**Projekt Kalkulatora**

****

**Narzędzia Pracy grupowej**

*Automatyka i Robotyka*

*Kraków, 2022 r.*

*Dawid Wiktorowski*

*Jakub Styczeń*

*Jakub Tomczyk*

*Jakub Żak*

1. **Cel Projektu**

Celem projektu jest stworzenie programu kalkulatora mającego spełniać następujące funkcje:

1. Przeprowadzanie działań: dodawanie, odejmowanie, mnożenie, dzielenie, potęgowanie, pierwiastkowanie.
2. Pamięć ostatnich 10 zadań.
3. Odwołanie się do zapamiętanego zadania.
4. Czyszczenie pamięci.
5. Obsługa liczb zespolonych.
6. Jedna zaproponowane przez grupę.
7. **Plan projektu:**
8. Ustalenie elementów niezbędnych w procesie powstawania projektu.
9. Utworzenie zespołów oraz projektów na platformie Asana oraz GitHub w celu ułatwienia komunikacji i konstrukcji projektu, umożliwiając szybszą realizację zadanych celów.
10. Utworzenie zdalnego repozytorium w GitHubie (stworzenie systemu umożliwiającego kontrolę wersji).
11. Podział pracy i przydzielenie ról w zespole.
12. Utworzenie dokumentu tekstowego w celu napisania dokumentacji projektu.
13. Stworzenie skryptu kalkulatora w Pythonie.
14. Testowanie wszystkich funkcjonalności programu.
15. Poprawa ewentualnych błędów.
16. Udokumentowanie pracy włożonej w projekt.
17. Oddanie projektu w pełni funkcjonalnego kalkulatora.
18. **Elementy niezbędne**

Jednym z podstawowych kroków w procesie wykonywania projektu jest ustalenie elementów niezbędnych do jego realizacji. W jego trakcie musieliśmy zastanowić się nad dostosowaniem wymagań do dostępnych środków i czasu potrzebnego na realizację projektu. Dodatkowo, rozróżnić rzeczy kluczowe od tych zbędnych i niepotrzebnie wydłużających czas pracy.

Celem projektu jest utworzenie programu kalkulatora, do jego realizacji w kilkuosobowej grupie niezbędny jest system pozwalający kontrolować ten proces przez inne osoby. Najprostszym i najszybszym sposobem jest realizacja projekty za pomocą systemu git oraz platformy GitHub. Innym z kluczowych elementów jest komunikacja w zespole, którą ułatwi nam platforma Asana. Następnym krokiem jest wybranie języka programowania, który będzie posiadał funkcje pozwalające nam spełnić wszystkie wymagania projektu, a dodatkowo będzie najprostszym dla nas w obsłudze. Dlatego językiem, który wybraliśmy jest Python (wersja z której korzystaliśmy to Python 3.9.10) zawierający wszystkie funkcje kluczowe przy tworzeniu kalkulatora bez potrzeby pobierania dodatkowych bibliotek.

1. **Podział obowiązków**

Po utworzeniu zespołów w wymienionych platformach zajęliśmy się przydzieleniem ról każdemu z nas. W celu przyspieszenia procesu tworzenia projektu podzieliliśmy się na dwa dwuosobowe zespoły. Dwóch z nas zajęło się obmyślaniem sposobu działania kalkulatora oraz pisaniem programu w Pythonie (Jakub Styczeń i Dawid Wiktorowski). Druga para zajęła się natomiast testowaniem działania kalkulatora i tworzeniem dokumentacji projektu, opisując działania przez nas wykonywane (Jakub Tomczyk i Jakub Żak). Dzięki tak przyjętemu podziałowi pracy możemy lepiej zorganizować system naszej pracy bez wprowadzania zbędnego chaosu oraz lepiej kontrolować ten proces.

