**Algoritmizace - Rekurze, Brute Force, Heuristiky, Nedeterministické algoritmy**

**Algoritmizace**

* Návod, kterým jde vyřešit daný typ úlohy
* Řeší obecnou třídu problémů
* Posloupnost kroků
* Je to způsob dělení na menši podprogramy – čitelnost, bez redundantnosti
* **elementární** – skládá se z konečného počtu jednoduchých kroků
* **deterministický** – po každém kroku se dá rozhodnout, zda proces skončil
* **hromadný** – lze použít pro řešení podobných úloh

**Rekurze**

Co to vlastně je?

Objekt je (v nějakém smyslu) součástí sebe samotného

Můžeme si to představit jako funkci, která obsahuje volání sebe sama

* Je samozřejmě dobé takovou funkci kontrolovat – měla by se ukončit při splnění nějaké podmínky
* jinak by došlo k nekonečnému cyklu a funkce by nikdy neskončila

Funkce se dá použít k prohledání stromu – nejkratší cesty v grafu

* Funkce se zavolá na „kořen“
* V kořenu se funkce zavolá na každého potomka
* V potomkovi se zavolá znova na každého potomka
* Tím projdeme každý prvek

Nevede ale často k optimálnímu řešení

* Při optimalizaci se většinou snažíme takové funkce odstranit/změnit

Výhody

* Jednoduchost
* Přehlednost

Nevýhody

* Každé volání zvyšuje hloubku zanoření funkce
* Vyžaduje dostatečné místo v paměti
* Vyžaduje dostatečný čas procesoru

**Typy volání**

* Přímá rekurze
  + Podprogram volá přímo sám sebe
* Nepřímá rekurze
  + Vzájemné volání podprogramů vytváří kruh
    - *ve funkci A se volá funkce B a ve funkci B se volá opět funkce A*

**Počet volání**

* Lineární rekurze
  + Volá sám sebe pouze jednou
* Stromová rekruze
  + V rámci jednoho úkolu vyvolá vícekrát

