

Lekce 12: Geoprocessingové nástroje I

Python pro GIS - Vektorové analýzy

Vojtěch Barták, FŽP ČZU Praha

2025-11-08

Table of contents

1	Cíle lekce	4
2	Proč Python pro vektorové analýzy?	4
2.1	Co už umíte	4
2.2	Výhody automatizace pomocí Pythonu	4
2.2.1	Manuálně v GUI	4
2.2.2	Automaticky v Pythonu	5
2.3	Praktické situace	5
2.4	Propojení s Lekcí 11	5
3	Od GUI k Pythonu	6
3.1	Copy Python Command	6
3.2	Struktura příkazu	6
3.3	Starý způsob volání nástrojů	6
4	Základní vektorové nástroje v Pythonu	7
4.1	Rychlý přehled syntaxe	7
5	Batch processing - Propojení s Lekcí 11	8
5.1	Problém: 50 vrstev, 50× klikat?	8
5.2	Batch processing s podmínkou	9
5.3	Batch processing s try-except	10
6	Adaptivní batch processing	11
6.1	Buffer podle typu geometrie	11
7	Clip - Ořezání	12
7.1	Co dělá Clip?	12

7.2	Základní použití	12
7.3	Parametry Clip	12
7.4	Praktický příklad	13
8	Intersect - Průnik	13
8.1	Co dělá Intersect?	13
8.2	Základní použití	14
8.3	Parametry Intersect	14
8.3.1	Klíčové parametry	14
8.4	Příklad: Které obce jsou v dosahu řek?	15
9	Union - Sjednocení	15
9.1	Co dělá Union?	15
9.2	Základní použití	16
9.3	Parametry Union	16
9.4	Praktický příklad	16
10	Select - Výběr podle atributů	17
10.1	Co dělá Select?	17
10.2	Základní použití	17
10.3	Parametry Select	17
10.3.1	Syntaxe where_clause	18
10.4	Praktický příklad	18
11	Dissolve - Rozpuštění hranic	19
11.1	Co dělá Dissolve?	19
11.2	Základní použití	19
11.3	Parametry Dissolve	19
11.3.1	Klíčové parametry	20
11.4	Praktický příklad	20
12	Práce s výstupy nástrojů	21
12.1	GetCount - Počet prvků	21
12.2	GetOutput - Získání výstupu	21
12.3	Řetězení nástrojů	21
13	Řetězení nástrojů	22
13.1	Použití výstupu jako vstupu	22
13.2	GetCount a GetOutput	22
14	Komplexní příklad: Batch analýza s reporting	23
14.1	Zadání	23
14.2	Řešení	23

15 Funkce pro opakované použití	25
15.1 Problém: Stejný kód vícekrát?	25
16 Shrnutí	26
16.1 Co jsme se naučili	26
16.2 Proč je Python lepší než GUI?	26
16.3 Klíčové myšlenky	27
16.4 Co bude přistě?	27
17 Praktická cvičení	27
17.1 Cvičení 1: Batch buffer s reportingem	27
17.2 Cvičení 2: Adaptivní batch processing	28
17.3 Cvičení 3: Analýza s try-except	28
18 Domácí úkol	28
18.1 Varianta A (základní)	28
18.2 Varianta B (střední)	29
18.3 Varianta C (pokročilá)	29
18.4 Varianta D (výzva)	30
19 Cheatsheet	31
20 Poznámky pro vyučujícího	33
20.1 Filozofie lekce	33
20.2 Běžné chyby studentů	33
20.3 Časový plán (90 min)	34
20.4 Klíčové momenty	34
20.4.1 Motivace (0-10 min):	34
20.4.2 Batch processing (10-25 min):	34
20.4.3 Describe + adaptivní zpracování (25-40 min):	34
20.4.4 Try-except (40-55 min):	35
20.4.5 Komplexní příklad (55-70 min):	35
20.5 Rizika	35
20.6 Tipy	35
20.7 Materiály k přípravě	36
20.8 Nejdůležitější sdělení	36
20.9 Co NEDEĚL	36
20.10 Co URČITĚ DĚLAT:	36

1 Cíle lekce

Po absolvování této lekce budete umět:

- Spouštět vektorové nástroje z Pythonu
- Automatizovat analýzy pomocí batch processingu
- Kombinovat nástroje s listing a describe z Lekce 11
- Řetěžit více nástrojů za sebou
- Vytvářet robustní analytické skripty s ošetřením chyb
- Využívat výhody Pythonu: cykly, podmínky, funkce

Časová dotace: 90 minut

2 Proč Python pro vektorové analýzy?

2.1 Co už umíte

V GIS 1 jste se naučili provádět vektorové analýzy v ArcGIS Pro:

- Buffer, Clip, Intersect, Union
- Select by Attributes, Select by Location
- Dissolve, Merge
- Spatial Join

Všechno to umíte klikáním v GUI. Tak proč Python?

2.2 Výhody automatizace pomocí Pythonu

2.2.1 Manuálně v GUI

- 100 vrstev → 100× klikat
- Opakování = nuda + chyby
- Nelze uložit postup
- Těžko reprodukovatelné
- Složité podmínky

2.2.2 Automaticky v Pythonu

- 100 vrstev → 1 cyklus
- Jednou napsat, spustit kdykoliv
- Skript = dokumentace
- Snadno reprodukovatelné
- Libovolně složitá logika

2.3 Praktické situace

Situace 1: “Vytvořte buffer 100 m pro všech 50 vektorových vrstev v geodatabázi.”

→ GUI: 50× klikat

→ Python: 5 řádků kódu

Situace 2: “Buffer 200 m pro polygony, 100 m pro linie, 50 m pro body.”

