Akademia Górniczo - Hutnicza
im. Stanisława Staszica w Krakowie

Teoria kompilacji i kompilatory

WAG - working and grounded

Paweł Kowal
Mikołaj Margański
Jakub Natonek
Aleksandra Wołowiec

Informatyka, 3 rok, WEAIIIB

1. Założenia projektu i przeznaczenie

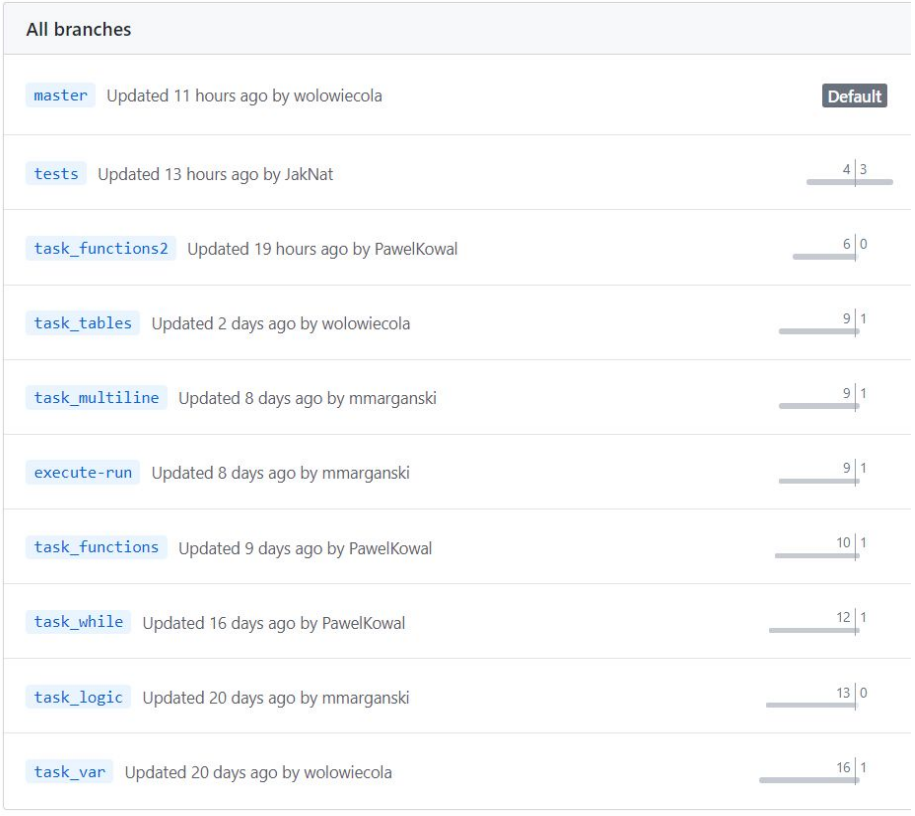
Celem projektu jest stworzenie własnego, interpretowanego języka programowania przy użyciu technologii Python. Prawdopodobnie nie będzie on równie dobry i szybki jak już istniejące języki, ale na pewno poszerzy naszą wiedzę i umiejętności programistyczne.

Jak sama nazwa wskazuje, chcemy aby język był działający i dostosowany do naszych realiów. Planujemy uzyskać to w następujących krokach:
1) Stworzenie Lexera, za pomocą którego zostanie przeprowadzona analiza leksykalna, dzielenie tekstu na tokeny
2) Stworzenie Parsera, za pomocą którego zostanie przeprowadzona analiza składniowa, zamiana listy tokenów w drzewo węzłów (AST)
3) Stworzenie Interpretera, który przy pomocy drzewa węzłów wykona program wejściowy
4) Planowana składnia języka - chcemy aby była prosta i funkcjonalna. Składnia będzie rozwijana wraz z powstawaniem języka.

---

2. Podział zadań

- ❏ PM - Aleksandra Wołowiec
- ❏ Tester - Jakub Natonek
- ❏ Programista - wszyscy



3. Inicjalizacja projektu

Projekt wymaga zainstalowanego `Pythona`.
Projekt należy sklonować lub pobrać z repozytorium.

git clone https://github.com/olawolowiec/WAG.git

Następnie przechodzimy w terminalu do folderu WAG i przy pomocy komendy:

```
python shell.py
```

