

# Interaktivní mapa

Zde se věnuji problematice zpracování geografických otevřených dat státní správy a jejich vizualizace v rámci interaktivní mapy.

Při zpracování tohoto tématu primárně využiji knihovnu *Folium*, což je Pythonovský wrapper JavaScriptové knihovny *Leaflet*. Mapový podklad, se kterým knihovna *Leaflet* pracuje, je založen na platformě *OpenStreetMap*.

Jako vstupní data použiji geografické soubory ve formátu GeoJSON. Jejich načtení a validaci zajistí v Pythonu „vestavěný“ modul *json*. Z dat uložených v každém geografickém souboru se vytvoří tzv. oblasti, tzn. překryvné vrstvy nad základním mapovým podkladem, které mají uživatelem zvolené barvy a názvy.

Výstupem bude HTML soubor s vygenerovanou mapou obsahující vizualizaci dat načtených z geografických souborů a s možností jejího interaktivního použití (např. volba zobrazení jednotlivých oblastí, dynamické popisky, zvětšování/zmenšování mapy atd.).

Pro výběr vstupních a výstupního souboru, jakožto i zobrazení informačních a dotazovacích oken využiji grafické prostředí (GUI) *Tk*, a to prostřednictvím interface Pythonu s názvem *TKinter*. Tento způsob se velmi osvědčil již při řešení úloh v rámci předchozích předmětů.

## Výsledky a fakta:

Při implementaci jednotlivých funkcí knihovny *Folium* jsem vycházel především z internetových referenčních zdrojů dané knihovny, a to konkrétně např.:

<https://python-visualization.github.io/folium/index.html> (úvodní stránka popisu knihovny)

<https://python-visualization.github.io/folium/quickstart.html> (rychlá reference knihovny)

<https://python-visualization.github.io/folium/modules.html> (detailní reference knihovny)

Neocenitelné byly též praktické příklady využití knihovny *Folium*, a to např:

<https://www.python-graph-gallery.com/288-map-background-with-folium>,

<https://pretagteam.com/question/how-do-you-add-geojsontooltip-to-foliumchoropleth-class-in-folium> či

<https://stackoverflow.com/questions/55088688/how-do-you-add-geojsontooltip-to-folium-choropleth-class-in-folium>.

Užitečnou se ukázala i detailní reference JavaScript knihovny Leaflet:

<https://leafletjs.com/reference.html>.

Rozsahem nevelkou, avšak velmi důležitou částí kódu je validace vstupních dat, neboť formát GeoJSON, resp. jeho výchozí platforma JSON (JavaScript Object Notation) má pochopitelně svá syntaktická pravidla (viz <https://www.json.org>). Zde jsem se inspiroval validační funkcí z následujícího zdroje: <https://stackoverflow.com/questions/11294535/verify-if-a-string-is-json-in-python>, kterou jsem modifikoval tak, aby kromě vlastní validace formátu JSON prováděla i otevírání vstupních souborů a v případě výskytu jakýchkoli problémů vyvolala výjimku. S problematikou výjimek v Pythonu jsem se seznámil zde: <https://www.itnetwork.cz/python/soubory/vyjimky-v-pythonu>.

Velká část kódu je věnována zabezpečení vstupu a výstupu z/do souborů, včetně dialogových oken pro jejich výběr. Jak již bylo řečeno, pro jejich implementaci je použit interface Pythonu s názvem Tkinter. Takto zpřístupněné grafické prostředí Tk pak je s výhodou využito i pro zobrazování informačních a dotazovacích oken. Při návrhu začlenění tohoto prostředí jsem vycházel z řady internetových zdrojů, např.:

<https://stackoverflow.com/questions/3579568/choosing-a-file-in-python-with-simple-dialog>,

<https://docs.python.org/3.9/library/tk.html>

<https://docs.python.org/3.9/library/dialog.html#module-tkinter.filedialog>,

<http://programujte.com/forum/vlakno/4391-neviditelny-py/#p34029>,

<https://docs.python.org/3.9/library/tkinter.messagebox.html>

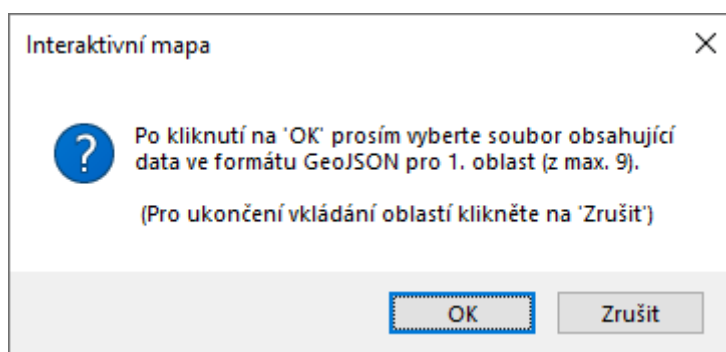
či <https://www.py.cz/TkinterSouboryAdresare>.

Konkrétní postup je velmi detailně okomentován přímo v daném zdrojovém kódu.

