Matematický software

Zápočtový dokument

Jméno: Radek Šmejkal

Kontaktní email: radeksmejky9@gmail.com

Datum odevzdání: 26.9.2023

Odkaz na repozitář: https://github.com/radeksmejky9/MSW

Formální požadavky

Cíl předmětu:
Cílem předmětu je ovládnout vybrané moduly a jejich metody pro jazyk Python, které vám mohou být užitečné jak v dalších semestrech vašeho studia, závěrečné práci (semestrální, bakalářské) nebo technické a výzkumné praxi.

Získání zápočtu:
Pro získání zápočtu je nutné částečně ovládnout alespoň polovinu z probraných témat. To prokážete vyřešením vybraných úkolů. V tomto dokumentu naleznete celkem 10 zadání, která odpovídají probíraným tématům. Vyberte si 5 zadání, vypracujte je a odevzdejte. Pokud bude všech 5 prací korektně vypracováno, pak získáváte zápočet. Pokud si nejste jisti korektností vypracování konkrétního zadání, pak je doporučeno vypracovat více zadání a budou se započítávat také, pokud budou korektně vypracované.

Korektnost vypracovaného zadání:
Konkrétní zadání je považováno za korektně zpracované, pokud splňuje tato kritéria:

1. Použili jste numerický modul pro vypracování zadání místo obyčejného pythonu
2. Kód neobsahuje syntaktické chyby a je interpretovatelný (spustitelný)
3. Kód je čistý (vygooglete termín clean code) s tím, že je akceptovatelné mít ho rozdělen do Jupyter notebook buněk (s tímhle clean code nepočítá)

Forma odevzdání:
Výsledný produkt odevzdáte ve dvou podobách:

1. Zápočtový dokument
2. Repozitář s kódem

Zápočtový dokument (vyplněný tento dokument, který čtete) bude v PDF formátu. V řešení úloh uveďte důležité fragmenty kódu a grafy/obrázky/textový výpis pro ověření funkčnosti. Stačí tedy uvést jen ty fragmenty kódu, které přispívají k jádru řešení zadání. Kód nahrajte na veřejně přístupný repozitář (github, gitlab) a uveďte v práci na něj odkaz v titulní straně dokumentu. Strukturujte repozitář tak, aby bylo pro nás hodnotitele intuitivní se vyznat v souborech (doporučuji každou úlohu dát zvlášť do adresáře).

Podezření na plagiátorství:
Při podezření na plagiátorství (významná podoba myšlenek a kódu, která je za hranicí pravděpodobnosti shody dvou lidí) budete vyzváni k fyzickému dostavení se na zápočet do prostor univerzity, kde dojde k vysvětlení podezřelých partií, nebo vykonání zápočtového testu na místě z matematického softwaru v jazyce Python.

Kontakt:
Při nejasnostech ohledně zadání nebo formě odevzdání se obraťte na vyučujícího.

1. Knihovny a moduly pro matematické výpočty

Zadání:
V tomto kurzu jste se učili s některými vybranými knihovnami. Některé sloužily pro rychlé vektorové operace, jako numpy, některé mají naprogramovány symbolické manipulace, které lze převést na numerické reprezentace (sympy), některé mají v sobě funkce pro numerickou integraci (scipy). Některé slouží i pro rychlé základní operace s čísly (numba).

Vaším úkolem je změřit potřebný čas pro vyřešení nějakého problému (např.: provést skalární součin, vypočítat určitý integrál) pomocí standardního pythonu a pomocí specializované knihovny. Toto měření proveďte alespoň pro 5 různých úloh (ne pouze jiná čísla, ale úplně jiné téma) a minimálně porovnejte rychlost jednoho modulu se standardním pythonem. Ideálně proveďte porovnání ještě s dalším modulem a snažte se, ať je kód ve standardním pythonu napsán efektivně.

Řešení:
Testoval jsem knihovnu NumPy a zjistil jsem, že pro všechny provedené úkony bylo pomalejší než čistý Python.

