# Kalkulačka pre rovnice s hracími kockami – Programátorká

## Program začína importovaním nasledujúcich knižníc:

* math: na vykonávanie matematických operácií, ako je dolná celá časť a kombinácie
* operator: na vykonávanie základných aritmetických operácií, ako je sčítanie a odčítanie
* matplotlib.pyplot ako plt: na vytvorenie grafov rozdelenia pravdepodobnosti

## Pokračuje popisovaním funkcí:

1. product(args, repeat=1):
   * Toto je generátor, ktorý prijíma ľubovoľný počet argumentov a voliteľný argument opakovania.
   * Funkcia vráti všetky možné kombinácie vstupných argumentov, opakovane zadaný počet krát.
   * Táto funkcia je upravená verzia produktu z itertools, pretože tento program potreboval výstup list a nie tuple.
   * Táto funkcia mohla byť využitá aj na kombinácie výsledkov kocky, ale je neefektívna a počítanie cez vzorec je rýchlejšie (funkcia by potrebovala potom sčitovať kombinácie a tým pádom by program mal exponenciálnu zložitosť)
2. solve(Input):
   * Táto funkcia preberá spracovanú vstupnú rovnicu a rieši ju iteráciou cez vstup, určujúc potrebné operácie (sčítanie alebo odčítanie) a vrátením pravdepodobnosti každého možného výsledku (ak rovnica obsahuje slovník).
   * Vysoko modifikovaná solve funkcia pre sčitovanie a odčitovanie.
3. prob\_kocky(s,n,kocka):
   * Táto funkcia bere tri argumenty: *s*, *n* a *kocka*.
   * Počíta pravdepodobnosť získať sumu *s* pri hode *n* kockami s *kocka* stenami.
   * Uses a math equation for more accurate and faster calculations.
4. dice(pocet\_kociek, pocet\_stien, nasob=1):
   * Táto funkcia bere tri argumenty: *pocet\_kociek* (number of dice), *pocet\_stien* (number of faces on each dice) a *nasob* (multiplier)
   * Vracia slovník kde klúče sú možné sumy pri hádzaní *počet\_kociek* s *počet\_stien* a hodnoty ako pravdepodobnosť na danú sumu.
   * Volá funkciu prob\_kocky
5. magic()
   * Táto funkcia je použitá na vyriešenie rovnice s kockou.
   * Volá funkciu dice
6. counting()
   * Táto funkcia je použitá na priebežné zisťovanie počtu slovníkov v polospracovanom (semi-processed) rovnici.

## Pokračuje načítanie a práca s inputom:

* zloží čísla rozdelené list()
* vypočíta rozsah listu
* vytvorenie listu vo väčšom rozsahu

## Hlavná časť kódu:

* výpočet kociek pomocou funkcie magic() a iterovaním cez *equation*
  + pri viacerých kockách prejde do elif vetvy, ktorá používa argument *nasob* v magic()
* po vypočítaní kociek rovnica vyzerá pre sčitovanie dvoch kociek a pripočítanie konštanty nasledovne [{KEYS:HODNOTY}],’+’,[{KEYS:HODNOTY},’+’,KONŠTANTA] a spracovanie rovnice je v momentálnom štádiu rozdelené na dve časti
  + sčitovanie (odčitovanie) kociek
    - zo slovníkov spraví kópie a vyhodí všetky nulové hodnoty
    - do premennej *operators\_lst* dá všetky operátory (+,-)
    - vytvorí kombinácie všetkých slovníkov v *equation*
    - použitím *operators\_lst* buď hodnotu nechá rovnakú alebo ju zmeníme na zápornú
    - do finálneho slovníku pridá sumu z kombinácií
  + počítanie s konštantou
    - rovnicu hodíme do funkcie solve()1, ktorá nám vyhodí posčitované (po-odčitované) slovníky
* pokiaľ sú na vstupe len konštanty, program nastaví hodnotu konštanty v slovníku na 100%

(Poznámka číslo 1: pričitovanie (odčitovanie) konštanty funguje na rovnakej báze ako násobenie matice konštantou kde všetky čísla musíme vynásobiť danou konštantou. Pre nás musíme ku všetkým kľúčom pripočítať danú konštantu.)

## Výpis:

* iterovaním cez kľúče postupne:
  + appendujeme hodnoty a kľúče do premenných *hodnoty* a *kluce*
  + vypisuje formátované kľúče a hodnoty do terminálu
* jednoduchým implementovaním grafu dáme na x-ovú os hodnoty a na y-ovú kľúče (pravdepodobnosť)

## Premenné:

* origolist: input rozdelený do listu, na konci zostane rovnaký [list]
* konštanta: hodnota ktorá ukazuje či sa program dostal do funkcie magic() t.j. či je v rovnici „kocka“ [boolean]
* rozsah: hodnota určujúca maximum sloníku [integer]
* dct: slovník do ktorého ukladáme hodnoty ohľadom pravdepodobnosti [dictionary]
* equation: rovnica vytvorená z inputu [list]