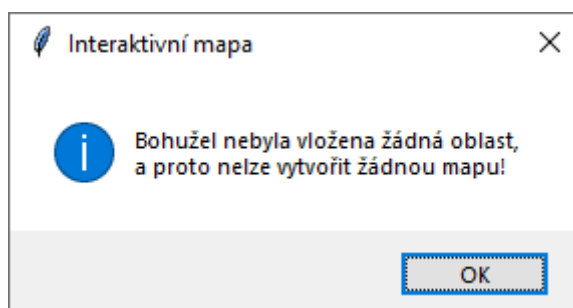
Vlastní prostředí Tk používá pro grafický výstup řadu prvků (tzv. widgetů), a to včetně tzv. hlavního okna („root“). Pro zobrazování dialogových oken (de facto náležejících přímo

operačnímu systému) však žádné widgety nejsou potřeba – naopak, zobrazování (v podstatě prázdného) hlavního okna „root“ by působilo rušivě, proto je toto skryto.

Po spuštění kódu je nejdříve zobrazeno informační okno, aby uživatel věděl, jak má dále postupovat. Dané okno se pak objevuje před každým dalším vložením nové oblasti. Maximální počet oblastí je omezen na 9, aby byla zajištěna přehlednost výsledné mapy:

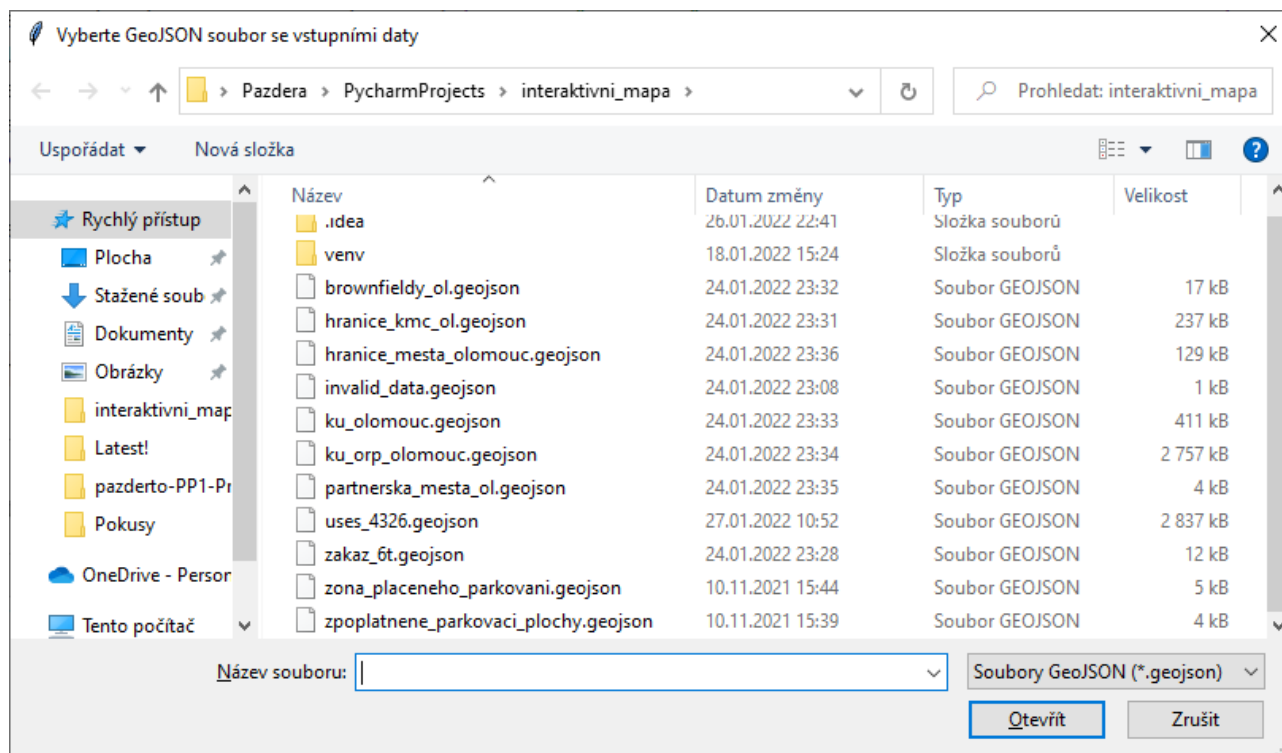


Pro zobrazení tohoto okna je využita metoda `askokcancel` importovaná z modulu `tkinter.messagebox`. Klikne-li uživatel na tlačítko Zrušit, tak se vkládání oblastí ukončí. V případě, že se jednalo o 1. oblast, a tudíž žádné oblasti de facto nebyly vloženy, tak nemá smysl generovat žádnou mapu a je zobrazeno následující informační okno, po jehož potvrzení tlačítkem OK se program ukončí:



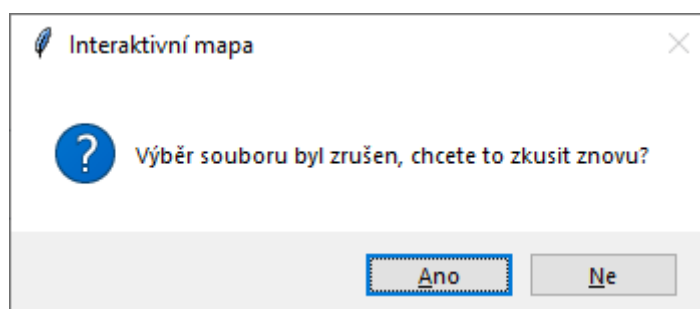
Pro zobrazení tohoto (i dalších následných informačních oken) je využita metoda `showinfo` importovaná z modulu `tkinter.messagebox`. Dané okno se pochopitelně zobrazí i v případě, že vkládání úvodní oblasti bude předčasně ukončeno i v následujících krocích (např. při volbě názvu oblasti, její barvy atd.).

Pokud naopak u úvodního informačního okna uživatel klikne na tlačítko OK, otevře se vlastní dialogové okno pro výběr vstupního GeoJSON souboru:



Zobrazení tohoto dialogového okna zajišťuje metoda `askopenfilename` importovaná z modulu `tkinter.filedialog`. Tato metoda vrátí název zvoleného souboru (samozřejmě i s kompletní cestou), který je následně podstoupen k otevření a validaci dat.

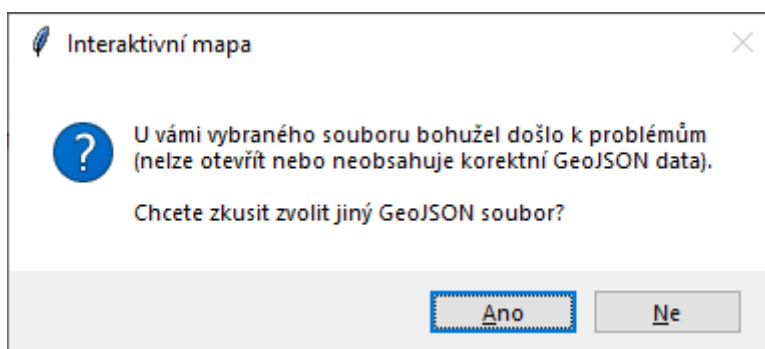
Pokud byl výběr souboru zrušen, objeví se následující dotazovací okno:



Pro zobrazení tohoto (i dalších následných dotazovacích oken) je využita metoda `askyesno` importovaná z modulu `tkinter.messagebox`. Po kliknutí na `Ano` se umožní opakovaný výběr vstupního GeoJSON souboru prostřednictvím výše zmiňovaného dialogového okna, kliknutím na `Ne` se vkládání oblastí do mapy ukončí.

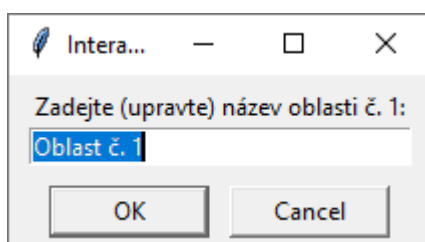
Zvolený název souboru (včetně cesty) je následně použit jako argument funkce `json_load_safe`, která provede otevření daného souboru v kódování UTF-8. (toto kódování je pro data v JSON formátu stanoveno jako výchozí, viz [https://cs.wikipedia.org/wiki/JavaScript\\_Object\\_Notation](https://cs.wikipedia.org/wiki/JavaScript_Object_Notation)). Otevřený soubor je předán jako argument metody `json.load` importované z vestavěného Python modulu `json`. (viz <https://docs.python.org/3/library/json.html>). V případě jakéhokoli problému s otevřením souboru a/nebo jeho nevalidním obsahem (nesprávným či chybějícím JSON formátem dat) je vyvolána výjimka a funkce `json_load_safe` vrací hodnotu `None`. V opačném případě je vrácen JSON objekt (založený na datovém typu Dictionary).

Pokud se uživatel pokusí zvolit soubor s nevalidními JSON daty, resp. vybraný soubor nelze otevřít, zobrazí se následující dotazovací okno:

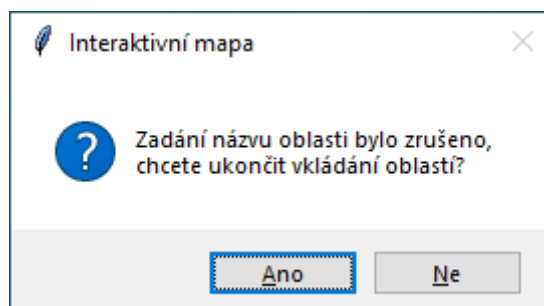


Po kliknutí na Ano se umožní opakovaný výběr vstupního GeoJSON souboru prostřednictvím výše zmiňovaného dialogového okna, kliknutím na Ne se vkládání oblastí ukončí.