Výpis z konzole

```
Čistý Python je výpočtu kořenů kvadratické rovnice rychlejší o 0.00013600 sekund
Čistý Python je ve výpočtu determinantu 3x3 matice rychlejší o 0.00005500 sekund
Čistý Python je v transpozici matice rychlejší o 0.00001900 sekund
Čistý Python je v násobení matic rychlejší o 0.00000420 sekund
Čistý Python je ve sčítání matic rychlejší o 0.00002050 sekund
```

Myslím si, že daný výsledek je způsobený malým objemem dat.
Ukázka jednoho úkonu:

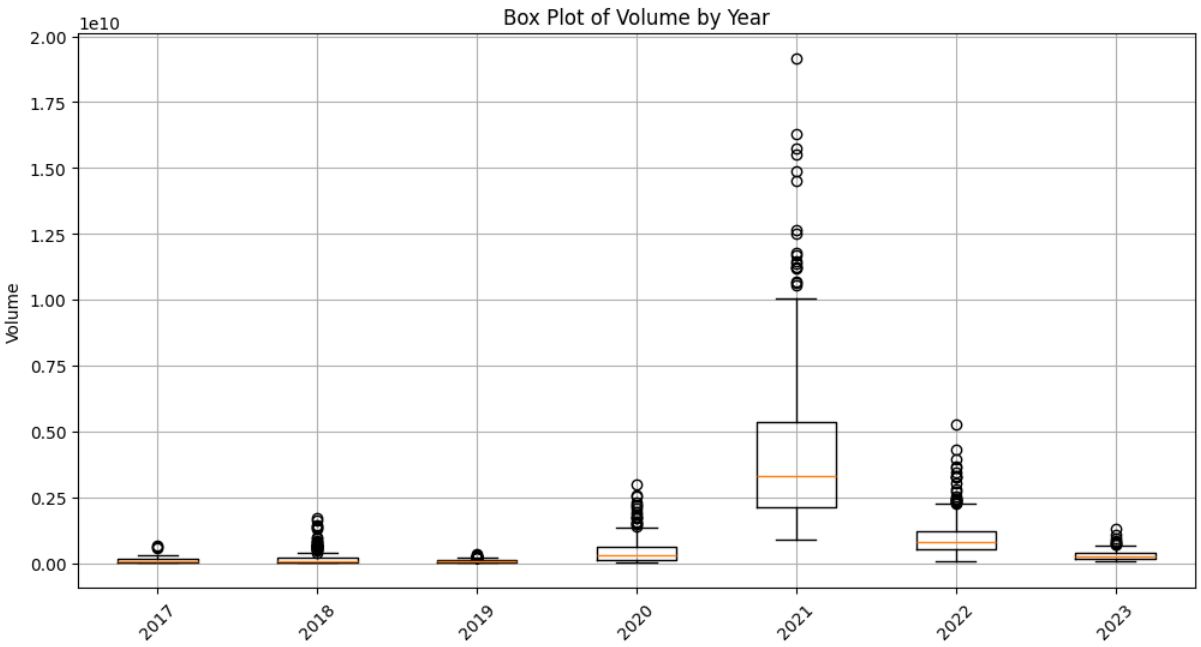
```python
# Funkce pro výpočet kořenů kvadratické rovnice pomocí čistého Pythonu
def roots_python(a, b, c):
    D = b**2 - 4 * a * c
    roots = []
    if D >= 0:
        x1 = (-b + np.sqrt(D)) / (2 * a)
        x2 = (-b - np.sqrt(D)) / (2 * a)
        if D > 0:
            roots.append(x1)
            roots.append(x2)
        else:
            roots.append(x1)
    return roots


# Funkce pro výpočet kořenů kvadratické rovnice pomocí numpy
def roots_numpy(a, b, c):
    return np.roots([a, b, c])


benchmark(
    lambda: roots_python(a, b, c),
    lambda: roots_numpy(a, b, c),
    "výpočtu kořenů kvadratické rovnice",
    1,
)
```

