

# Lekce 11: Správa dat a workspace

Python pro GIS - Zjišťování obsahu a práce s daty

Vojtěch Barták, FŽP ČZU Praha

2025-11-19

## Table of contents

<b>1 Cíle lekce</b>	<b>3</b>
<b>2 Proč potřebujeme listing a describe?</b>	<b>3</b>
2.1 Motivace . . . . .	3
2.2 Praktické situace . . . . .	3
<b>3 Zjišťování obsahu workspace</b>	<b>4</b>
3.1 Nastavení workspace . . . . .	4
3.2 ListFeatureClasses - seznam vektorových vrstev . . . . .	4
3.2.1 Filtrování podle wildcard . . . . .	5
3.2.2 Filtrování podle typu geometrie . . . . .	5
3.3 ListRasters - seznam rastrů . . . . .	6
3.3.1 Filtrování podle typu . . . . .	6
3.4 ListDatasets - seznam feature datasetů . . . . .	6
3.5 ListFields - seznam atributů vrstvy . . . . .	7
3.5.1 Vlastnosti Field objektu . . . . .	7
<b>4 Describe object - detailní informace</b>	<b>8</b>
4.1 Co je Describe? . . . . .	8
4.2 Vlastnosti pro Feature Class . . . . .	8
4.3 Vlastnosti pro Raster . . . . .	9
4.4 Praktický příklad - kontrola typu geometrie . . . . .	9
<b>5 Kontrola existence - Exists()</b>	<b>10</b>
5.1 Proč kontrolovat existenci? . . . . .	10
5.2 Základní použití . . . . .	10
5.3 Praktický příklad - kontrola před vytvořením . . . . .	10
5.4 Exists() pro různé typy dat . . . . .	11

<b>6 Batch processing - zpracování více souborů</b>	<b>11</b>
6.1 Základní batch processing . . . . .	11
6.2 Batch processing s filtrem . . . . .	12
6.3 Batch processing s Describe . . . . .	13
<b>7 Ošetření chyb - try-except</b>	<b>13</b>
7.1 Proč ošetřovat chyby? . . . . .	13
7.2 Základní struktura . . . . .	14
7.3 Try-except v ArcPy . . . . .	14
7.4 Praktický příklad - robustní batch processing . . . . .	15
<b>8 Praktická cvičení</b>	<b>17</b>
8.1 Cvičení 1: Inventura geodatabáze . . . . .	17
8.2 Cvičení 2: Kontrola souřadnicových systémů . . . . .	18
8.3 Cvičení 3: Batch Clip . . . . .	18
<b>9 Shrnutí</b>	<b>19</b>
9.1 Co jsme se naučili . . . . .	19
9.2 Kdy použít co? . . . . .	19
9.3 Co bude příště? . . . . .	20
<b>10 Domácí úkol</b>	<b>20</b>
10.1 Varianta A (základní) . . . . .	20
10.2 Varianta B (střední) . . . . .	20
10.3 Varianta C (pokročilá) . . . . .	21
10.4 Varianta D (výzva) . . . . .	22
<b>11 Cheatsheet</b>	<b>23</b>
<b>12 Poznámky pro vyučujícího</b>	<b>26</b>
12.1 Běžné chyby studentů . . . . .	26
12.2 Časový plán (90 min) . . . . .	27
12.3 Klíčové momenty . . . . .	27
12.3.1 Listing funkce (10-30 min): . . . . .	27
12.3.2 Describe (30-45 min): . . . . .	27
12.3.3 Batch processing (45-65 min): . . . . .	27
12.3.4 Try-except (65-80 min): . . . . .	27
12.4 Rizika . . . . .	28
12.5 Tipy . . . . .	28
12.6 Materiály k přípravě . . . . .	28

## 1 Cíle lekce

Po absolvování této lekce budete umět:

- Zjišťovat obsah geodatabáze (ListFeatureClasses, ListRasters, ListFields)
- Používat Describe object pro získání informací o datech
- Kontrolovat existenci dat pomocí Exists()
- Procházet data pomocí cyklů (batch processing)
- Ošetřovat chyby pomocí try-except
- Vytvářet robustní skripty pro automatizaci