V případě, že jsou data načtená ze souboru korektní, program pokračuje dotazem na název právě vkládané oblasti:

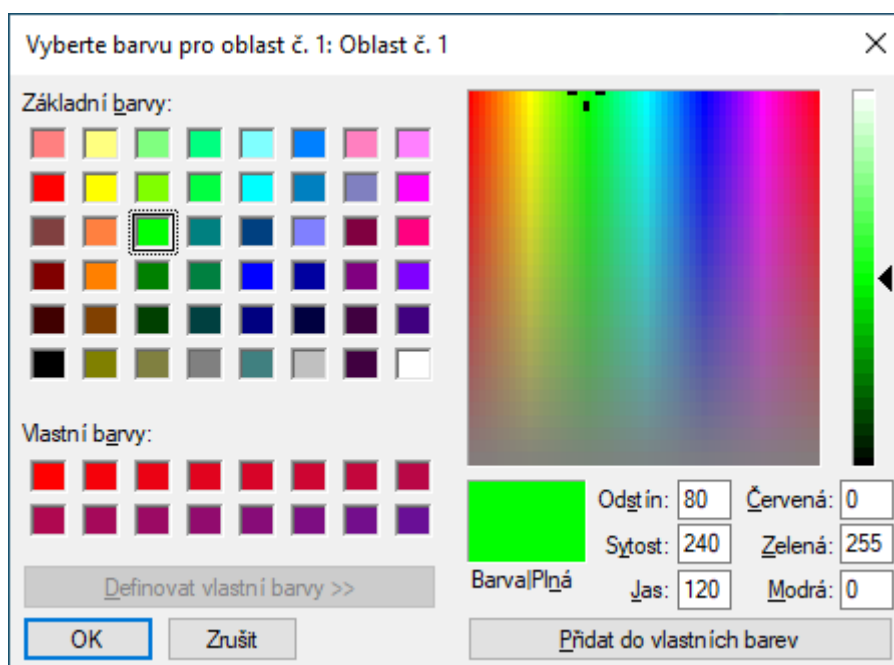


Pro zobrazení tohoto vstupního okna je využita metoda `askstring` importovaná z modulu `tkinter.simpledialog`. Jako výchozí název je přednastaven text „Oblast č. x“, kde x je číslo aktuální oblasti. Program nedovolí vložení prázdného názvu. Po kliknutí na OK je zadaný název akceptován, po kliknutí na Cancel (bohužel okna `simpledialog` nejsou lokalizovaná) se zobrazí následující dotazovací okno:



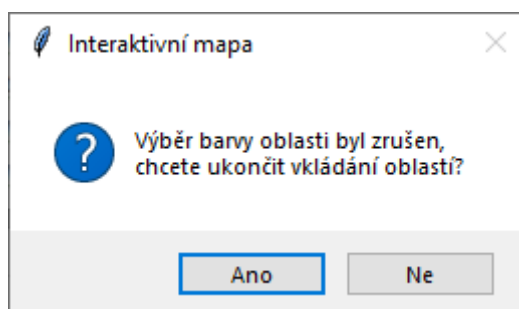
Kliknutím na Ne se umožní opakované zadání názvu právě vkládané oblasti, kliknutím na Ano se vkládání oblastí do mapy ukončí.

Dalším krokem je volba barvy oblasti. Pro tento účel je použito dialogového okna `askcolor` z modulu `tkinter.colorchooser`:



Jako výchozí barva je pro první oblast přednastavena „čistá“ zelená ( $R = 0$ ,  $G = 255$ ,  $B = 0$ ), pro další oblasti v pořadí je přednastavována barva, kterou uživatel zvolil pro předchozí oblast.

Kliknutím na OK uživatel vybranou barvu potvrdí, pokud se však rozhodne výběr barvy zrušit, se zobrazí následující dotazovací okno:



Kliknutím na Ne se umožní opakovaný výběr barvy právě vkládané oblasti, kliknutím na Ano se vkládání oblastí do mapy ukončí.

Po potvrzení barvy oblasti dojde v případě vkládání první oblasti k vytvoření objektu s mapovým podkladem pomocí metody `folium.Map` importované v rámci modulu knihovny Folium. Při vkládání následných oblastí je pak používán tento prvotně vytvořený mapový podklad.

Vlastní vizualizace jednotlivých oblastí (vrstev) nad mapovým podkladem je zajišťována formou vytváření speciálních objektů a jejich následným vkládáním do objektu mapového podkladu. Postupně tak vzniká určitá variace tzv. Choropletové mapy (neboli kartogramu). Skutečný kartogram je jednoduchá mapa, v níž je graficky vyjádřena (barvou či rastrem) intenzita jevu ve sledovaném území (viz <https://cs.wikipedia.org/wiki/Kartogram>). V našem případě není třeba vizualizovat intenzity žádných jevů – prostředky pro tvorbu Choropletových map v rámci knihovny Folium jsou využity pouze pro barevné odlišení jednotlivých oblastí.