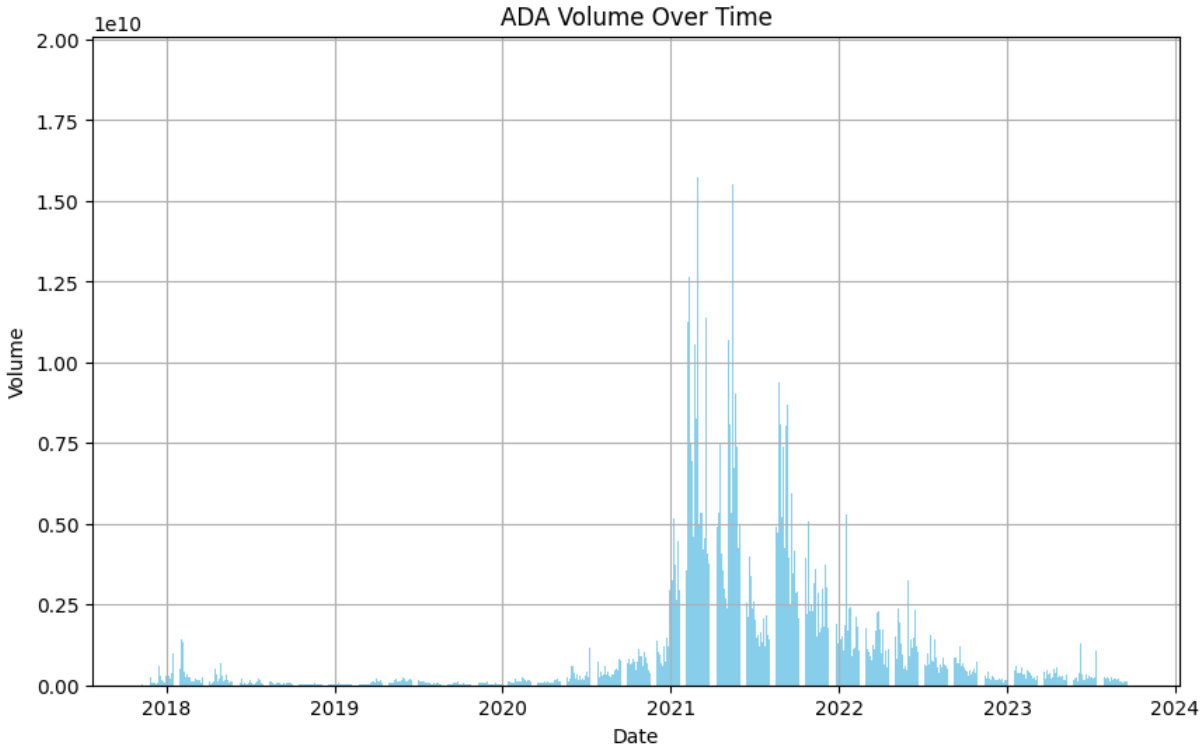
2. Vizualizace dat

Zadání:
V jednom ze cvičení jste probírali práci s moduly pro vizualizaci dat. Mezi nejznámější moduly patří matplotlib (a jeho nadstavby jako seaborn), pillow, opencv, aj. Vyberte si nějakou zajímavou datovou sadu na webovém portále Kaggle a proveďte datovou analýzu datové sady. Využijte k tomu různé typy grafů a interpretujte je (minimálně alespoň 5 zajímavých grafů). Příklad interpretace: z datové sady pro počasí vyplynulo z liniového grafu, že v létě je vyšší rozptyl mezi minimální a maximální hodnotou teploty. Z jiného grafu vyplývá, že v létě je vyšší průměrná vlhkost vzduchu. Důvodem vyššího rozptylu může být absorpce záření vzduchem, který má v létě vyšší tepelnou kapacitu.