**Časová dotace:** 90 minut

---

## 2 Proč potřebujeme listing a describe?

### 2.1 Motivace

V předchozí lekci jste psali skripty s **pevně danými názvy** vrstev:

```
 arcpy.analysis.Buffer("silnice", "silnice_buffer", "100 Meters")
```

**Problémy:**

- Co když nevíte přesně, jak se vrstva jmeneje?
- Co když chcete zpracovat **všechny** vrstvy v geodatabázi?
- Co když potřebujete vědět, **jaký typ** geometrie vrstva má?

**Řešení:** Použít **listing** a **describe** funkce ArcPy!

### 2.2 Praktické situace

#### Situace 1: Batch processing

“Vytvořte buffer 100 m pro **všechny** polylinové vrstvy v geodatabázi.”

→ Nemůžete psát názvy ručně! Musíte zjistit seznam vrstev automaticky.

#### Situace 2: Kontrola dat

“Zkontrolujte, zda vrstva **silnice** existuje. Pokud ne, stáhněte ji z webu.”

→ Potřebujete  `arcpy.Exists()`

### Situace 3: Analýza metadat

“Vypište pro všechny vrstvy jejich typ geometrie a souřadnicový systém.”

→ Potřebujete `Describe` object

---

## 3 Zjištování obsahu workspace

### 3.1 Nastavení workspace

Připomenutí z Lekce 10:

```
import arcpy

# Nastavení geodatabáze jako workspace
arcpy.env.workspace = r"C:\TEMP\Python_GIS\Lekce_11\cviceni_11.gdb"

# Všechny listing funkce budou hledat v této geodatabázi
```

### 3.2 ListFeatureClasses - seznam vektorových vrstev

```
import arcpy

arcpy.env.workspace = r"C:\TEMP\Python_GIS\Lekce_11\cviceni_11.gdb"

# Vypsat všechny feature classes
fc_list = arcpy.ListFeatureClasses()

print("Feature classes v geodatabázi:")
for fc in fc_list:
    print(f" - {fc}")
```

Výstup:

Feature classes v geodatabázi:

- silnice
- rybníky
- lesy
- obce
- reký

### 3.2.1 Filtrování podle wildcard

```
# Jen vrstvy začínající na "r"
fc_list = arcpy.ListFeatureClasses("r*")
print(fc_list) # ['rybníky', 'reký']

# Jen vrstvy obsahující "s"
fc_list = arcpy.ListFeatureClasses("*s*")
print(fc_list) # ['silnice', 'lesy', 'obce']
```

### 3.2.2 Filtrování podle typu geometrie

```
# Jen bodové vrstvy
points = arcpy.ListFeatureClasses(feature_type="Point")
print(points) # ['rybníky', 'obce']

# Jen liniové vrstvy
lines = arcpy.ListFeatureClasses(feature_type="Polyline")
print(lines) # ['silnice', 'reký']

# Jen polygonové vrstvy
polygons = arcpy.ListFeatureClasses(feature_type="Polygon")
print(polygons) # ['lesy']
```

Typy geometrie:

- "Point" - body
- "Multipoint" - multipoint
- "Polyline" - linie
- "Polygon" - polygony
- "Annotation" - anotace

### Kombinace filtrů

```
# Polygony začínající na "l"  
polygons = arcpy.ListFeatureClasses("l*", feature_type="Polygon")
```

## 3.3 ListRasters - seznam rastrů

```
arcpy.env.workspace = r"C:\TEMP\Python_GIS\Lekce_11\cviceni_11.gdb"  
  
# Vypsat všechny rastry  
raster_list = arcpy.ListRasters()  
  
print("Rastry v geodatabázi:")  
for raster in raster_list:  
    print(f" - {raster}")
```

### 3.3.1 Filtrování podle typu

```
# Jen TIFF rastry  
tiff_list = arcpy.ListRasters("*", "TIF")  
  
# Jen IMG rastry  
img_list = arcpy.ListRasters("*", "IMG")
```