→ GUI: Třídít ručně, pak buffer po skupinách

→ Python: Describe + if-else

Situace 3: “Ořízněte všechny vrstvy hranicí okresu, ale jen ty ve WGS84.”

→ GUI: Kontrolovat každou vrstvu ručně

→ Python: Describe + podmínka

2.4 Propojení s Lekcí 11

V minulé lekci jste se naučili:

- `ListFeatureClasses()` - seznam vrstev
- `Describe()` - informace o vrstvě
- `Exists()` - kontrola existence
- `try-except` - ošetření chyb

Dnes: Zkombinujete toto s vektorovými analýzami!

3 Od GUI k Pythonu

3.1 Copy Python Command

Nejjednodušší způsob, jak zjistit syntaxi:

1. V ArcGIS Pro spusťte nástroj **manuálně** v Geoprocessing Pane
2. Po dokončení: **Copy Python Command** (pravý dolní roh)
3. Vložte do skriptu

Příklad:

Vytvoříte buffer v GUI → Copy Python Command:

```
arcpy.analysis.Buffer(  
    in_features="C:\\Data\\Projekt.gdb\\silnice",  
    out_feature_class="C:\\Data\\Projekt.gdb\\silnice_Buffer",  
    buffer_distance_or_field="100 Meters",  
    line_side="FULL",  
    line_end_type="ROUND",  
    dissolve_option="NONE"  
)
```

3.2 Struktura příkazu

```
arcpy.<toolbox>.<Tool>(  
    povinný_parametr1,  
    povinný_parametr2,  
    volitelný_parametr=hodnota  
)
```

3.3 Starý způsob volání nástrojů

Ve starších verzích se geoprocessingové nástroje volaly takto:

```
arcpy.<Tool>_<toolbox>(  
    povinný_parametr1,  
    povinný_parametr2,  
    volitelný_parametr=hodnota  
)
```

Tedy například:

```
arcpy.Buffer_analysis("silnice", "buf", "100 Meters")
```

Tento způsob stále funguje a můžete se s ním setkat ve starších skriptech.

Běžné toolboxy:

- `arcpy.analysis` - vektorové analýzy (Buffer, Clip, Intersect)
- `arcpy.management` - správa dat (Delete, CopyFeatures, Dissolve)
- `arcpy.conversion` - konverze formátů

💡 Nemusíte zadávat všechny parametry!

Mnoho parametrů má výchozí hodnoty:

```
# Plná verze z Copy Python Command
arcpy.analysis.Buffer("silnice", "buf", "100 Meters",
                     line_side="FULL", line_end_type="ROUND")

# Stačí:
arcpy.analysis.Buffer("silnice", "buf", "100 Meters")
```

4 Základní vektorové nástroje v Pythonu

4.1 Rychlý přehled syntaxe

Tyto nástroje už znáte z GIS 1. Zde je jen Python syntaxe:

Buffer:

```
arcpy.analysis.Buffer("vstup", "vystup", "100 Meters")
arcpy.analysis.Buffer("vstup", "vystup", "100 Meters", dissolve_option="ALL")
```

Clip:

```
arcpy.analysis.Clip("vrstva", "maska", "vystup")
```

Intersect:

```
arcpy.analysis.Intersect(["vrstva1", "vrstva2"], "vystup")
```

Select:

```
arcpy.analysis.Select("vstup", "vystup", '"populace" > 5000')
```

Dissolve:

```
arcpy.management.Dissolve("vstup", "vystup", "pole")
arcpy.management.Dissolve("vstup", "vystup", "pole",
                           [["Shape_Area", "SUM"]])
```

💡 Neznáte syntaxi?

1. Spustte nástroj v GUI
2. **Copy Python Command**
3. Vložte do skriptu

Nebo: najděte stránku s nápovědou k nástroji. Používejte ArcGIS nápovědu!

5 Batch processing - Propojení s Lekcí 11

5.1 Problém: 50 vrstev, 50× klikat?

Scénář: Máte geodatabázi s 50 vrstvami. Potřebujete vytvořit buffer 100 m pro **všechny** polygonové vrstvy.

Řešení z Lekce 11: ListFeatureClasses() + cyklus!

```
import arcpy

arcpy.env.workspace = r"C:\TEMP\Python_GIS\Lekce_12\cviceni_12.gdb"
arcpy.env.overwriteOutput = True

# Získat všechny polygonové vrstvy (z Lekce 11!)
```



```

polygons = arcpy.ListFeatureClasses(feature_type="Polygon")

print(f"Nalezeno {len(polygons)} polygonových vrstev\n")

# Batch buffer
for fc in polygons:
    vystup = f"{fc}_buffer100"
    print(f"Buffer pro {fc}...")
    arcpy.analysis.Buffer(fc, vystup, "100 Meters")
    print(f"    Hotovo: {vystup}")

print("\nVšechny buffery vytvořeny!")

```

Výsledek: 50 bufferů vytvořeno za pár sekund!

5.2 Batch processing s podmínkou

Scénář: Buffer, ale jen pro vrstvy ve správném souřadnicovém systému.

