Základy jazyka Python

web scraping, GUI, typová kontrola

prednáška 11

Katedra kybernetiky a umelej inteligencie
Technická univerzita v Košiciach
Ing. Ján Magyar

Web scraping

- process získavania informácií z internetu
- zvyčajne sa pod tým myslí automatizovaný scraping
- web stránky sú rôznorodé – špecifická aplikácia
- web stránky sa často menia – je potrebné často aktualizovať riešenie
- môže byť nelegálne!

Získavanie html kódu web stránky

- cez modul `urllib/requests`
- funguje iba pre statické stránky
- pre dynamické stránky (napr. vykonávanie javascript skriptov) môžete použiť `requests-html` alebo použiť Selenium

BeautifulSoup

- modul pre parsovanie html súborov
- súbor reprezentuje ako stromovú štruktúru
- môžeme prechádzať prvkami ako stromom (rodičovské uzly, potomkovia) alebo vyhľadávať podľa rôznych atribútov prvkov
- pred implementáciou riešenia musíme analyzovať zdrojový kód web stránky

Selenium

- najmä pre testovanie webových stránok
- získanie kódu dynamických web stránok
- umožňuje ovládať webový prehliadač pomocou kódu
- je potrebné nainštalovať driver pre podporu prehliadača a modul `selenium`
- niektoré stránky ho môžu blokovať
- je umožnená interakcia s web stránkou
    - zadávanie vstupov do formulárov
    - posielanie formulárov
    - kliknutie na tlačidlá

Grafické rozhrania v Pythone

- najčastejšie používané moduly
  - Tkinter
  - PyQt
  - wxPython

Tkinter

- štandardný modul pre GUI v Pythone
- súčasťou Python distribúcie
- funguje na Windowse, Mac OS, niektorých Unix distribúciach
- nepodporuje natívny vzhľad
- iba pre jednoduché aplikácie
- bez editora designu (WYSIWYG)

wxPython

- rozšírenie Pythonu
- podpora Windowsu, Mac OS, Linux, Unix-distribúcií
- natívny vzhľad na viacerých platformách bez modifikácie kódu
- stále pomerne malý počet widgetov
- bez WYSIWYG editora

PyQt

- wrapper pre Qt softvérový rámec
- podobné verzie existujú pre ďalšie jazyky s podobným API – veľká komunita
- najväčší počet widgetov a funkcionalít
- súčasťou anacondy
- dosť zložitý modul
- Qt Designer pre WYSIWYG editovanie vzhľadu

Metódy a triedy ako premenné

- v Pythone každá metóda a trieda je zároveň aj objektom, vieme s nimi narábať ako s hodnotami
- metódy a triedy vieme uložiť do premenných a následne ich používať cez danú premennú

```
def my_func():
    return 5
```

```
result = my_func()
result ← 5
```

```
result = my_func
result ← <function my_func>
```

Metódy ako premenné

- ak v čase písania kódu nevieme, ktorá funkcia sa má zavolať
- zvyčajne to používame, ak chceme používateľom umožniť prepínať medzi viacerými funkcionalitami (napr.: rôzne testy pri simuláciach)
- vieme to urobiť v rámci jednej funkcie, pomocou vetvenia
- ak funkcie uložíme do premennej, mali by mať približne rovnaké rozhranie API
    - rovnaký počet a typ parametrov
    - rovnaká návratová hodnota

Triedy ako premenné

- ak v čase písania kódu nevieme, z ktorej triedy chceme vytvoriť inštanciu
- napr.: máme niekoľko výpočtových modelov, ktoré definujú špecifikáciu toho istého hlavného konceptu, a chceme používateľom umožniť, aby si vybrali konkrétnu špecifikáciu počas behu programu (podtriedy `Drunk` v simulácii biased random walk)
- ekvivalent je využiť návrhový vzor abstract factory, resp. builder
- API rozhranie tried musí byť rovnaké
  - definovať rovnaké metódy s rovnakými parametrami a návratovými hodnotami
  - definovať rovnaké atribúty
  - triedy môžu definovať funkcionalitu navyše, ale zdieľaná časť funkcionality by mala byť rovnaká