## 3.4 ListDatasets - seznam feature datasetů

```
# Vypsat všechny feature datasety  
dataset_list = arcpy.ListDatasets()  
  
for dataset in dataset_list:  
    print(f"Feature dataset: {dataset}")  
  
    # Feature classes uvnitř datasetu  
    arcpy.env.workspace = f"{arcpy.env.workspace}/{dataset}"  
    fc_list = arcpy.ListFeatureClasses()
```

```
for fc in fc_list:  
    print(f" - {fc}")
```

### 3.5 ListFields - seznam atributů vrstvy

```
arcpy.env.workspace = r"C:\TEMP\Python_GIS\Lekce_11\cviceni_11.gdb"  
  
# Vypsat všechny atributy vrstvy "silnice"  
fields = arcpy.ListFields("silnice")  
  
print("Atributy vrstvy 'silnice':")  
for field in fields:  
    print(f" - {field.name} ({field.type})")
```

Výstup:

```
Atributy vrstvy 'silnice':  
- OBJECTID (OID)  
- Shape (Geometry)  
- nazev (String)  
- delka (Double)  
- trida (Integer)  
- Shape_Length (Double)
```

#### 3.5.1 Vlastnosti Field objektu

```
fields = arcpy.ListFields("silnice")  
  
for field in fields:  
    print(f"Název: {field.name}")  
    print(f" Typ: {field.type}")  
    print(f" Délka: {field.length}")  
    print(f" Nullable: {field.isNullable}")  
    print()
```

## 4 Describe object - detailní informace

### 4.1 Co je Describe?

Describe vrací objekt s **detailními informacemi** o datech.

```
import arcpy

arcpy.env.workspace = r"C:\TEMP\Python_GIS\Lekce_11\cviceni_11.gdb"

# Získání Describe objektu
desc = arcpy.Describe("silnice")

# Přístup k vlastnostem
print(f"Název: {desc.name}")
print(f"Typ: {desc.dataType}")
print(f"Cesta: {desc.catalogPath}")
```

### 4.2 Vlastnosti pro Feature Class

```
desc = arcpy.Describe("silnice")

# Základní info
print(f"Název: {desc.name}")
print(f"Typ dat: {desc.dataType} # FeatureClass

# Geometrie
print(f"Typ geometrie: {desc.shapeType} # Polyline
print(f"Has M: {desc.hasM}")
print(f"Has Z: {desc.hasZ}")

# Souřadnicový systém
sr = desc.spatialReference
print(f"Souřadnicový systém: {sr.name}")
print(f"EPSG kód: {sr.factoryCode}")

# Rozsah (extent)
extent = desc.extent
print(f"XMin: {extent.XMin}")
print(f"XMax: {extent.XMax}")
```

```
print(f"YMin: {extent.YMin}")
print(f"YMax: {extent.YMax}")
```

#### 4.3 Vlastnosti pro Raster

```
desc = arcpy.Describe("dem")

# Základní info
print(f"Typ: {desc.dataType}") # RasterDataset

# Rastrové vlastnosti
print(f"Formát: {desc.format}")
print(f"Komprese: {desc.compressionType}")
print(f"Počet pásem: {desc.bandCount}")

# Rozlišení
print(f"Velikost pixelu X: {desc.meanCellWidth}")
print(f"Velikost pixelu Y: {desc.meanCellHeight}")

# Rozsah
print(f"Počet sloupců: {desc.width}")
print(f"Počet řádků: {desc.height}")
```

#### 4.4 Praktický příklad - kontrola typu geometrie

```
def je_polygon(fc_name):
    """Kontrola, zda je feature class polygon."""
    desc = arcpy.Describe(fc_name)
    return desc.shapeType == "Polygon"

# Použití
arcpy.env.workspace = r"C:\TEMP\Python_GIS\Lekce_11\cviceni_11.gdb"

fc_list = arcpy.ListFeatureClasses()

for fc in fc_list:
    if je_polygon(fc):
        print(f"{fc} je polygon")
```

```
    else:  
        print(f"{fc} NENÍ polygon")
```

---

## 5 Kontrola existence - Exists()

### 5.1 Proč kontrolovat existenci?

Problém:

```
arcpy.analysis.Buffer("neexistujici_vrstva", "vystup", "100 Meters")  
# ERROR: neexistujici_vrstva does not exist
```

Řešení: Zkontrolovat před použitím!