1. **Proces tworzenia**

Projekt został zdefiniowany przez naszego prowadzącego przedmiotu Narzędzia Pracy Grupowej dr inż. Waldemara Bauera. Na początku zaplanowaliśmy jak nasz kalkulator ma wyglądać i jakie będą jego funkcje (5 z nich było z góry narzucone, a od siebie dodaliśmy obsługę logarytmów). Nasi programiści zabrali się za realizację głównego tematu projektu, a pozostali testowali program i tworzyli dokumentację. Do komunikacji wykorzystaliśmy Asanę, dzięki czemu mogliśmy w jednym miejscu zamieszczać i przydzielać zadania, a w komentarzach zwracać uwagę na ewentualne błędy i określać dalsze kroki. Przez czas tworzenia projektu zorganizowaliśmy również trzy krótkie spotkania, na których każdy opowiadał, co udało mu się osiągnąć, czego nie udało się zrobić oraz o tym, co chce zrobić w najbliższym czasie. Wszyscy również monitorowaliśmy się nawzajem i rozmawialiśmy jeśli czyjaś praca nam się nie podobała. Ustaliliśmy sobie kilka deadline’ów co do tworzenia programu i dokumentacji, jednak nie zawsze każdemu udawało się ich przestrzegać. Gdy uznaliśmy, że program jest gotowy, stworzyliśmy plik.exe i uznaliśmy projekt za zakończony.

1. **Funkcje użyte w programie**

W tym rozdziale zajmiemy się opisem działania i celem zastosowania poszczególnych funkcji oraz komend w naszym kodzie.



- importuje moduł wyrażenia regularnego, czyli pewnego rodzaju "wzorców", za pomocą których można opisać ciąg znaków;

 - importuje bibliotekę zawierającą niezbędne do stworzenia kalkulatora funkcje matematyczne (np.: pierwiastek);

- określa formę w jakiej występuje równanie, dzięki czemu program może zauważyć błąd w trakcie wpisywania danych (np.: po wpisaniu w konsolę znaku, program sprawdzi jego zgodność z ustalonym wzorem i zwróci błąd w przypadku nieuwzględnienia go w warunku równania), w tym przypadku widzimy, że program dopuszcza jedynie liczby (dodatnie i  ujemne) wymierne oraz zespolone z oznaczeniem „j”;

Obraz zawierający tekst, zegar, pomarańczowy, ciemny

Opis wygenerowany automatycznie- klasa tworząca kolejkę do przechowywania w pamięci wyniki 10 ostatnich działań;

- inicjalizuje powstanie pustej listy pozwalającej na odwoływanie się do samej siebie;

Obraz zawierający tekst, pomarańczowy

Opis wygenerowany automatycznie - wpisuje na pierwsze miejsce kolejki nowo uzyskany wynik;

Obraz zawierający tekst

Opis wygenerowany automatycznie- usuwa z kolejki najstarszy wynik robiąc miejsce w pamięci dla najnowszego wyniku;

Obraz zawierający tekst, pomarańczowy

Opis wygenerowany automatycznie- Zwraca długość kolejki;

Obraz zawierający tekst, pomarańczowy

Opis wygenerowany automatycznie

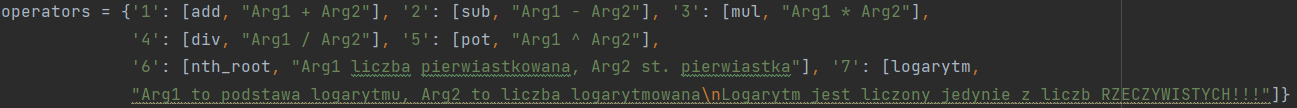
- Umożliwia iterowanie w kolejce po kolejnych elementach kolejki;

Obraz zawierający tekst

Opis wygenerowany automatycznie- Tworzy funkcję pozwalającą wprowadzić użytkownikami liczby do równania spełniające wcześniejsze warunki z funkcji pattern (zwracając błąd w wypadku niezgodności i  komentarz o poprawnym formacie danych);

Obraz zawierający tekst

Opis wygenerowany automatycznie- Tworzy funkcję pozwalającą użytkownikowi wprowadzić tylko określone w kodzie wartości automatycznie sprawdzając zgodność wpisanych w konsoli danych z tymi wpisanymi w string (rozumiemy to jako zgodność z  danymi dozwolonymi w tekście wyświetlanym podczas wpisywania danych w konsolę);

- Tworzy słownik przypisujący liczby danym operacją, tak aby łatwiejsze było wpisywanie potrzebnych działań w konsolę. Dodatkowo wyświetla tekst pomagający w poprawnym wpisaniu działania, tak aby użytkownik wiedział np.: który z argumentów odpowiada za stopień pierwiastka, a który za liczbę pod nim;