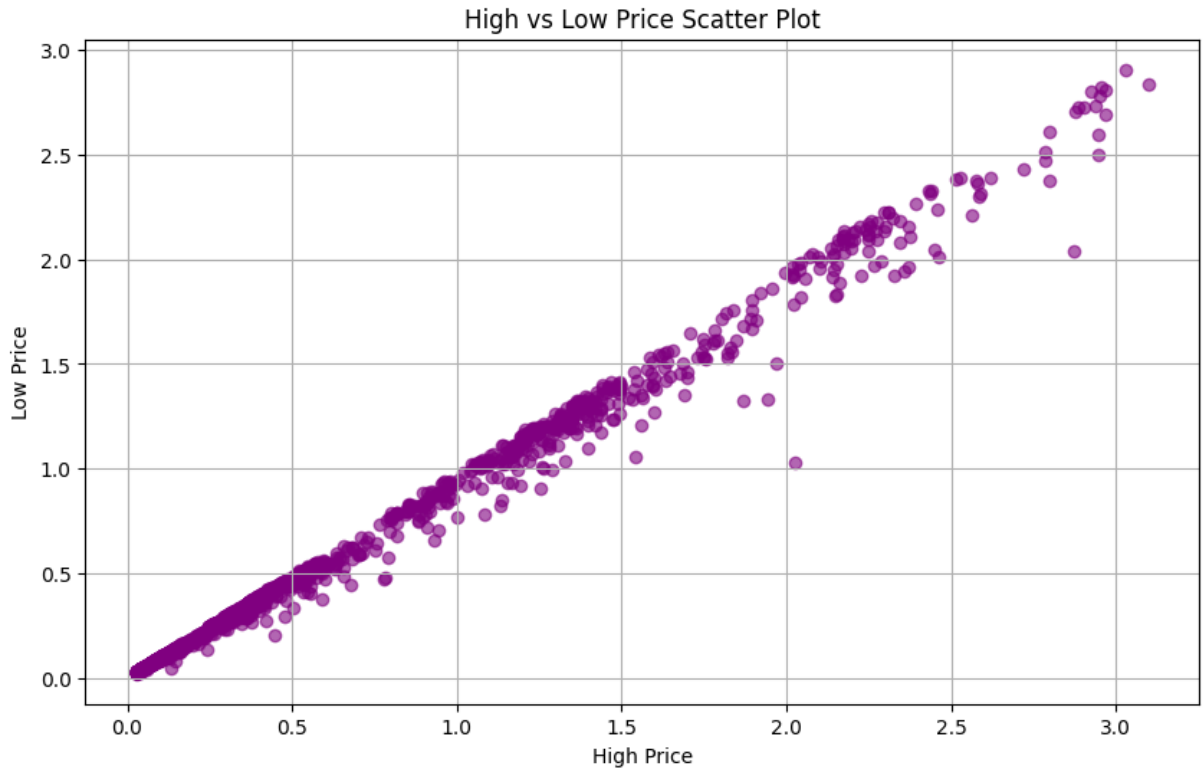
Řešení:
Pro tuto analýzu a vizualizaci jsem využil data kurzu kryptoměny Cardano a dolaru.

Z těchto grafů lze vidět, že Cardano bylo nejvíce obchodováno v roce 2021.

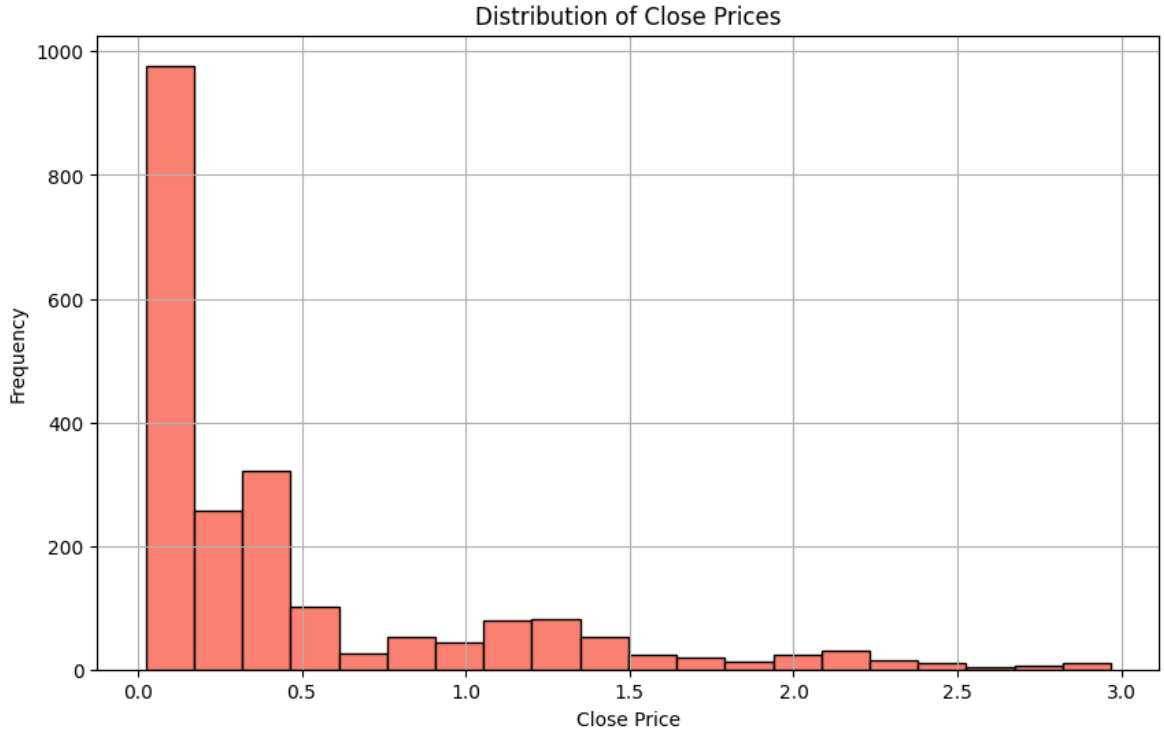




Tento graf ukazuje stav ceny v jednotlivých dnech. Respektive nejvyšší a nejnižší hodnotu v daný den. Cenové rozdíly v rámci dnů jsou relativně nízké, blížící se k diagonále.