Řešení: Describe (z Lekce 11!) + podmínka

```

import arcpy

arcpy.env.workspace = r"C:\TEMP\Python_GIS\Lekce_12\cviceni_12.gdb"
arcpy.env.overwriteOutput = True

fc_list = arcpy.ListFeatureClasses()

# Požadovaný souřadnicový systém (EPSG: 5514 = S-JTSK)
pozadovany_epsg = 5514

uspesnych = 0
preskocenych = 0

for fc in fc_list:
    # Describe z Lekce 11!
    desc = arcpy.Describe(fc)
    sr = desc.spatialReference

    # Kontrola souřadnicového systému
    if sr.factoryCode == pozadovany_epsg:

```

```

    # Správný souř. systém → buffer
    vystup = f"{fc}_buffer"
    arcpy.analysis.Buffer(fc, vystup, "100 Meters")
    print(f"Buffer pro {fc} vytvořen.")
    uspesnych += 1
else:
    # Jiný souř. systém → přeskočit
    print(f"{fc} přeskočen (EPSG: {sr.factoryCode}).")
    preskoceny += 1

print(f"\nÚspěšných: {uspesnych}, Přeskočených: {preskoceny}")

```

5.3 Batch processing s try-except

Scénář: Jedna vrstva je poškozená → celý skript spadne?

Řešení: try-except (z Lekce 11!)

```

import arcpy

arcpy.env.workspace = r"C:\TEMP\Python_GIS\Lekce_12\cviceni_12.gdb"
arcpy.env.overwriteOutput = True

fc_list = arcpy.ListFeatureClasses()

uspesnych = 0
chybnych = 0

for fc in fc_list:
    try:
        # Pokus o buffer
        vystup = f"{fc}_buffer"
        arcpy.analysis.Buffer(fc, vystup, "100 Meters")
        print(f"{fc} hotovo.")
        uspesnych += 1

    except arcpy.ExecuteError:
        # Chyba v ArcGIS nástroji
        print(f"{fc} - Chyba: {arcpy.GetMessages(2)}")
        chybnych += 1

except Exception as e:

```

```

# Jiná Python chyba
print(f"{fc} - Python chyba: {str(e)}")
chybných += 1

print(f"\n{'='*50}")
print(f"Úspěšných: {uspesnych}")
print(f"Chybných: {chybných}")

```

Výhoda: Skript pokračuje i při chybě jedné vrstvy!

6 Adaptivní batch processing

6.1 Buffer podle typu geometrie

Scénář: Různé vzdálenosti pro body, linie, polygony.

```

import arcpy

arcpy.env.workspace = r"C:\TEMP\Python_GIS\Lekce_12\cviceni_12.gdb"
arcpy.env.overwriteOutput = True

fc_list = arcpy.ListFeatureClasses()

for fc in fc_list:
    # Zjistit typ geometrie (Describe z Lekce 11!)
    desc = arcpy.Describe(fc)
    shape_type = desc.shapeType

    # Nastavit vzdálenost podle typu
    if shape_type == "Point":
        vzdalenost = "50 Meters"
    elif shape_type == "Polyline":
        vzdalenost = "100 Meters"
    elif shape_type == "Polygon":
        vzdalenost = "200 Meters"
    else:
        print(f"Přeskakuji {fc} - neznámý typ")
        continue

```

```
# Buffer
vystup = f"{fc}_buffer"
print(f"{fc} ({shape_type}): {vzdalenost}")
arcpy.analysis.Buffer(fc, vystup, vzdalenost)
print(f"{vystup}")
```

7 Clip - Ořezání

7.1 Co dělá Clip?

Ořízne vstupní vrstvu podle hranic jiné vrstvy (clip features).

Použití:

- Extrakce dat pro území
- Oříznutí podle administrativní hranice

7.2 Základní použití

```
# Ořízne řeky hranicemi lesa
arcpy.analysis.Clip(
    in_features="reky",
    clip_features="lesy",
    out_feature_class="reky_v lese"
)

print("Oříznutí dokončeno!")
```

7.3 Parametry Clip

```
arcpy.analysis.Clip(
    in_features,      # Vrstva k oříznutí
    clip_features,    # Ořezávací maska
    out_feature_class, # Výstup
```

```
cluster_tolerance=None # Tolerance pro topologii
)
```

Poznámka: Clip zachovává **pouze ty části** vstupní vrstvy, které jsou **uvnitř** clip_features.

7.4 Praktický příklad

```
import arcpy

arcpy.env.workspace = r"C:\TEMP\Python_GIS\Lekce_12\cviceni_12.gdb"
arcpy.env.overwriteOutput = True

# Úloha: Najdi silnice v lese

# 1. Ořízne silnice hranicí lesa
arcpy.analysis.Clip(
    in_features="silnice",
    clip_features="lesy",
    out_feature_class="silnice_v lese"
)

# 2. Zjistí počet úseků
result = arcpy.management.GetCount("silnice_v lese")
count = int(result[0])

print(f"Počet úseků silnic v lese: {count}")
```

8 Intersect - Průnik

8.1 Co dělá Intersect?

Vytvoří **průnik** dvou nebo více vrstev. Výsledek obsahuje pouze plochy, které jsou **ve všech** vstupních vrstvách.

Použití:

- Najdi oblasti, kde se překrývají dva jevy
- Které parcely jsou v záplavové oblasti?

8.2 Základní použití

```
# Průnik silnic a lesů
arcpy.analysis.Intersect(
    in_features=["silnice", "lesy"],
    out_feature_class="silnice_lesy_intersect"
)
```

8.3 Parametry Intersect

```
arcpy.analysis.Intersect(
    in_features,          # Seznam vrstev
    out_feature_class,    # Výstup
    join_attributes="ALL", # ALL | NO_FID | ONLY_FID
    cluster_tolerance=None,
    output_type="INPUT"   # INPUT | LINE | POINT
)
```

8.3.1 Klíčové parametry

in_features: Seznam vrstev

```
# Průnik tří vrstev
arcpy.analysis.Intersect(
    in_features=["parcely", "zaplavove_uzemi", "chranena_uzemi"],
    out_feature_class="vysledek"
)
```

join_attributes:

- "ALL" - všechny atributy ze všech vrstev
- "NO_FID" - bez FID polí
- "ONLY_FID" - jen FID

output_type:

- "INPUT" - geometrie jako vstup (polygon → polygon)
- "LINE" - vždy linie
- "POINT" - vždy body

8.4 Příklad: Které obce jsou v dosahu řek?

```
import arcpy

arcpy.env.workspace = r"C:\TEMP\Python_GIS\Lekce_12\cviceni_12.gdb"
arcpy.env.overwriteOutput = True

# 1. Buffer 500 m kolem řek
arcpy.analysis.Buffer(
    in_features="reky",
    out_feature_class="reky_buffer500",
    buffer_distance_or_field="500 Meters",
    dissolve_option="ALL"
)

# 2. Průnik bufferů s obcemi
arcpy.analysis.Intersect(
    in_features=["obce", "reky_buffer500"],
    out_feature_class="obce_u_rek"
)

# 3. Počet obcí
result = arcpy.management.GetCount("obce_u_rek")
print(f"Počet obcí do 500 m od řeky: {result[0]}")
```

9 Union - Sjednocení

9.1 Co dělá Union?

Vytvoří **sjednocení** polygonových vrstev. Výsledek obsahuje **všechny** plochy ze všech vrstev.

Rozdíl od Intersect:

- **Intersect** = jen překryv
- **Union** = vše + informace o překryvu

9.2 Základní použití

```
# Union lesů a luk
arcpy.analysis.Union(
    in_features=["lesy", "louky"],
    out_feature_class="lesy_louky_union"
)
```

9.3 Parametry Union

```
arcpy.analysis.Union(
    in_features,          # Seznam polygonových vrstev
    out_feature_class,    # Výstup
    join_attributes="ALL", # ALL | NO_FID | ONLY_FID
    cluster_tolerance=None,
    gaps="GAPS"           # GAPS | NO_GAPS
)
```

 Union funguje JEN pro polygony!

Pro linie a body použijte **Intersect**.

9.4 Praktický příklad

```
# Úloha: Zjistí, kde se překrývají okresy a lesy

arcpy.env.workspace = r"C:\TEMP\Python_GIS\Lekce_12\cviceni_12.gdb"
arcpy.env.overwriteOutput = True

# Union okresů a lesů
arcpy.analysis.Union(
    in_features=["okresy", "lesy"],
    out_feature_class="okresy_lesy_union"
)

# Výsledek: každý polygon má atributy jak z okresů, tak z lesů
# Můžeme spočítat rozlohu lesa v každém okrese
```



```
# Sumarizace podle okresu
arcpy.analysis.Statistics(
    in_table="okresy_lesy_union",
    out_table="les_podle_okresu",
    statistics_fields=[["Shape_Area", "SUM"]],
    case_field="nazev_okresu"
)
```

10 Select - Výběr podle atributů

10.1 Co dělá Select?

Vytvoří **novou vrstvu** obsahující pouze prvky splňující **SQL dotaz**.

Použití:

- Filtrace podle hodnot
- Extrakce podmnožiny dat

10.2 Základní použití

```
# Vyber vodní plochy větší než 1000 m²
arcpy.analysis.Select(
    in_features="rybníky",
    out_feature_class="velke_rybniky",
    where_clause='"rozloha" > 1000'
)
```

10.3 Parametry Select

```
arcpy.analysis.Select(
    in_features,          # Vstupní vrstva
    out_feature_class,    # Výstup
    where_clause=None     # SQL dotaz
)
```

10.3.1 Syntaxe where_clause

Porovnání čísel:

```
where_clause = '"populace" > 10000'
where_clause = '"vyska" >= 500 AND "vyska" < 1000'
```

Porovnání textu:

```
where_clause = '"nazev" = \'Brno\''          # Rovná se
where_clause = '"nazev" LIKE \'%Praha%\''      # Obsahuje
where_clause = '"typ" IN (\'A\', \'B\')'        # Je v seznamu
```

! Pozor na uvozovky!

Správně:

```
where_clause = '"nazev" = \'Praha\''
```

Proč?

- Vnější uvozovky "... " jsou Python string
- Uvnitř: "nazev" = název sloupce (geodatabáze používá ")
- Text v SQL: 'Praha' = hodnota (SQL používá ')

10.4 Praktický příklad

```
import arcpy

arcpy.env.workspace = r"C:\TEMP\Python_GIS\Lekce_12\cviceni_12.gdb"
arcpy.env.overwriteOutput = True

# Úloha: Zjistí počet obcí s populací nad 5000

# 1. Select
arcpy.analysis.Select(
    in_features="obce",
    out_feature_class="velke_obce",
    where_clause='"populace" > 5000'
)
```

```
# 2. GetCount
result = arcpy.management.GetCount("velke_obce")
count = int(result[0])

print(f"Počet obcí s populací > 5000: {count}")
```

11 Dissolve - Rozpuštění hranic

11.1 Co dělá Dissolve?

Spojí sousední prvky se stejnou hodnotou atributu do jednoho prvku.

