašavajući modularnost koda. Nizovi se prosljeđuju referencom, proomičući učinkovitost u korištenju memorije.

Python definira funkcije koristeći termin „def“, potičući modularne strukture koda koje se mogu ponovno koristiti.

```
def save_file():
    global filename, lines
    if filename:
        with open(filename, 'w') as file:
            file.writelines(lines)
        print("Datoteka uspješno spremljena.")
    else:
        print("Error: Datoteka trenutno nije otvorena.")
```

6. Uvjetna grananja

Perl koristi „if – elsif - else“ kao konstrukciju za donošenje odluka, pružajući prilagodljivost, ali potencijalno rezultira opširnošću za složene uvjete.

Python, na drugu ruku, pojednostavljuje situaciju koristeći „if – elif- else“ nudeći malo čišću i intuitivniju sintaksu, što u konačnici doprinosi čitljinosti koda i lakoći održavanja.

```
elsif ($choice eq '2') {
    print "Unesi ime datoteke: ";
    $filename = <STDIN>;
    chomp $filename;

    if (open my $file, '<', $filename) {
        @lines = <$file>;
        close $file;
        print "Datoteka uspjesno otvorena.\n";
    } else {
        print "Error: Datoteku nemoguce otvoriti.\n";
    }
    print "_____\n\n";
}
elsif ($choice eq '3') {
    if ($filename) {
        if (open my $file, '>', $filename) {
            print $file join("\n", @lines);
            close $file;
            print "Datoteka uspjesno spremljena.\n";
        } else {
            print "Error: Spremanje datoteke neuspjesno.\n";
        }
    } else {
        print "Error: Datoteka trenutno nije otvorena.\n";
    }
    print "_____\n\n";
```

```
choice = input("Unesi odabir: ")
print("_____\n\n")

if choice == '1':
    create_new_file()
    print("_____\n\n")

elif choice == '2':
    open_file()
    print("_____\n\n")

elif choice == '3':
    save_file()
    print("_____\n\n")
elif choice == '4':
    add_text_to_file()
    print("_____\n\n")
elif choice == '5':
    edit_file()
    print("_____\n\n")
elif choice == '6':
    read_file()
    print("_____\n\n")
elif choice == '7':
    print("Izlaz....")
    print("_____\n\n")
    break
else:
    print("Krivi odabir. Pokušaj ponovno.")
    print("_____\n\n")
```

7. Zaključak

U konačnici, i Perl i Python skripte izvode slične temeljne aktivnosti, ali se razlikuju u specifičnostima implementacije, paradigmi, dok je programski stil drugačiji. Perl skripta strogo slijedi proceduralni stil programiranja s eksplicitnim deklaracijama varijabli i naglaskom na uvjetne izjave i petlje. U usporedbi, Python skripta integrira i objektno orijentirane i proceduralne značajke pomoću implicitnih deklaracija varijabli, modula za modularnost uz kraću sintaksu.

Dvije se opcije razlikuju po preferencijama programera, potrebama projekta i zahtjevima za čitljivost, te lakoću održavanja. Mješovita paradigma i čišća sintaksa Python skripte mogle bi je učiniti privlačnijom programerima koji teže kompromisu između objektno orijentiranog ili proceduralnog stila programiranja. U isto vrijeme, ljubitelji proceduralizma mogli bi više voljeti Perl skriptu zbog svoje eksplicitnosti i strukture.

Reference

n.d. *CodeConvert.* https://www.codeconvert.ai/python-to-perl-converter.

n.d. *OS - Miscellaneous operating system interfaces.* https://docs.python.org/3/library/os.html.

n.d. *Perl.* https://www.perl.org/docs.html.

n.d. *Perl Tutorial - Perl Read FIle.* https://www.perltutorial.org/perl-read-file/.

n.d. *Perthon - Python to Perl Language Translation.* https://perthon.sourceforge.net/.

Porters, Perl 5. n.d. *Perldoc Browser* . https://perldoc.perl.org/.

n.d. *Python Documentation.* https://docs.python.org/3/.

n.d. *Stack overflow.* https://stackoverflow.com/.

Szabo, Gabor. n.d. *Perl Maven - Open and read from text files.* https://perlmaven.com/open-and-read-from-files.

thibaultduponchelle. n.d. *Github - TryPerl.* https://github.com/thibaultduponchelle/tryperl.