- pozwala określić użytkownikowi działanie jakie chce wykonać (zgodne z funkcją validInputString, czyli tylko z 7 możliwych działań określonych w komentarzu);

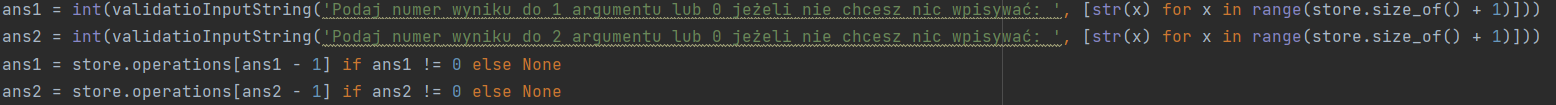


- Wprowadza warunek umożliwiający w dalszej części wywołanie elementów z pamięci kalkulator (gdy jest ona większa od 0);

Obraz zawierający tekst

Opis wygenerowany automatycznie

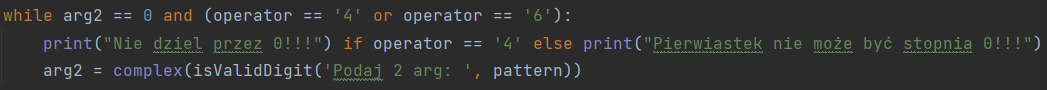
- Daje możliwość wykorzystania wyników działań z pamięci kalkulatora, a w wypadku wybrania tej opcji indexuje elementy z pamięci kalkulatora i wypisuje je w konsoli;



- Pozwala użytkownikowi wpisać numer działania z pamięci (1- najnowsze; 10 – najstarsze) a  w wypadku wstawienia „0” prowadzi do wpisania argumentu ręcznie. Została tutaj wykorzystana wcześniej opisana funkcja validInputString do wpisywania jedynie określonych wcześniej wartości;

Obraz zawierający tekst

Opis wygenerowany automatycznie- Pozwala wprowadzić argumenty odwołując się do wcześniej opisanej funkcji isValidDigit, czyli tylko te zgodne z formułą zapisaną w warunku pattern;

-Sprawdza warunek wprowadzenia „0” jako drugi argument dzielenia oraz pierwiastka (liczba w mianowniku oraz stopień pierwiastka), zwracając komunikat o błędzie oraz dając możliwość poprawienia wprowadzonej wartości na zgodną z wymaganiami;

Obraz zawierający tekst

Opis wygenerowany automatycznie

- Sprawdza poprawność argumentów w logarytmie. Jeżeli podstawa jest dodatnia lub równa „1” program wyświetla komunikat o błędzie i cofa użytkownika do etapu wpisywania argumentów. W przypadku niedodatniej wartości liczby logarytmowanej postępuje podobnie, zmieniając jedynie treść tego komunikatu;

Obraz zawierający tekst

Opis wygenerowany automatycznieObraz zawierający tekst

Opis wygenerowany automatycznie - Oblicza wprowadzone działanie;

- Dopisuje otrzymany wynik do kolejki, a  w  wypadku przekroczenia pamięci kalkulatora usuwa najstarszą wartość;

Obraz zawierający tekst

Opis wygenerowany automatycznie

- Wyświetla na konsoli wynik działania (w nowej linijce tekstu);

-daje możliwość wyświetlenia całej pamięci kalkulatora, korzystając ponownie z funkcji funkcja validInputString udostepniającej dwie możliwe odpowiedzi użytkownika;

Obraz zawierający tekst

Opis wygenerowany automatycznie-W wypadku udzielenia pozytywnej odpowiedzi na ostatnie pytanie, program „pyta się” o chęć wyczyszczenia pamięci kalkulatora. Jeżeli i ta odpowiedź jest pozytywna, lista zawierająca elementy z historii zostaje opróżniona oraz pojawia się komunikat o tej operacji;

Obraz zawierający tekst

Opis wygenerowany automatycznie-jeżeli odpowiedź na pytanie o czyszczenie pamięci była negatywna, program indexuje elementy z pamięci i wypisuje je do konsoli (podobnie jak we wcześniej wykonywanej operacji wykorzystywania wyników z pamięci do wykonywanego działania;