Pro vytvoření příslušného objektu (tj. překryvné – overlay vrstvy nad mapovým podkladem) vyjadřujícího konkrétní oblast je v rámci modulu knihovny Folium použita metoda `folium.Choropleth`. Ta má poměrně velkou řadu parametrů (viz <https://python-visualization.github.io/folium/modules.html#folium.features.Choropleth>), z nichž bezesporu nejdůležitějším je parametr určený pro vstup požadovaných geografických dat, v našem

případě tedy JSON objektu, obsahujícího příslušná GeoJSON data popisující konkrétní oblast. V rámci tohoto programu je pak využíváno i několik dalších parametrů, a to pro nastavení názvu oblasti, barvy její výplně včetně ohraničení (jejich hodnoty jsou získávány od uživatele výše uváděnými postupy) a dále pak šířky ohraničení, úrovně zvýraznění a stupně průhlednosti (hodnoty pro tři posledně jmenované parametry byly pro jednoduchost stanoveny empiricky). K vytvořenému objektu s oblastí je dále ještě přidána popiska („*Tooltip*“) umožňující dynamické zobrazení názvu oblasti při najetí myši (metody `add_child` a `folium.features.Tooltip`) a tento celek je pak přidán do mapy metodou `add_to`.

Poslední operací s objektem obsahujícím aktuální oblast je zjištění rozměrů (ohraničení) této oblasti (tzv. *bounds*). Jedná se o geografické souřadnice okrajů oblasti ve formátu `[[lat_min, lon_min], [lat_max, lon_max]]` (tj. dvouprvkový seznam obsahující další dva vnořené dvouprvkové seznamy) kde:

`lat_min` = minimální zeměpisná šířka levého dolního okraje oblasti

`lon_min` = minimální zeměpisná délka levého dolního okraje oblasti

`lat_max` = maximální zeměpisná šířka pravého horního okraje oblasti

`lon_max` = maximální zeměpisná délka pravého horního okraje oblasti

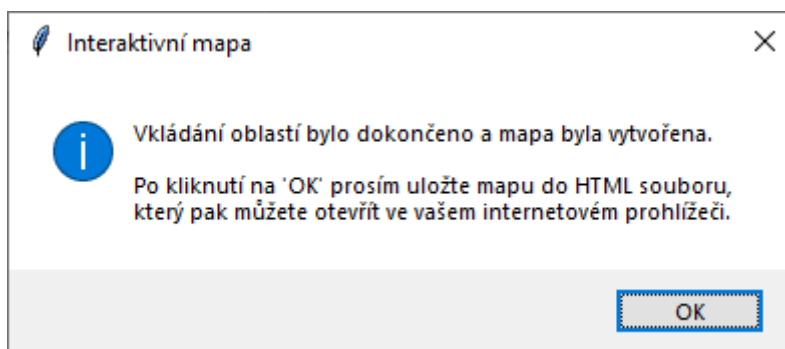
Pro načtení souřadnic ohraničení oblasti je použita metoda `get_bounds`. V průběhu vkládání jednotlivých oblastí jsou pak vypočítávány největší možné souřadnice ohraničení všech oblastí (tj. de facto sjednocení těchto oblastí), a to tak, že jsou hledány minima a maxima jednotlivých prvků příslušných seznamů (tj. seznamu obsahující zeměpisné souřadnice aktuálně vkládané oblasti a seznamu se souřadnicemi sjednocení všech předchozích oblastí).

Jakmile jsou vloženy všechny oblasti, nastává finální fáze tvorby interaktivní mapy, a to nastavení pozice mapy, úrovně zvětšení a přidání ovládacích prvků. Pro první dvě akce je použita dříve vypočtená hodnota pro ohraničení sjednocení všech oblastí. Souřadnice pozice mapy (tj. de facto geografický střed zobrazované mapy) jsou vypočítány jako aritmetické průměry maximálních a minimálních hodnot zeměpisných šířek, resp. zeměpisných délek a výsledek je přiřazen vlastnosti `location` objektu s mapou. Seznam definující ohraničení sjednocení všech oblastí je pak ještě použit i pro nastavení aktuálního zvětšení (zoomu) interaktivní mapy, a to jako argument metody `fit_bounds` objektu s mapou. Tím je dosaženo, že se výsledná mapa zobrazí s maximálním možným zvětšením, tj. tak, aby byly

