Uporabniška dokumentacija

Uporabniška dokumentacija je napisana za različico: `Commit # 4b3203a`. Ta različica je na voljo na: `https://github.com/jakobb01/BinCovering`.

3.1 Začetek uporabe

Sistemske zahteve

- Sodoben spletni brskalnik (Chrome, Firefox, Safari, Edge)
  Dodatne zahteve za možnost 1:
- Python 3.7 ali novejši
- Git

Možnost 1: Lokalna namestitev

1. **Klonirajte ali prenesite projekt** na vaš lokalni računalnik

```
git clone https://github.com/jakobb01/BinCovering.git
cd web_interface/
```

2. **Namestite potrebne odvisnosti**:

```
pip install flask
```

3. **Zaženite aplikacijo**:

```
python app.py
```

4. **Odprite brskalnik** in pojdite na naslov `http://localhost:5000`

Možnost 2: Dostop preko spletnega brskalnika

1. **Odprite brskalnik** in pojdite na naslov `http://4xww0fuh2a1u96pm.myfritz.net:5000/`

Pregled vmesnika

Spletni vmesnik je sestavljen iz treh glavnih področij:

- **Levi panel**: Brskalnik po datotekah s shranjenimi skriptami in dnevniki.
- **Sredinski panel**: Urejevalnik kode s poudarjanjem skladnje.

- **Desni panel**: Terminalski vmesnik za izvajanje ukazov.
- **Spodnji panel**: Vizualizacija skripte, ki prikazuje njeno strukturo.

3.2 Uporaba urejevalnika kode

Osnovne operacije

- **Pisanje kode**: Kliknite v urejevalnik in začnite pisati Python kodo.
- **Poudarjanje skladnje**: Urejevalnik samodejno poudarja skladnjo jezika Python.
- **Preklop teme**: Uporabite preklopnik za temo (zgoraj desno) za preklop med svetlim in temnim načinom.

Shranjevanje skript

1. Napišite kodo v urejevalnik.
2. Uporabite funkcionalnost za shranjevanje (na sredini spodaj) za shranjevanje skript v določene kategorije:
   - **Generate Bins** - skripte za generiranje predmetov.
   - **Strategies** - algoritmi za pokrivanje posod.
   - **Custom Code** - razne skripte, skripte za avtomatizacijo, splošni nameni itd.

Nalaganje skript

- Brskajte po razpoložljivih skriptah po kategorijah z uporabo brskalnika datotek.
- Kliknite na katerokoli skripto, da jo naložite v urejevalnik.
- Skripte so organizirane v drevesni strukturi map.

3.3 Terminalski vmesnik

Ukazi na voljo

- `help`: Prikaže seznam razpoložljivih ukazov in njihovih opisov.
- `run`: Izvede kodo, ki je trenutno v urejevalniku.
- `<skripta> <argumenti>`: Izvede katerokoli shranjeno skripto z argumenti.

Izvajanje skript

Izvedite shranjene skripte tako, da vpišete njihovo ime:

```
generate_random 100 50    # Generira 100 predmetov z največjo velikostjo 50
DualNextFit data/items.txt # Zažene algoritem Dual Next Fit
```

Primeri ukazov

```
# Generiraj naključne predmete in jih shrani v datoteko
generate_random 50 100

# Izvedi kodo, naloženo v urejevalniku
run
```

3.4 Delo z generatorji in strategijami

Ustvarjanje generatorjev predmetov

Generatorji predmetov dedujejo od osnovnega razreda Generate in morajo implementirati naslednje metode: start(), next(), stop().

Primer strukture za generator:

```
from generate import Generate

class MyGenerator(Generate):
    def __init__(self):
        super().__init__()

    def start(self):
        # Vaša logika za začetek tukaj
        pass

    def next(self):
        return 1

    def stop(self):
        return f"generator uststopped"
```

Uporaba vgrajenih generatorjev

Sistem vključuje dva pred-izdelana generatorja:

- generate_random

- ItemGenerator

Primer:

```
generate_random 100 10  # Generira 100 predmetov z največjo velikostjo 10
```

Ustvarjanje strategij za pokrivanje košev

Strategije dedujejo od osnovnega razreda `Strategy` in implementirajo specifične algoritme za pakiranje v koše. Vsaka `strategija` mora implementirati naslednje metode: `start()`, `next()`, `stop()`.

Vgrajene strategije:

- Harmonic
- DualNextFit

Izvajanje strategije

1. **Pripravite podatke** - uporabite generator za ustvarjanje podatkov o predmetih.
2. **Zaženite strategijo** - izvedite strategijo z datoteko s podatki.
3. Oglejte si rezultate - preverite izpis v terminalu ali dnevnike za rezultate.

Primeri ukazov:

```
# Korak 1: Generiranje testnih podatkov
generate_random 100 10

# Korak 2: Zaženite strategijo (uporabite ime generirane datoteke)
harmonic data/items_100_80_timestamp.txt
```

3.5 Razvoj

Kadar želite pisati lastne strategije ali generatorje, bodite pozorni na:

Uvažanje modulov

Lahko uvozite in uporabite osnovne razrede:

```
from generate import Generate
from strategy import Strategy
```

Struktura datotek

- **Data** - za generirane nabore podatkov.
- **Strategy** - za implementacije algoritmov.
- **Generator** - za generatorje predmetov.

3.6 Vizualizacija skripte

Razumevanje vizualizacije

Panel za vizualizacijo skripte prikazuje:

- **Uvozi**: zunanji moduli in lokalni uvozi.
- **Razredi**: definirani razredi z informacijami o dedovanju.
- **Metode**: funkcije in metode razredov.
- **Spremenljivke**: pomembne priredbe spremenljivk.

3.7 Odpravljanje težav

Skripta se ne izvede

- **Preverite skladnjo**: zagotovite pravilno Python skladnjo.
- **Preverite uvoze**: prepričajte se, da so vsi potrebni moduli na voljo.
- **Preverite poti do datotek**: preverite, ali podatkovne datoteke obstajajo na pravilni lokaciji.

Terminal se ne odziva

- **Osvežite stran**.

Napake "Datoteka ni najdena"

- **Preverite imena datotek**: zagotovite pravilno črkovanje in pot.
- **Preverite lokacijo datoteke**: datoteke morajo biti v direktoriju `data`.

Težave z zmogljivostjo

- **Zmanjšajte velikost nabora podatkov** pri uporabi javne aplikacije.

Ste našli hrošča?

- Prijavite ga na moj e-mail: `89242105@student.upr.si`