Obraz zawierający tekst

Opis wygenerowany automatycznie -Moduł opisujący wyjątki, który powstrzymuje program przed systemowym zwróceniem błędu w wypadku błędnych komend z klawiatury (program sam się zamyka);

- Moduł opisujący wyjątki, który powstrzymuje program przed systemowym zwróceniem błędu w wypadku zbyt dużego formatu wprowadzonej liczby;

Obraz zawierający tekst

Opis wygenerowany automatycznie- Definiuje zawartość okienka w trybie okienkowym kalkulatora;

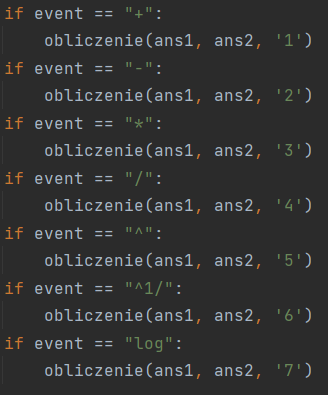
Obraz zawierający tekst

Opis wygenerowany automatycznie- tworzenie okienka;

 - Przypisuje właściwości guzikom typów obliczeń w trybie okienkowym;

Obraz zawierający tekst

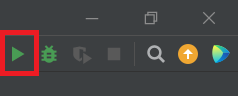
Opis wygenerowany automatycznie - Sprawdza czy użytkownik chce wyłączyć program;

- Dodaje wyniki działań w trybie okienkowym ;

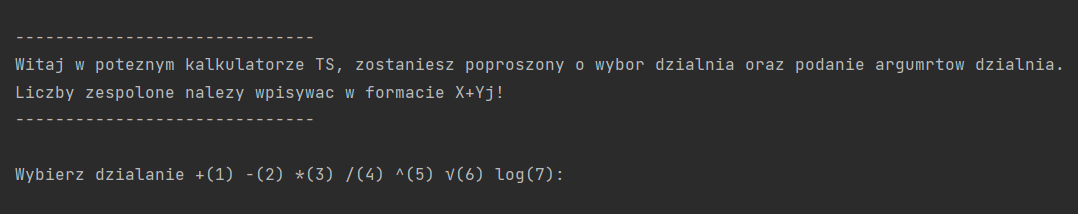
 - Zamyka okienko;

1. **Instrukcja używania**
2. **Tryb konsolowy**

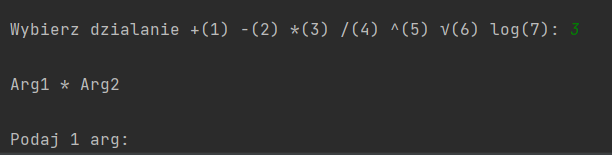
Aby skorzystać z konsolowego trybu kalkulatora należy wejść na nasz projekt na GitHubie (<https://github.com/JakubStyczen/Kalkulator->), pobrać plik calc.exe i go uruchomić. Można też skopiować kod z pliku **calc.py** oraz wkleić go do Pythona. Aby uruchomić program trzeba kliknąć w przycisk „Run” w prawym górnym rogu okna zaznaczony czerwonym prostokątem na zrzucie poniżej.



Po uruchomieniu programu w konsoli na dole okna pojawi się następujący komunikat:



Teraz trzeba wybrać typ działania jakiego chcemy użyć, należy to zrobić wpisując w konsolę cyfrę od 1 do 7, odpowiednią do typu działania, którego chcemy użyć („1” dla dodawania, „2” dla odejmowania, „3” dla mnożenia itd.). Po wybraniu typu działania naszej konsoli pojawi się taki komunikat (przedstawione dla mnożenia):



Następnie wybieramy argument 1 oraz argument 2 i obserwujemy nasz cudowny wynik 😊. Warto dodać, że kalkulator w pełni obsługuje liczby zespolone, aby skorzystać z tej funkcji liczby należy podać w formacie **X**, **Yj** lub **X+Yj**.