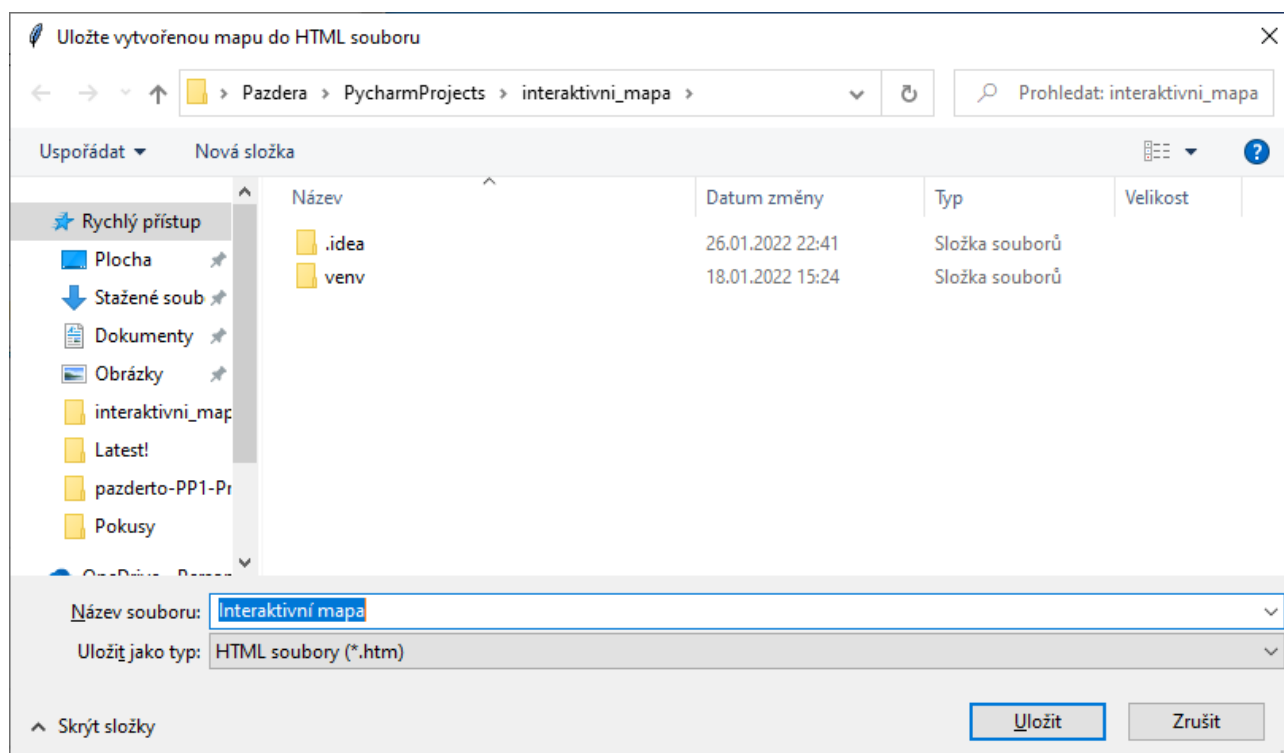


právě viditelné všechny vložené oblasti. Posledním krokem je vytvoření objektu s ovládacími prvky mapy (např. výběr zobrazení jednotlivých oblastí atd.), a to pomocí metody `folium.LayerControl`. Takto vytvořený objekt je pak vložen do mapy metodou `add_to`.

Po kompletním vytvoření interaktivní mapy je zobrazeno následující okno, informující o této skutečnosti uživatele a upozorňující na potřebu uložení této mapy do HTML souboru:



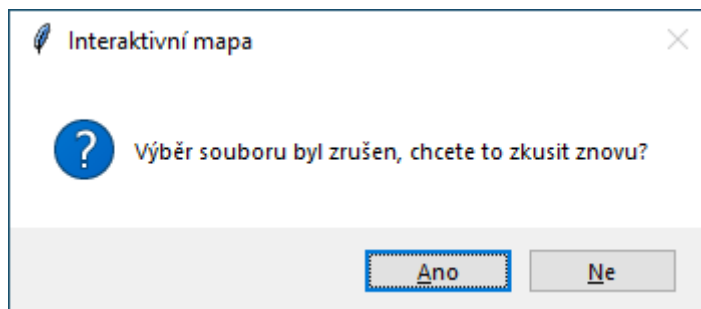
Po kliknutí na tlačítko OK je uživateli umožněn výběr výstupního HTML souboru pro uložení mapy (jako výchozí je nastaven název souboru „Interaktivní mapa“):



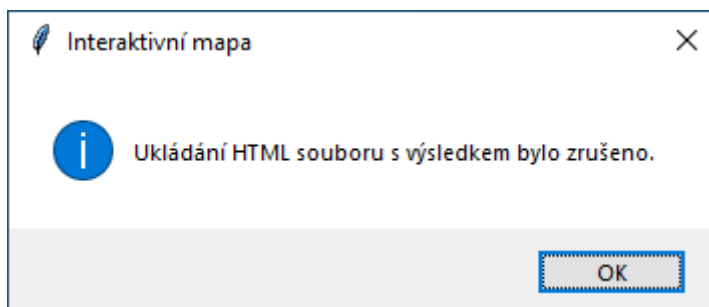
Zobrazení dialogového okna pro ukládání zajišťuje metoda `asksaveasfilename` importovaná z modulu `tkinter.filedialog`. Po kliknutí na tlačítko Uložit vrací daná

metoda název zvoleného souboru (samozřejmě i s kompletní cestou), který je následně použit jako argument metody `save` objektu `s` mapou. Daná metoda pak zajistí nejen uložení kompletní mapy do HTML souboru, ale následně i uzavření daného souboru.