Na tomto grafu lze vidět že cena se od zavedení kryptoměny na burzu vyšplhala relativně vysoko, ale většinu své existence ztrávila na ceně do 0.5 USD.



Tento graf ukazuje analýzu ceny od samotného začátku kryptoměny, největší cenový vzrůst zde nastal, když se kryptoměna nejvíce obchodovala.

6. Generování náhodných čísel a testování generátorů

Zadání:
Tento úkol bude poněkud kreativnější charakteru. Vaším úkolem je vytvořit vlastní generátor semínka do pseudonáhodných algoritmů. Jazyk Python umí sbírat přes ovladače hardwarových zařízení různá fyzická a fyzikální data. Můžete i sbírat data z historie prohlížeče, snímání pohybu myší, vyzvání uživatele zadat náhodné úhozy do klávesnice a jiná unikátní data uživatelů.

Řešení:
Rozhodl jsem se využít uživatele a systémový čas k vytvoření semínka. Ve své podstatě si uložím počet mikrosekund na začátku programu, nechám uživatele zadat libovolný počet znaku a poté udělám jejich ASCII součet, následně s ASCII součtem vytvořím násobek s původním počtem mikrosekund za účelem dalšího snížení šance, že by se někdy mohl počet mikrosekund shodovat. Dodal jsem ještě počet mikrosekund při provádění kódu a tím jsem znásobil hodnotu ještě jednou. Hashovací algoritmus jsem využil pro další zmenšení šance shody semínka a za účelem dosažení fixní délky.

```
 1 import datetime
 2 import hashlib
 3 import random
 4
 5
 6 tstart = str(datetime.datetime.now().microsecond)
 7 text = input("Zadej text k vytvoření semínka: ")
 8 ascii_sum = sum(ord(char) for char in text) # ASCII součet
 9 seed = str(datetime.datetime.now().microsecond * ascii_sum * tstart)
10 seed = hashlib.sha256(seed.encode()).hexdigest()
11 random.seed(seed)
12
13 print(random.randint(0, 100000))
```

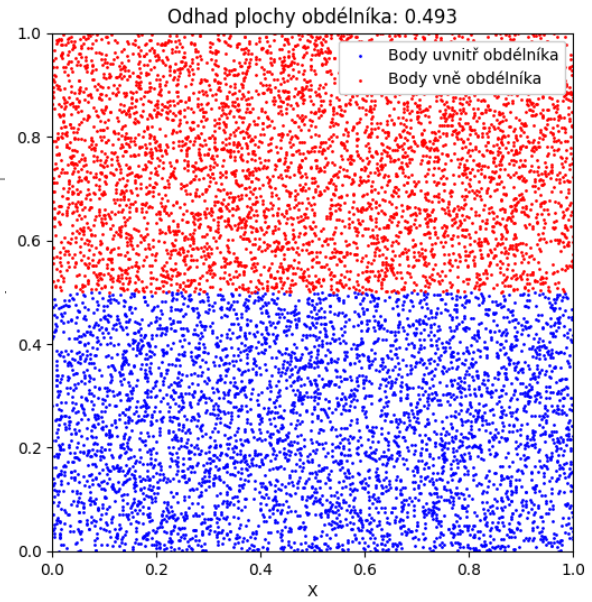
Testovací výpis:
6: Počátek: 70715 mikrosekund
7: Zadejte text k vytvoření semínka: Traktor
8: Součet: 743
9: 56146 mikrosekund
10: aac6e8820ef04b333eb16c4711bd339870ad0eede7555c73eaa004b38d89e8fd

7. Metoda Monte Carlo

Zadání:
Metoda Monte Carlo představuje rodinu metod a filozofický přístup k modelování jevů, který využívá vzorkování prostoru (například prostor čísel na herní kostce, které mohou padnout) pomocí pseudonáhodného generátoru čísel. Jelikož se jedná spíše o filozofii řešení problému, tak využití je téměř neomezené. Na hodinách jste viděli několik aplikací (optimalizace portfolia aktiv, řešení Monty Hall problému, integrace funkce, aj.). Nalezněte nějaký zajímavý problém, který nebyl na hodině řešen, a získejte o jeho řešení informace pomocí metody Monte Carlo. Můžete využít kódy ze sešitu z hodin, ale kontext úlohy se musí lišit.