Použití:

- Seskupení podle kategorie
- Vytvoření hranic územních celků

11.2 Základní použití

```
# Rozpusť okresy podle krajů
arcpy.management.Dissolve(
    in_features="okresy",
    out_feature_class="kraje",
    dissolve_field="nazev_kraje"
)
```

11.3 Parametry Dissolve

```
arcpy.management.Dissolve(
    in_features,          # Vstup
    out_feature_class,    # Výstup
    dissolve_field=None,  # Pole pro seskupení (seznam)
    statistics_fields=None, # Statistiky [[pole, typ]]
    multi_part="MULTI_PART" # MULTI_PART | SINGLE_PART
)
```

11.3.1 Klíčové parametry

dissolve__field: Pole pro seskupení

```
# Jeden kraj = jeden polygon
dissolve_field="nazev_kraje"

# Kombinace polí
dissolve_field=["nazev_kraje", "typ_uzemi"]
```

statistics__fields: Sumarizace atributů

```
# Sečti populaci, spočítej průměrnou rozlohu
statistics_fields=[
    ["populace", "SUM"],
    ["rozloha", "MEAN"]
]
```

Typy statistik:

- "SUM" - součet
- "MEAN" - průměr
- "MIN" - minimum
- "MAX" - maximum
- "COUNT" - počet

11.4 Praktický příklad

```
import arcpy

arcpy.env.workspace = r"C:\TEMP\Python_GIS\Lekce_12\cviceni_12.gdb"
arcpy.env.overwriteOutput = True

# Úloha: Zjistí celkovou délku vodních toků podle názvu

# 1. Dissolve podle názvu vodního toku
arcpy.management.Dissolve(
    in_features="reky",
    out_feature_class="reky_dissolve",
    dissolve_field="nazev",
```

```

    statistics_fields=["Shape_Length", "SUM"]
)

# 2. Vypsát výsledky
with arcpy.da.SearchCursor("reky_dissolve", ["nazev", "SUM_Shape_Length"]) as cursor:
    for row in cursor:
        print(f"{row[0]}: {row[1]:.1f} m")

```

12 Práce s výstupy nástrojů

12.1 GetCount - Počet prvků

```

# Získat počet prvků
result = arcpy.management.GetCount("obce")
count = int(result[0])
print(f"Počet obcí: {count}")

```

12.2 GetOutput - Získání výstupu

Některé nástroje **vrací** výsledek:

```

# Buffer vrátí cestu k výstupní vrstvě
result = arcpy.analysis.Buffer("silnice", "silnice_buffer", "100 Meters")

# Získat cestu
output_path = result.getOutput(0)
print(f"Buffer uložen v: {output_path}")

# Použít výstup v dalším nástroji
arcpy.analysis.Clip(output_path, "lesy", "vysledek")

```

12.3 Řetězení nástrojů

```
# Vytvoř buffer a rovnou ho ořízni
buffer_result = arcpy.analysis.Buffer("silnice", "buffer_temp", "100 Meters")
clip_result = arcpy.analysis.Clip(buffer_result, "lesy", "vysledek")

print(f"Hotovo: {clip_result}")
```

13 Řetězení nástrojů

13.1 Použití výstupu jako vstupu

Nástroje vracejí **Result object**, který můžete použít jako vstup:

```
# Buffer -> Clip v jednom řetězci
buffer_result = arcpy.analysis.Buffer("reky", "reky_buf", "100 Meters")
clip_result = arcpy.analysis.Clip(buffer_result, "lesy", "vysledek")

print(f"Výsledek: {clip_result}")
```

13.2 GetCount a GetOutput

```
# Počet prvků
result = arcpy.management.GetCount("obce")
count = int(result[0])
print(f"Počet: {count}")

# Cesta k výstupu
result = arcpy.analysis.Buffer("silnice", "buf", "100 Meters")
output_path = result.getOutput(0)
print(f"Uloženo v: {output_path}")
```

14 Komplexní příklad: Batch analýza s reporting

14.1 Zadání

Pro **všechny** liniové vrstvy v geodatabázi:

1. Vytvoř buffer 50 m
2. Zjisti, kolik polygonů lesa protíná tento buffer
3. Zapiš výsledky do textového souboru

14.2 Řešení

```
import arcpy

# Nastavení
arcpy.env.workspace = r"C:\TEMP\Python_GIS\Lekce_12\cviceni_12.gdb"
arcpy.env.overwriteOutput = True

# Soubor pro report
report_file = open("analyza_linie_lesy.txt", "w", encoding="utf-8")
report_file.write("Analýza průniku linií a lesů\n")
report_file.write("=*50 + "\n\n")

# Získat všechny liniové vrstvy (Lekce 11!)
lines = arcpy.ListFeatureClasses(feature_type="Polyline")

if not lines:
    print("Žádné liniové vrstvy nenalezeny!")
    report_file.close()
else:
    print(f"Nalezeno {len(lines)} liniových vrstev\n")

    for line in lines:
        try:
            print(f"Zpracovávám: {line}")

            # 1. Buffer 50 m
            buffer_name = f"{line}_buffer50_temp"
            arcpy.analysis.Buffer(line, buffer_name, "50 Meters",
                                  dissolve_option="ALL")
```

```

# 2. Intersect s lesy
intersect_name = f"{line}_lesy_temp"
arcpy.analysis.Intersect([buffer_name, "lesy"], intersect_name)

# 3. Počet polygonů
count = int(arcpy.management.GetCount(intersect_name)[0])

# 4. Zapiš do reportu
report_file.write(f"{line}:\n")
report_file.write(f"  Polygonů lesa v bufferu: {count}\n\n")

print(f"  Polygonů lesa: {count}")

# 5. Smazat dočasné vrstvy
arcpy.management.Delete(buffer_name)
arcpy.management.Delete(intersect_name)

except arcpy.ExecuteError:
    error_msg = arcpy.GetMessages(2)
    report_file.write(f"{line}: CHYBA\n {error_msg}\n\n")
    print(f"  Chyba: {error_msg}")

except Exception as e:
    report_file.write(f"{line}: CHYBA\n {str(e)}\n\n")
    print(f"  Python chyba: {str(e)}")

report_file.write("="*50 + "\n")
report_file.write("Analýza dokončena\n")
report_file.close()

print(f"\nHotovo! Report uložen v: analiza_linie_lesy.txt")

```

Co tento skript dělá:

1. **ListFeatureClasses** - najde všechny linie (Lekce 11)
2. **Cyklus** - zpracuje všechny automaticky
3. **Buffer** → **Intersect** - řetězení nástrojů
4. **GetCount** - zjištění počtu
5. **try-except** - ošetření chyb (Lekce 11)
6. **Zápis do souboru** - reporting
7. **Delete** - úklid dočasných vrstev

V GUI: Museli byste toto udělat **ručně pro každou vrstvu!**

15 Funkce pro opakované použití

15.1 Problém: Stejný kód vícekrát?

Pokud často děláte stejnou analýzu, vytvořte **funkci**:

```
import arcpy

def buffer_intersect_count(linie_vrstva, polygon_vrstva, vzdalenost):
    """
    Vytvoří buffer kolem linie a spočítá průnik s polygony.

    Args:
        linie_vrstva: Název liniové vrstvy
        polygon_vrstva: Název polygonové vrstvy
        vzdalenost: Vzdálenost bufferu (str, např. "50 Meters")

    Returns:
        int: Počet polygonů v bufferu
    """
    # Buffer
    buffer_temp = "buffer_temp"
    arcpy.analysis.Buffer(linie_vrstva, buffer_temp, vzdalenost,
                          dissolve_option="ALL")

    # Intersect
    intersect_temp = "intersect_temp"
    arcpy.analysis.Intersect([buffer_temp, polygon_vrstva], intersect_temp)

    # Počet
    count = int(arcpy.management.GetCount(intersect_temp)[0])

    # Úklid
    arcpy.management.Delete(buffer_temp)
    arcpy.management.Delete(intersect_temp)

    return count
```

```
# Použití
arcpy.env.workspace = r"C:\TEMP\Python_GIS\Lekce_12\cviceni_12.gdb"
arcpy.env.overwriteOutput = True

# Kolik lesů je do 50 m od silnic?
pocet = buffer_intersect_count("silnice", "lesy", "50 Meters")
print(f"Lesních polygonů do 50 m od silnic: {pocet}")

# Kolik lesů je do 100 m od řek?
pocet = buffer_intersect_count("reky", "lesy", "100 Meters")
print(f"Lesních polygonů do 100 m od řek: {pocet}")
```

Výhoda: Jednou napsat, používat opakovaně!

16 Shrnutí

16.1 Co jsme se naučili

Spouštění nástrojů z Pythonu - Copy Python Command

Batch processing - jeden skript, mnoho vrstev

Propojení s Lekcí 11: - ListFeatureClasses → cyklus přes vrstvy - Describe → adaptivní zpracování podle typu - try-except → robustní skripty

Řetězení nástrojů - výstup jednoho = vstup druhého

Funkce - opakované použití kódu

Reporting - zápis výsledků do souboru

16.2 Proč je Python lepší než GUI?

Úloha	GUI	Python
Buffer 50 vrstev	50× klikat	1 cyklus
Různé vzdálenosti podle typu	Třídít ručně	if-else + Describe
Zpracovat jen vrstvy v S-JTSK	Kontrolovat ručně	Describe + podmínka
Analýza + report	Notepad	Automatický zápis
Opakovat příští týden	Znovu od začátku	Spustit skript

16.3 Klíčové myšlenky

! Nástroje znáte, Python je automatizuje!

- **Nástroje** jste se naučili v GIS 1 a 2 (Buffer, Clip, ...)
- **Python** umožňuje je spouštět automaticky pro mnoho dat
- **Kombinace** Pythonu + GIS nástrojů = super síla!

16.4 Co bude příště?

V příští lekci (Lekce 13):

- **Rastrové analýzy** - Slope, Aspect, Hillshade
 - **Map Algebra** - rastrová kalkulačka
 - **Batch processing** rastrů - stejný princip jako dnes!
-

17 Praktická cvičení

17.1 Cvičení 1: Batch buffer s reportingem

Vytvořte skript, který:

1. Najde **všechny** bodové vrstvy v geodatabázi
2. Pro každou vytvoří buffer 200 m
3. Zapiš do souboru `report_buffery.txt`:
 - Název vrstvy
 - Počet bodů v původní vrstvě
 - Počet bufferů (mělo by být stejné)

Použijte: - `ListFeatureClasses(feature_type="Point")` - Cyklus - `GetCount()` - Zápis do souboru

17.2 Cvičení 2: Adaptivní batch processing

Vytvořte skript, který:

1. Zpracuje **všechny** vrstvy v geodatabázi
2. **Pouze** pro vrstvy v S-JTSK (EPSG: 5514):
 - Vytvoří buffer (vzdálenost podle typu geometrie)
 - Point: 50 m, Polyline: 100 m, Polygon: 200 m
3. Vrstvy v jiném souř. systému přeskočí s varováním

Použijte: - `ListFeatureClasses()` - `Describe()` - souř. systém + typ geometrie - if-elif-else

17.3 Cvičení 3: Analýza s try-except

Vytvořte robustní skript:

1. Pro **všechny** liniové vrstvy:
 - Vytvoř buffer 50 m
 - Proveď intersect s vrstvou “lesy”
 - Zjisti celkovou délku průniku (`Shape_Length`)
 2. Použij try-except pro ošetření chyb
 3. Na konci vypiš statistiku (úspěšných / chybných)
-

18 Domácí úkol

18.1 Varianta A (základní)

Vytvořte skript pro batch clip:

1. Najděte všechny feature classes v geodatabázi
2. Ořízněte je vrstvou “okres_praha”
3. Výstupy pojmenujte: `<nazev>_praha`
4. Vypište, kolik vrstev bylo oříznutých

Odevzdejte: `du_batch_clip.py` + screenshot výstupu

18.2 Varianta B (střední)

Vytvořte skript, který analyzuje dostupnost:

Zadání: Kolik budov je do 500 m od zastávek MHD?