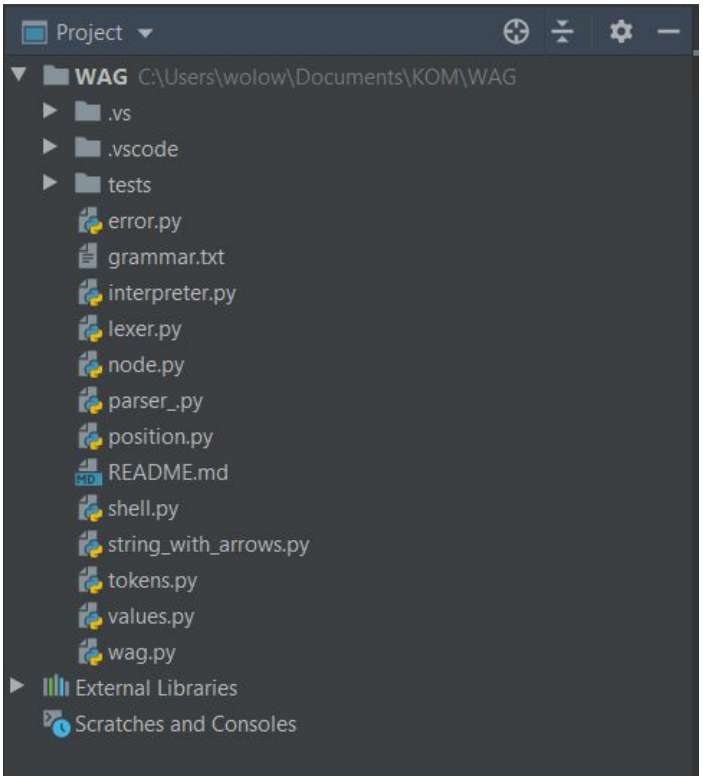
uruchamiamy go. Projekt można również uruchomić bezpośrednio z IDE obsługującego `Pythona`, wybierając **`shell.py`** jako plik startowy.

Po uruchomieniu w zobaczymy w konsoli następujący prompt:

```
wag>
```

gdzie możemy wpisywać kod w języku `WAG`.

4. Struktura projektu



5. Składnia języka

Naszym celem było stworzenie języka programowania w języku polskim przystępnego dla osób, które chcą poznać podstawy programowania. Z tego powodu składnia jest zbliżona do klasycznych języków programowania, lecz słowa kluczowe są w języku polskim. Prezentuje się ona następująco:

- przykłady działań arytmetycznych:

```
4+6, 5.0-2.3, 2*2.7, 5/4, 2^4
```

- deklaracje zmiennych:

```
ZMIENNA a = 1
ZMIENNA b = "abc"
ZMIENNA c = [1,2,3]
```

- porównania i operacje logiczne:

```
(2>=4 ORAZ false) LUB (2!=3 ORAZ true)
```

- pętle:

```
ZMIENNA a=1
DOPÓKI a<5 DOPÓTY ZMIENNA a=a+1


DLA i=0 DO 10 WYKONAJ PRINT(i)
```

- instrukcje warunkowe:

```
JEŻELI a>b WYKONAJ true BĄDŹ a<b WYKONAJ false PRZECIWNIE PRINT("a i b są równe")
```

- funkcje:

```
TEZA dodaj(a,b): a+b
dodaj(2,1)
>3
```

- multiline:

```
TEZA test();
ZMIENNA foo=5;
PODSUMOWUJĄC foo;
CO_KOŃCZY_DOWÓD
test()
>5
```

Pełna gramatyka znajduje się w pliku **`grammar.txt`** w folderze projektu.

6. Obsługiwane operacje

- ❏ matematyczne - dodawanie, odejmowanie, mnożenie, dzielenie, potęgowanie
- ❏ deklaracja zmiennych
- ❏ operatory porównania
- ❏ operatory logiczne
- ❏ pętle while, for
- ❏ funkcje
- ❏ instrukcje warunkowe
- ❏ listy
- ❏ multiline
- ❏ odczyt z pliku

7. Testy

Wykonano testy integracyjne dla każdej funkcjonalności języka.
Testy opierają się na uruchamianiu języka z podanym wejściem a następnie porównywanie wyniku z odpowiednim elementem zwróconej listy(‘sterty’)

8. Lista wykorzystywanych technologii

Do napisania prostego interpretera został wykorzystany język Python, ponieważ jest intuicyjny i wygodny dla osób realizujących projekt.

9. Lista narzędzi użytych w projekcie

- **Repozytorium**

  Github - serwis internetowy wykorzystujący system kontroli wersji Git. Jest popularnym i wygodnym narzędziem, o przyjaznym dla użytkownika interfejsie, znanym osobom realizującym projekt.

- **IDE**

  ### Visual Studio Code

  Darmowy edytor Visual Studio Code obsługuje wiele języków programowania w tym języka Python. Oprogramowanie posiada wsparcie dla debugowania kodu, zarządzania wersjami kodu źródłowego za pośrednictwem systemu kontroli wersji Git, automatycznego uzupełniania kodu intelliSense oraz jego refaktoryzacji. Oferuje liczne wtyczki i rozszerzenia ze snippetami znacznie przyspieszające pisanie kodu.

  ### Pycharm

  Do jego głównych zalet należą - współpraca z systemem kontroli wersji git, ogrom skrótów klawiszowych pozwalających na pracę praktycznie bez używania myszy. Nie jest to co prawda IDE pozbawione wad - tutaj można wymienić m.in. potężne zużycie pamięci czy długie indeksowanie projektu - to jednak naszym zdaniem warto go używać, bo można w ten sposób znacznie zwiększyć wydajność pracy programisty

10. Możliwe funkcjonalności do dodania podczas przyszłej rozbudowy języka

- komentarze
- switch-case
- złożone typy danych
- operacje na plikach
- generowanie liczb pseudolosowych
- tworzenie i wykorzystanie klas

Mar 29, 2020 – Jun 18, 2020

Contributions: **Commits** ▾

Contributions to master, excluding merge commits