Řešení:

Použil jsem metodu Monte Carlo k odhadnutí plochy obdélníka. Definoval jsem si velký čtverec a v něm menší obdélník, následně jsem pomocí cyklu vytvořil náhodně položené „body“ a kontroloval jsem, jestli se nachází uvnitř obdélníka, či nikoliv. Poté jsem vypočítal odhad plochy podle celkového počtu bodů uvnitř. Došel jsem k závěru, že při jednom tisíci bodů je odhad vzdálený od výsledku relativně hodně. Samozřejmě štěstí rozhoduje o výsledku, takže při stovkách tisíc bodů už je zaokrouhlený odhad v podstatě jistotou.




```python
width_large = 1
height_large = 1
width_small = 1
height_small = 0.5

num_points = 10000

inside_rectangle = 0
points_inside_x = []
points_inside_y = []
points_outside_x = []
points_outside_y = []

for _ in range(num_points):
    x = random.uniform(0, width_large)
    y = random.uniform(0, height_large)

    if 0 <= x <= width_small and 0 <= y <= height_small:
        points_inside_x.append(x)
        points_inside_y.append(y)
    else:
        points_outside_x.append(x)
        points_outside_y.append(y)

rectangle_area_estimate = (len(points_inside_x) / num_points) * (width_large * height_large)
```

9. Integrace funkce jedné proměnné

Zadání:
V oblasti přírodních a sociálních věd je velice důležitým pojmem integrál, který představuje funkci součtů malých změn (počet nakažených covidem za čas, hustota monomerů daného typu při posouvání se v řetízku polymeru, aj.). Integraci lze provádět pro velmi jednoduché funkce prostou Riemannovým součtem, avšak pro složitější funkce je nutné využít pokročilé metody. Vaším úkolem je vybrat si 3 různorodé funkce (polynom, harmonická funkce, logaritmus/exponenciála) a vypočíst určitý integrál na dané funkci od nějakého počátku do nějakého konečného bodu. Porovnejte, jak si každá z metod poradila s vámi vybranou funkcí na základě přesnosti vůči analytickému řešení.

Řešení:

Výsledky pro Logaritmickou funkci

Lichoběžníková metoda	1.8381486503400901
Rombergova metoda	1.838339455060071
Simpsonova metoda	1.8383398721115707
Analytické řešení	$\int_0^1 \log\left(\frac{8x}{3}+5\right)dx = -1-\frac{15\log(5)}{8}+\log\left(\frac{529}{9}\left(\frac{23}{3}\right)^{7/8}\right) \approx 1.8383$

Výsledky pro Goniometrickou funkci

Lichoběžníková metoda	0.1232759146813425
Rombergova metoda	0.1315821280213854
Simpsonova metoda	0.13299663988169022
Analytické řešení	$\int_0^1 \cos\left(4x^2-\frac{7}{3}\right)dx = \frac{1}{2}\sqrt{\frac{\pi}{2}}\left(C\left(2\sqrt{\frac{2}{\pi}}\right)\cos\left(\frac{7}{3}\right)+S\left(2\sqrt{\frac{2}{\pi}}\right)\sin\left(\frac{7}{3}\right)\right) \approx 0.131582$

Výsledky pro Polynomickou funkci

Lichoběžníková metoda	2.907788446883097
Rombergova metoda	2.8666666666666667
Simpsonova metoda	2.8677538993039677
Analytické řešení	$\int_0^1 \left(6x^4+4x^3+2x^2\right)dx = \frac{43}{15} \approx 2.8667$

```python
def print_results(text, f, x, a, b):
    y = f(x)
    trapezoid_result = integrate.trapezoid(y, x, dx=dx)
    romberg_result = integrate.romberg(f, a, b)
    simpson_result = integrate.simpson(y, x)

    print(f"\nVýsledky pro {text}:")
    print(f"použitím lichoběžníkové metody: {trapezoid_result}")
    print(f"použitím   Rombergovy   metody: {romberg_result}")
    print(f"použitím   Simpsonovy   metody: {simpson_result}")
```