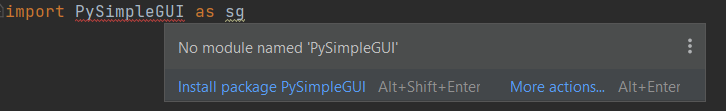
Po każdym udanym obliczeniu program pokaże komunikat: „Czy chcesz zobaczyć historie działań? Y/N”. Jeśli chcemy skorzystać z tej funkcji należy wpisać w konsoli „Y”, a jeśli nie to „N”. Kalkulator posiada również opcję korzystania z poprzednich wyników – po każdym obliczeniu oprócz pierwszego w konsoli pojawi się komunikat: „Czy chcesz użyć poprzednich wyników? Y/N”. Tak samo jak poprzednio – jeśli chcemy skorzystać z tej funkcji wpisujemy w konsoli „Y”, a jeśli nie to „N”. Po wpisaniu „Y” w konsoli pojawią się kolejne komunikaty prowadzące nas do osiągnięcia wyniku. Program zamyka się przy użyciu kombinacji CTRL+C.

UWAGI:

* W funkcji pierwiastka „**√**” możemy policzyć nie tylko pierwiastek drugiego stopnia, ale pierwiastek każdego stopnia. Używając tej funkcji argumentem pierwszym jest liczba pierwiastkowana, a drugim stopień pierwiastka. (poprawione);
* W funkcji logarytmu „log” podstawą logarytmu jest pierwszy argument, a drugi jest liczbą logarytmowaną (poprawione);
* Używając kalkulatora do obliczeń na liczbach zespolonych, gdy chcemy w argumencie użyć dokładnie liczby „i” należy wpisać „1j” (przykładowo zamiast wpisywać „2+j” trzeba wpisać „2+1j”, w innym wypadku program wyrzuci błąd o niepoprawnym formacie);

1. **Tryb okienkowy**

Żeby skorzystać z okienkowego trybu kalkulatora trzeba użyć kodu **Okienkowo.py** na naszym GitHubie. Po tym trzeba zainstalować paczkę **PySimpleGUI**. Można to zrobić między innymi nakierowując myszką na napis **PySimpleGUI** oraz klikając **Install package PySimpleGUI.**

****

Po zainstalowaniu powyższej paczki możemy uruchomić program i powinno otworzyć się takie okienko:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Teraz wystarczy tylko wpisać argumenty i kliknąć w pożądany typ działania. Przykładowo, jeśli chcemy obliczyć :

Obraz zawierający tekst

Opis wygenerowany automatycznie

Aby wyłączyć program należy wcisnąć przycisk „Quit” w lewym dolnym rogu.

UWAGI:

* Wszystkie uwagi z trybu konsolowego
* W trybie okienkowym nie ma możliwości używania opcji historii wyników ani korzystania z poprzednich wyników

1. **Sprawdzanie działania programu**

W trakcie testowania programu staraliśmy się sprawdzić poprawność działania wszystkich funkcji kalkulatora. Zwracaliśmy szczególną uwagę na możliwość pojawienia się błędów związanych z warunkami niezbędnymi dla danych działań takimi jak: niemożliwość dzielenia przez zero, liczenie pierwiastka z liczb ujemnych, obliczanie logarytmu o podstawie ujemnej bądź równej 1. Sprawdzaliśmy także działanie dla funkcji z którymi kalkulator powinien sobie radzić lub zwracać błędy przy niepoprawności formy m.in. wprowadzanie liczb ujemnych w liczbach zespolonych, wstawianie przecinków do liczby z ułamkiem dziesiętnym lub innych znaków oraz liter. Udało nam się usunąć dużą ilość błędów związanych między innymi z:

* dzieleniem przez zero (program zwracał informację o błędzie, lecz wyrzucał z konsoli) – naprawiliśmy to wprowadzając prostą pętlę cofającą nas do wyboru argumentu;
* niezgodnością z warunkami działania przy wywoływaniu liczby z pamięci kalkulatora (uwzględniał jedynie dane wpisywane w konsolę przy aktualnie wykonywanym działaniem) – dodaliśmy przypisanie wartości z pamięci do aktualnie wykonywanego elementu, tak, aby mógł on zostać uwzględniony w  kodzie warunkowym;
* wpisywaniem liczb ujemnych do logarytmu (wyskakuje błąd kompilacji) –  poprawiliśmy działanie funkcji warunkowej tak, aby działała tylko na liczbach rzeczywistych;
* kolejnością wpisywanych argumentów dla logarytmu oraz pierwiastka (nie znaliśmy kolejności wpisywania podstawy i liczby logarytmowanej oraz stopnia i liczby pod pierwiastkiem) nie było w tym co prawda błędu programu, ale była to rzecz niezbędna do korekty w celu usprawnienia działania i ułatwiania odczytu – skorygowaliśmy to poprzez dodanie komentarza o oczekiwanej kolejności danych argumentów;
* dzieleniem przez zero w funkcji pierwiastka n-tego stopnia (pierwiastek n-tego stopnia to liczba podniesiona do potęgi );
* niemożliwością policzenia dużych potęg liczb, ponieważ program wyrzucał błąd i kalkulator wyłączał się – pozbyliśmy się błędu systemowego używając „wyjątku”, przez co program zwraca nam komentarz o błędzie i pozwala na powtórne wpisanie argumentu;

1. **Wkład pracy:**

|  |  |
| --- | --- |
| Imię i Nazwisko | Jakub Styczeń |
| E-mail | jakubstyczen@student.agh.edu.pl |
| Rola w zespole | Programista (Stworzenie kodu do wersji konsolowej kalkulatora) |
| Zrealizowane zadania | * Utworzenie zdalnego repozytorium * Utworzenie modułu opisującego format wpisywanego działania * Utworzenie kolejki do zapisywania pamięci kalkulatora * Utworzenie funkcji obliczeniowych kalkulatora * Utworzenie funkcji walidacyjnych * Utworzenie pętli warunkowych dla logarytmu oraz pierwiastka * Utworzenie wyjątków programu * Pomoc w korekcie błędów wersji okienkowej |
| Liczba linii kodu | 131 |

|  |  |
| --- | --- |
| Imię i Nazwisko | Dawid Wiktorowski |
| E-mail | dawidwiktorowski18@gmail.com |
| Rola w zespole | Programista (Stworzenie kodu do wersji konsolowej okienkowej) |
| Zrealizowane zadania | * Zdefiniowanie zawartości okienka i stworzenie go * Wyświetlanie i interakcja z okienkiem * Wyjściowe wiadomości dla trybu okienkowego * Utworzenie pętli, która przyjmuje dane od użytkownika, podaje wynik, pyta o czyszczenie pamięci i dodaje wyniki do pamięci * Dodawanie wyników do kolejki * Korekta błędów |
| Liczba linii kodu | 113 |

|  |  |
| --- | --- |
| Imię i Nazwisko | Jakub Tomczyk |
| E-mail | jakubtomczyk@student.agh.edu.pl |
| Rola w zespole | Tworzenie dokumentacji projektu |
| Zrealizowane zadania | * Utworzenie zespołu w Asanie * Opisanie celu oraz planu projektu * Opisanie kamieni milowych dla projektu * Opisanie podziału pracy w zespole * Opis funkcji programu * Przeprowadzenie testów działania programu * Korekta oraz opis niektórych błędów * Opis wkładu pracy |
| Liczba linii kodu | 9 |

|  |  |
| --- | --- |
| Imię i Nazwisko | Jakub Żak |
| E-mail | kubza55@gmail.com |
| Rola w zespole | Tworzenie dokumentacji projektu |
| Zrealizowane zadania | * Opis procesu tworzenia projektu * Opis niektórych funkcji programu * Przeprowadzanie testów programu * Korekta literówek w kodzie * Opis nielicznych błędów * Instrukcja korzystania z programu * Pomoc w opisie wkładu pracy |
| Liczba linii kodu | 4 |

1. **Linki do projektu**

Kod w GitHubie:

* <https://github.com/JakubStyczen/Kalkulator->

Dokumentacja projektu:

* [Projekt - NPG.docx](https://aghedupl-my.sharepoint.com/:w:/g/personal/jakubtomczyk_student_agh_edu_pl/EWGrvt7zE7xDpHp48rQ-pmsBHfCLb5sm1qbPCHTiYN0B4A?e=Adneq1)