1. Buffer 500 m kolem zastávek (dissolve!)
2. Intersect s budovami
3. GetCount
4. Výsledek zapiš do `dostupnost_mhd.txt`

Odevzdejte: `du_dostupnost.py` + `dostupnost_mhd.txt`

18.3 Varianta C (pokročilá)

Vytvořte univerzální funkci:

```
def batch_buffer_by_type(workspace, output_folder):  
    """  
    Vytvoří buffery pro všechny vrstvy v workspace.  
    Vzdálenost podle typu geometrie:  
    - Point: 50 m  
    - Polyline: 100 m  
    - Polygon: 200 m  
  
    Pouze vrstvy v S-JTSK (EPSG: 5514).  
    Zapiš report do souboru.  
  
    Returns:  
        tuple: (úspěšných, přeskočených, chybných)  
    """  
    # Váš kód
```

Použití:

```
uspech, preskok, chyby = batch_buffer_by_type(  
    r"C:\Data\Projekt.gdb",  
    r"C:\Data\Buffery.gdb"  
)  
print(f"Výsledek: {uspech} / {preskok} / {chyby}")
```

Odevzdejte: `du_funkce_batch.py` + report soubor

18.4 Varianta D (výzva)

Vytvořte komplexní analytický skript:

Zadání: Najdi “environmentálně citlivé silnice”

Silnice je citlivá, pokud: - Prochází lesem (intersect s lesy) - Je do 200 m od vodního toku (buffer řek) - Je v nadmořské výšce nad 500 m (DEM)

Kroky:

1. Batch processing pro všechny silniční vrstvy
2. Pro každou vrstvu proved všechny tři analýzy
3. Kombinuj výsledky (které úseky splňují všechny 3 podmínky?)
4. Vytvoř HTML report s tabulkou výsledků

Bonus: Mapa s barevným označením citlivých úseků

Odevzdejte: Skript + HTML report + (volitelně) mapa

19 Cheatsheet

```
# === BUFFER ===
# Základní buffer
arcpy.analysis.Buffer("vstup", "vystup", "100 Meters")

# Buffer s rozpuštěním
arcpy.analysis.Buffer("vstup", "vystup", "100 Meters", dissolve_option="ALL")

# Buffer podle atributu
arcpy.analysis.Buffer("vstup", "vystup", "pole_vzdalenost")

# === CLIP ===
# Oříznutí
arcpy.analysis.Clip("vrstva", "maska", "vystup")

# === INTERSECT ===
# Průnik dvou vrstev
arcpy.analysis.Intersect(["vrstva1", "vrstva2"], "vystup")

# Průnik více vrstev
arcpy.analysis.Intersect(["v1", "v2", "v3"], "vystup")

# === UNION ===
# Sjednocení polygonů
arcpy.analysis.Union(["polygon1", "polygon2"], "vystup")

# === SELECT ===
# Výběr podle atributu
arcpy.analysis.Select("vstup", "vystup", '"pole" > 1000')

# Výběr textu
arcpy.analysis.Select("vstup", "vystup", '"nazev" = \'Praha\'')

# Složitější podmínka
arcpy.analysis.Select("vstup", "vystup",
                      '"populace" > 5000 AND "rozloha" < 10')

# === DISSOLVE ===
# Rozpuštění podle pole
arcpy.management.Dissolve("vstup", "vystup", "pole")
```

```

# Se statistikami
arcpy.management.Dissolve("vstup", "vystup", "pole",
                          [["Shape_Area", "SUM"], ["populace", "MEAN"]])

# Všechny prvky do jednoho
arcpy.management.Dissolve("vstup", "vystup")

# === GETCOUNT ===
# Počet prvků
result = arcpy.management.GetCount("vrstva")
count = int(result[0])

# === ŘETĚZENÍ ===
# Použití výstupu jako vstupu
buffer = arcpy.analysis.Buffer("silnice", "buf", "100 Meters")
clip = arcpy.analysis.Clip(buffer, "lesy", "vysledek")

# === KOMPLEXNÍ PŘÍKLAD ===
# Buffer → Clip → GetCount
arcpy.env.workspace = r"C:\Data\Projekt.gdb"
arcpy.env.overwriteOutput = True

# 1. Buffer
arcpy.analysis.Buffer("reky", "reky_buf", "100 Meters", dissolve_option="ALL")

# 2. Clip
arcpy.analysis.Clip("obce", "reky_buf", "obce_u_rek")

# 3. Počet
result = arcpy.management.GetCount("obce_u_rek")
print(f"Počet obcí: {result[0]}")

```


20 Poznámky pro vyučujícího

20.1 Filozofie lekce

KLÍČOVÉ: Studenti už znají nástroje z GIS 1 a 2!