Algoritmus rekurze

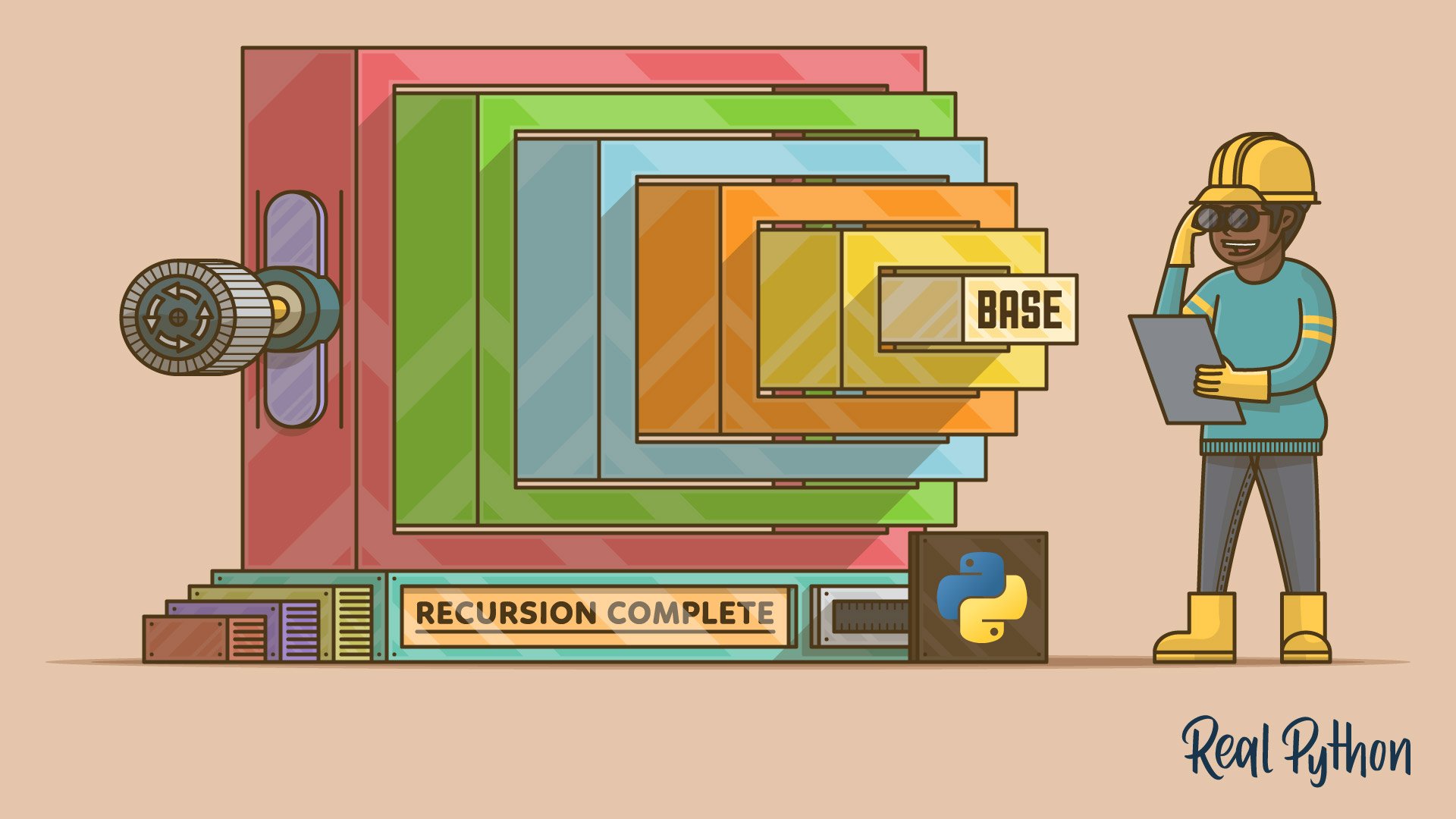
* Faktoriál



A screenshot of a computer code

Description automatically generated with low confidence A picture containing text, font, screenshot, design

Description automatically generated



**Brute force**

Co to vlastně je?

Technika hledání řešení prostřednictvím vyzkoušení všech možností v určitém rozsahu, dokud nebude nalezeno řešení

Metoda tzv. HRUBÉ SÍLY

Nejznámější případy užití algoritmu

* Hádání hesla
  + Algoritmusu je představen list všech nejpoužívanějších hesel, ty následně zkouší na svojí cíl
  + Příklady takových hesel:
    - Summer2023
    - Winter2023
    - Qwerty123
    - Atd.

Tento algoritmus je velice náročný na čas, hlavně pokud je počet možností velmi rozsáhlý

Příklad brute force v PYTHON

A picture containing text, screenshot, font

Description automatically generated

**Heuristika**

Co to vlastně je?

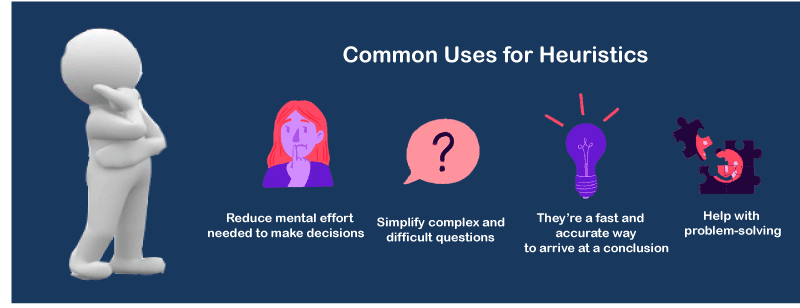
Řešení problému, na který neznáme algoritmus pomocí logiky

Často je takovéto řešení založené na zkušenostech a nebo intuici.

Jejím hlavním cílem je najít co nejpřesnější výsledek v co nejkratším čase pro konkrétní data a proto je možné, že při jiných datech by algoritmus nefungoval.

Příkladem může být řešení rozesazení lidí do lodiček podle váhy.

* Bruteforce by vygeneroval stavový prostor a řešil by, kolik váží levá a pravá strana, načež by vybral nejvhodnější možnost.
* Monte Carlo, nedeterministický algoritmus, by někoilkrát náhodně rozsadil a poté vybran nejvhodnější možnost.
* Heuristický algoritmus by nejdříve všechny podle váhy seřadil a následně je řadil postupně doleva, doprava, doleva, doprava.



Obrazek jen aby bylo :!

**Deterministické a nedeterministické algoritmy**

* **Deterministický algoritmus**
  + takový algoritmus, který za stejných podmínek vrátí stejný
  + výsledek.
  + Pevně stanovený
* **Nedeterministický** 
  + Opak
  + Ten stejnou hodnotu nevrátí.
  + Může mít totiž krok, který má více možností, zatímco deterministické je pevně stanovené mají.
  + Příkladem může být například algoritmus Monte Carlo, které pracuje jenom s přibližným výsledkem, jelikož má pokaždé jiná data na vstupu.
* **Monte Carlo**
  + Pracuje jenom s určitou částí dat
  + Pokud máme třeba 100 hodnot a máme vybrat nejnizší
    - Monte Carlo náhodně zvolí například 30 hodnot
      * Z nich vrátí přibližný výsledek
      * Proto může být nejmenší číslo 1 a v dalším případě 50

A picture containing diagram, text, line, plot

Description automatically generated