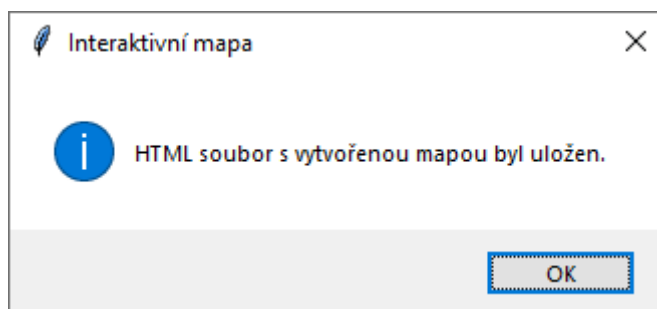
Klikne-li uživatel na tlačítko Zrušit, zobrazí se následující dotazovací okno:



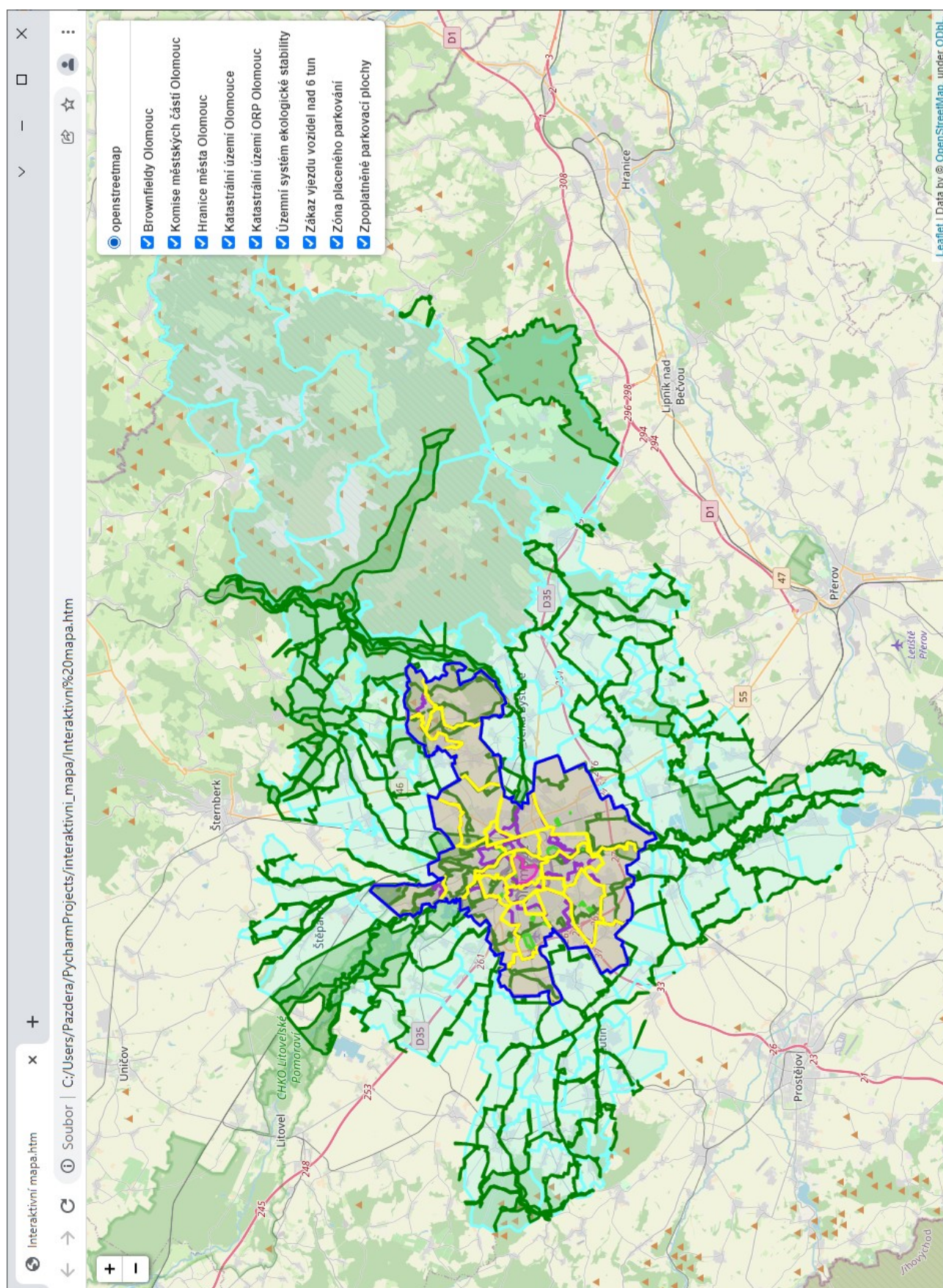
Po kliknutí na Ano se znovu umožní výběr výstupního HTML souboru pro uložení mapy, kliknutím na Ne se zobrazí se následující informační okno (po kliknutí na OK se pak definitivně program ukončí):



Naopak proběhlo-li vše v pořádku, je vytvořen příslušný HTML soubor, což je potvrzeno následujícím informačním oknem (po kliknutí na OK se pak program finálně ukončí):



Takto vytvořený soubor je pak možné zobrazit v libovolném moderním internetovém prohlížeči:

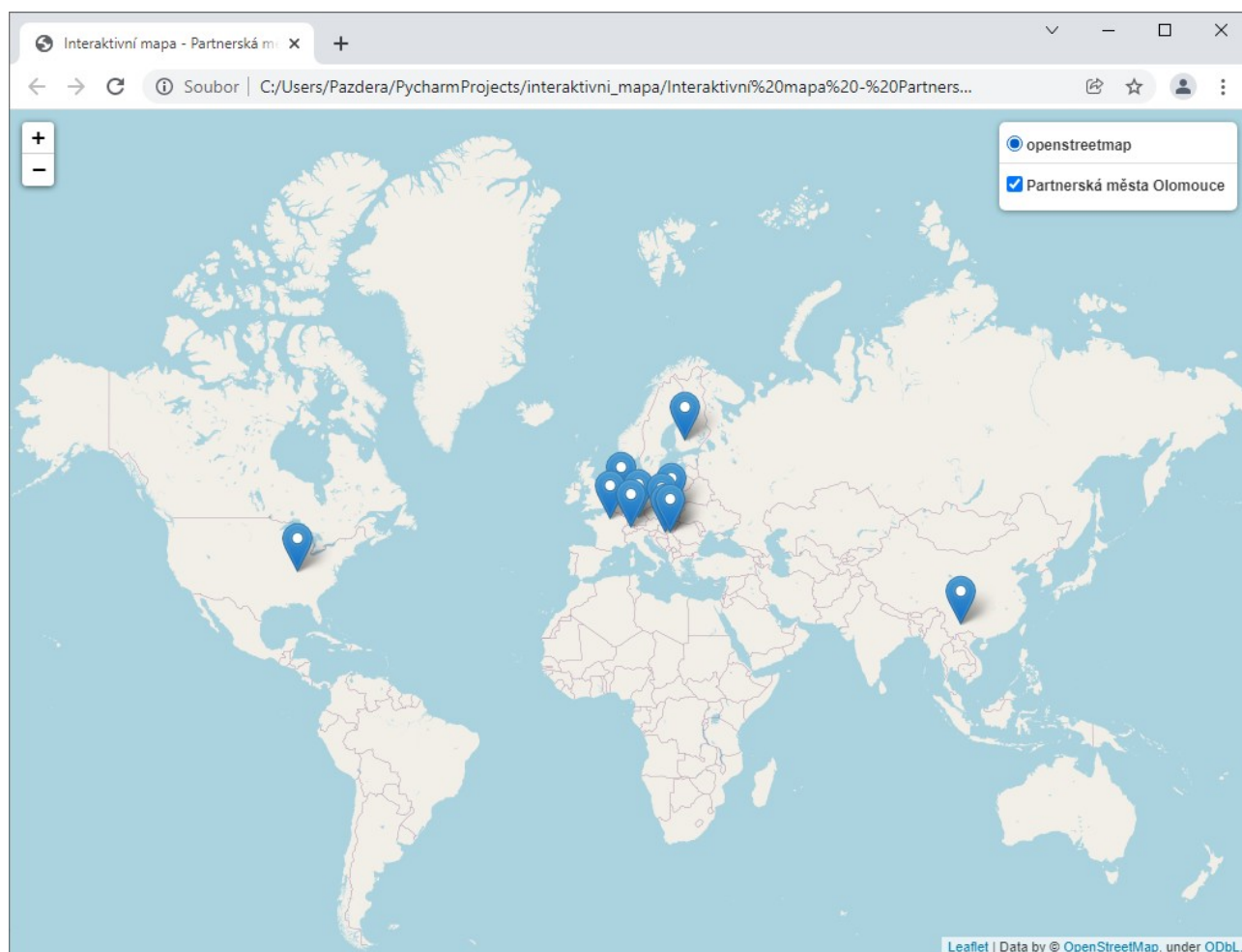




## Závěr:

Vzniklý program jsem se samozřejmě snažil co možná nejdůkladněji otestovat. Kromě zpracovávání GeoJSON souborů stažených z „Portálu otevřených dat“ Ministerstva vnitra České republiky, a to konkrétně datových sad Statutárního města Olomouc (viz <https://data.gov.cz/datov%C3%A9-sady?poskytovatel=https%3A%2F%2Frp-opendata.egon.gov.cz%2Fodrpp%2Fzdroj%2Forg%C3%A1n-ve%C5%99ejn%C3%A9-moci%2F00299308>) jsem si vytvořil i pomocný testovací soubor `invalid_data.geojson`, záměrně obsahující nekorektní vstupní data. I tento soubor pak byl programem správně zpracován, tj. odmítnout z důvodu nesprávných vstupních dat.

V rámci výše uváděných datových sad Statutárního města Olomouc je k dispozici i GeoJSON soubor `partnerska_mesta_ol.geojson`, který je poněkud odlišný od všech ostatních – jedná se totiž o tzv. bodovou vrstvu, tj. nedefinuje určitou plochu na mapě (tzv. polygon) jako ostatní soubory, ale označuje konkrétní místa na mapě body (tzv. markery). Navíc tato vrstva pokrývá celý svět. I tento datový soubor však program bez problémů zpracoval:



Výše zmiňované datové soubory jsou součástí „balíku“ souborů, který dále obsahuje výsledné HTML soubory (`Interaktivní mapa*.htm`), vlastní zdrojový kód programu (`interaktivni_mapa.py`), a dále též konfigurační soubory vývojového prostředí PyCharm. Tento „balík“ se nachází ve složce projektu s názvem `interaktivni_mapa`.