Type hinty v Pythone

- dynamicky typovaný jazyk, interpreter neumožňuje statické typovanie, avšak niektoré IDE môžu poskytovať istú mieru podpory
- pomocou type hintov dokážeme naznačiť typ premenných, parametrov a návratovej hodnoty
- najmä pre účely dokumentácie
- väčšia podpora autosuggestu v IDE
- lepšia softvérová architektúra
- pomalší kód
- časovo náročné
- type hinty sa odporúča použiť vo verejných knižniciach a v tímových projektoch
- môžu pomôcť pri testovaní

Zapnutie typovej kontroly

- definovaná v PEP 484
- cez lintery
- v PyCharme dostupná
- v SublimeTexte cez linter `SublimeLinter-contrib-mypy`
- vo Visual Code napr. Python Type Hint
- platformovo-nezávislé riešenie: knižnica `mypy`

Práca s mypy

- zoznam hintov cez __annotations__
- získanie typov: reveal_type()
- získanie typov atribútov v mennom priestore: reveal_locals()
- spustenie skriptu cez `mypy nazov.py`

Zložitejšie typy

- cez modul `typing`
- definícia type aliasov – ukladanie zápisov v premenných
- podpora základných typov: `Dict`, `List`, `Tuple`
- ďalšie typy: `Deque`, `FrozenSet`, `Set`
- je možné použiť aj vlastné metódy ako typy
- `NewType` pre vlastné typy, `TypeVar` pre typovú premennú
- všeobecné typy:
  - `Sequence` – ľubovoľná sekvencia
  - `Any` – ľubovoľný typ
  - `Optional` – pre nepovinné parametre
- pracuje sa na protokoloch pre typy so špecifickou vlastnosťou (napr. `Sized`)
- pre funkcie môžete použiť `Callable`

Menný priestor

- menný priestor mapuje mená na objekty a hodnoty
- v Pythone je menný priestor implementovaný ako dictionary
- menný priestor môže obsahovať
  - defaultné mená (kľúčové slová, funkcie, výnimky)
    - vytvorí sa pri spustení interpretera, zanikne po skončení programu
  - globálne mená v moduloch
    - vytvorí sa pri importe, zanikne po skončení programu
  - lokálne mená vo funkciách
    - vytvorí sa pri spustení funkcie, zanikne po skončení vykonávania

Oblasť - scope

- oblasť definuje časť programu, z ktorej je daný menný priestor priamo dostupný
- oblasť slúži na vyhľadávanie mien počas behu programu:
  - vnútorná oblasť
  - zapuzdrujúca oblasť
  - globálne názvy v module
  - vonkajšia oblasť
- nové atribúty sa vytvoria defaultne v rámci lokálnej oblasti

Global a nonlocal

- Python umožňuje určiť oblasť pre vytvorené atribúty
- `global`
  - atribút bude dostupný z oblasti modulu (nie z interpretera!)
- `nonlocal`
  - upravuje atribúty, ktoré sú dostupné z inej ako vnútornej oblasti
  - bez `nonlocal` sa vytvorí nový atribút s rovnakým názvom
  - bez `nonlocal` budú atribúty read-only

Vnorené funkcie

- Python umožňuje definíciu funkcie v tele inej funkcie
- vnútorná funkcia bude dostupná iba z vonkajšej funkcie
- ideálne pre riešenie podúlohy, ak podúloha sa má riešiť iba v rámci jednej funkcie

Closure

- špeciálny prípad vnorených funkcií
  - musíme mať vnorenú funkciu (funkciu definovanú v tele inej funkcie)
  - vnorená funkcia musí pracovať s hodnotou definovanou vo vonkajšej funkcii
  - vonkajšia funkcia musí vrátiť vnútornú funkciu
- telo, resp. atribúty vnútornej funkcie existujú aj po vymazaní vonkajšej funkcie
- umožňuje nám predísť použitiu globálnych premenných, podporuje zapuzdrenie, a generuje funkcie (metaprogramovanie - dekorátory)