NEZTRÁCET ČAS: - Vysvětlováním CO dělá Buffer - Detaily parametrů (to umí najít v dokumentaci) - Vysvětlováním prostorových analýz

ZAMĚŘIT SE NA: - JAK spustit nástroje z Pythonu - PROČ je automatizace lepší než GUI - Propojení s Lekcí 11 (listing, describe, try-except) - Batch processing = super síla - Praktické příklady s reálným dopadem

20.2 Běžné chyby studentů

```
# 1. Zapomínají workspace
arcpy.analysis.Buffer("silnice", "buf", "100 Meters") # Hledá v aktuálním adresáři!
# Správně:
arcpy.env.workspace = r"C:\Data\Projekt.gdb"

# 2. Zapomínají jednotky
arcpy.analysis.Buffer("vstup", "vystup", 100) # CHYBA!
# Správně:
arcpy.analysis.Buffer("vstup", "vystup", "100 Meters")

# 3. Where clause uvozovky (z GUI zkopírováno špatně)
where = "nazev = 'Praha'" # Nebude fungovat v geodatabázi!
# Správně:
where = '"nazev" = \'Praha\''

# 4. GetCount vrací Result object
count = arcpy.management.GetCount("vrstva") # Result object!
if count > 10: # CHYBA - nelze porovnat Result > int
# Správně:
count = int(arcpy.management.GetCount("vrstva")[0])

# 5. Nepoužívají dissolve_option u Buffer
# Výsledek: tisíce překrývajících se bufferů
# Řešení: dissolve_option="ALL"
```

20.3 Časový plán (90 min)

Čas	Obsah
0-10 min	Motivace: PROČ Python? Copy Python Command
10-25 min	Batch processing s ListFeatureClasses - DEMO
25-40 min	Adaptivní batch s Describe - studenti zkoušejí
40-55 min	Try-except v batch processingu
55-70 min	Komplexní příklad - řetězení + reporting
70-85 min	Praktické cvičení studenty
85-90 min	Shrnutí: nástroje znáte, Python automatizuje!

20.4 Klíčové momenty

20.4.1 Motivace (0-10 min):

- **ZAČÍT:** “Kolik času strávíte klikáním v GUI?”
- Ukázat: 50 vrstev → GUI = 50× klikat, Python = 1 cyklus
- **ZDŮRAZNIT:** “Nástroje už znáte, teď je budete používat chytře!”
- Ukázat Copy Python Command - NĚKOLIKRÁT!

20.4.2 Batch processing (10-25 min):

- **KRITICKÉ:** Propojení s Lekcí 11
- Připomenout: “Minule jste se naučili ListFeatureClasses...”
- Ukázat na projektoru: List → cyklus → buffer
- **NECHAT JE PSÁT SOUČASNĚ** - ne jen koukat

20.4.3 Describe + adaptivní zpracování (25-40 min):

- **PŘIPOMENOUT:** “Z minulé lekce znáte Describe...”
- Ukázat: různé vzdálenosti podle shapeType
- Ukázat: filtrování podle souř. systému
- **ZDŮRAZNIT:** “Tohle v GUI nejde snadno!”

20.4.4 Try-except (40-55 min):

- **PŘIPOMENOUT:** “Minule jste se naučili try-except...”
- Ukázat: batch bez try-except → spadne na první chybě
- Ukázat: batch s try-except → pokračuje
- **KLÍČOVÁ ZPRÁVA:** “Produkční skripty MUSÍ být robustní!”

20.4.5 Komplexní příklad (55-70 min):

- **KOMBINUJE VŠE:** listing + describe + try-except + nástroje
- Projít KROK ZA KROKEM na projektoru
- Ukázat reporting do souboru
- **WOW efekt:** “Tohle by v GUI trvalo hodiny!”

20.5 Rizika

1. Studenti chtějí detaily parametrů (10-25 min → 35 min)

- Řešení: “Detaily v dokumentaci. Dnes se učíte automatizovat!”
- Odkázat na Online Help
- Zdůraznit: Copy Python Command = nejlepší učitel

2. Nepochopí propojení s Lekcí 11

- Řešení: OPAKOVANĚ připomínat: “Z minulé lekce...”
- Na začátku rychlý recap (2 min): “Co jsme se učili minule?”

3. Where clause syntax zmatek

- Řešení: Cheatsheet s příklady
- Zdůraznit: Copy Python Command z GUI!

4. Komplexní příklad nestihnou

- Řešení: Projít společně, poskytnout hotový kód ke studiu
- Zdůraznit: “Toto je vzor pro vaše projekty!”

20.6 Tipy

- **Začít WOW efektem:** “Podívejte, jak Python zpracuje 50 vrstev za 10 sekund”
- **Opakovaně říkat:** “Nástroje už znáte, Python je automatizuje”
- **Copy Python Command** - ukázat minimálně 5×
- **Propojení s Lekcí 11** - připomínat průběžně
- Zdůrazněte: “GUI je OK pro jednu vrstvu, Python pro batch”

- **Praktické příklady:** “Co vám trvalo hodiny, teď 5 minut”
- Ukažte Online dokumentaci jen jednou, pak už ne (Google > mě)

20.7 Materiály k přípravě

- ☐ Geodatabáze cviceni_12.gdb - alespoň 20 vrstev různých typů (pro batch processing)
- ☐ Připravený batch processing skript ke spuštění živě
- ☐ Cheatsheet: where clause syntaxe (vytisknout)
- ☐ Screenshot: Copy Python Command
- ☐ Recap slide: Co jsme se učili v Lekci 11?
- ☐ Řešení všech domácích úkolů
- ☐ Stopky - hlídat čas! (Batch demo nesmí trvat > 15 min)

20.8 Nejdůležitější sdělení

! Sdělení pro studenty

1. **Nástroje už znáte** z GIS 1 a 2
2. **Python automatizuje** to, co umíte ručně
3. **Batch processing** = super síla Pythonu
4. **Propojení s Lekcí 11** = robustní skripty
5. **Copy Python Command** = váš nejlepší učitel

20.9 Co NEDĚL

AT: - Vysvětlovat detailně CO dělá každý nástroj - Probírat všechny parametry - Trávit čas teorií prostorových analýz - Pouštět se do GIS teorie

20.10 Co URČITĚ DĚLAT:

- Ukázat batch processing ŽIVĚ
- Nechat je psát s vámi
- Připomínat Lekci 11 průběžně
- Zdůrazňovat výhody automatizace
- WOW efekt - “vidíte, jak je to rychlé?”