### 5.2 Základní použití

```
import arcpy  
  
arcpy.env.workspace = r"C:\TEMP\Python_GIS\Lekce_11\cviceni_11.gdb"  
  
# Kontrola existence feature class  
if arcpy.Exists("silnice"):  
    print("Vrstva 'silnice' existuje")  
else:  
    print("Vrstva 'silnice' NEexistuje")
```

### 5.3 Praktický příklad - kontrola před vytvořením

```
arcpy.env.workspace = r"C:\TEMP\Python_GIS\Lekce_11\cviceni_11.gdb"  
arcpy.env.overwriteOutput = False # Nepřepisovat!  
  
vstup = "silnice"  
vystup = "silnice_buffer"
```

```

# Kontrola vstupu
if not arcpy.Exists(vstup):
    print(f"CHYBA: Vstupní vrstva '{vstup}' neexistuje!")
else:
    # Kontrola výstupu
    if arcpy.Exists(vystup):
        print(f"Varování: '{vystup}' už existuje, přeskakuji.")
    else:
        # Vytvoření bufferu
        arcpy.analysis.Buffer(vstup, vystup, "100 Meters")
        print(f"Buffer vytvořen: {vystup}")

```

## 5.4 Exists() pro různé typy dat

```

# Feature class
arcpy.Exists("silnice")

# Raster
arcpy.Exists("dem")

# Geodatabáze
arcpy.Exists(r"C:\Data\Projekt.gdb")

# Shapefile
arcpy.Exists(r"C:\Data\silnice.shp")

# Složka
arcpy.Exists(r"C:\Data")

```

# 6 Batch processing - zpracování více souborů

## 6.1 Základní batch processing

**Úloha:** Vytvořte buffer 50 m pro všechny feature classes.

```

import arcpy

arcpy.env.workspace = r"C:\TEMP\Python_GIS\Lekce_11\cviceni_11.gdb"
arcpy.env.overwriteOutput = True

# Získat seznam všech feature classes
fc_list = arcpy.ListFeatureClasses()

print(f"Zpracovávám {len(fc_list)} vrstev...")

for fc in fc_list:
    # Název výstupu
    vystup = f"{fc}_buffer50"

    # Vytvoření bufferu
    print(f"  Vytvářím buffer pro: {fc}")
    arcpy.analysis.Buffer(fc, vystup, "50 Meters")
    print(f"    Vytvořeno: {vystup}")

print("Hotovo!")

```

## 6.2 Batch processing s filtrem

**Úloha:** Buffer jen pro **bodové** vrstvy.

```

import arcpy

arcpy.env.workspace = r"C:\TEMP\Python_GIS\Lekce_11\cviceni_11.gdb"
arcpy.env.overwriteOutput = True

# Získat jen bodové vrstvy
points = arcpy.ListFeatureClasses(feature_type="Point")

print(f"Nalezeno {len(points)} bodových vrstev")

for fc in points:
    vystup = f"{fc}_buffer"

    print(f"Buffer pro {fc}...")
    arcpy.analysis.Buffer(fc, vystup, "100 Meters")
    print(f"    Hotovo: {vystup}")

```

## 6.3 Batch processing s Describe

Úloha: Buffer 50 m pro polygony, 100 m pro linie.

```
import arcpy

arcpy.env.workspace = r"C:\TEMP\Python_GIS\Lekce_11\cviceni_11.gdb"
arcpy.env.overwriteOutput = True

fc_list = arcpy.ListFeatureClasses()

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

    # Nastaví vzdálenost podle typu
    if shape_type == "Polygon":
        vzdalenost = "50 Meters"
    elif shape_type == "Polyline":
        vzdalenost = "100 Meters"
    elif shape_type == "Point":
        vzdalenost = "25 Meters"
    else:
        print(f"Přeskakuji {fc} - neznámý typ: {shape_type}")
        continue

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

---

## 7 Ošetření chyb - try-except

### 7.1 Proč ošetřovat chyby?

Problém:

```
# Co když jeden soubor je poškozený?
for fc in fc_list:
    arcpy.analysis.Buffer(fc, f"{fc}_buffer", "100 Meters")
    # CHYBA na 3. vrstvě → celý skript spadne!
```

**Řešení:** Použít try-except pro zachycení chyb!

## 7.2 Základní struktura

```
try:
    # Kód, který může vyvolat chybu
    vysledek = 10 / 0
except:
    # Co udělat při chybě
    print("Nastala chyba!")
```

## 7.3 Try-except v ArcPy

```
import arcpy

arcpy.env.workspace = r"C:\TEMP\Python_GIS\Lekce_11\cviceni_11.gdb"
arcpy.env.overwriteOutput = True

fc_list = arcpy.ListFeatureClasses()

uspesnych = 0
chybnych = 0

for fc in fc_list:
    try:
        vystup = f"{fc}_buffer"
        print(f"Zpracovávám: {fc}")

        arcpy.analysis.Buffer(fc, vystup, "100 Meters")

        print(f"    Úspěch: {vystup}")
        uspesnych += 1
    except:
        chybnych += 1
```

```

except arcpy.ExecuteError:
    # Chyba v geoprocessingu
    print(f"    CHYBA při zpracování: {fc}")
    print(f"    {arcpy.GetMessages()}")
    chybnych += 1

except Exception as e:
    # Jiná chyba (Python)
    print(f"    NEOČEKÁVANÁ CHYBA: {fc}")
    print(f"    {str(e)}")
    chybnych += 1

print(f"\nHotovo! Úspěšných: {uspesnych}, Chybných: {chybnych}")

```

**!** arcpy.ExecuteError vs. Exception

- `arcpy.ExecuteError` - chyby v ArcGIS nástrojích (neexistující vrstva, špatné parametry, ...)
- `Exception` - obecné Python chyby (dělení nulou, chybějící soubor, ...)

**Doporučení:** Zachytávejte obě!

## 7.4 Praktický příklad - robustní batch processing

```

import arcpy

def batch_buffer(workspace, vzdalenost, feature_type=None):
    """
    Vytvoří buffer pro všechny feature classes ve workspace.

    Args:
        workspace: Cesta ke geodatabázi
        vzdalenost: Vzdálenost bufferu (str, např. "100 Meters")
        feature_type: Typ geometrie (volitelné)

    Returns:
        Tuple: (počet úspěšných, počet chybných)
    """
    arcpy.env.workspace = workspace
    arcpy.env.overwriteOutput = True

```

```

# Seznam feature classes
fc_list = arcpy.ListFeatureClasses(feature_type=feature_type)

if not fc_list:
    print("Žádné feature classes k zpracování!")
    return (0, 0)

print(f"Nalezeno {len(fc_list)} vrstev")
print(f"Vzdálenost bufferu: {vzdalenost}\n")

uspesnych = 0
chybnych = 0

for fc in fc_list:
    try:
        # Kontrola existence
        if not arcpy.Exists(fc):
            print(f"    Přeskakuji {fc} - neexistuje")
            chybnych += 1
            continue

        # Název výstupu
        vystup = f"{fc}_buffer"

        print(f"    Zpracovávám: {fc}")

        # Buffer
        arcpy.analysis.Buffer(fc, vystup, vzdalenost)

        print(f"    Hotovo: {vystup}")
        uspesnych += 1

    except arcpy.ExecuteError:
        print(f"    Chyba ArcGIS: {fc}")
        print(f"    {arcpy.GetMessages(2)}") # Jen error zprávy
        chybnych += 1

    except Exception as e:
        print(f"    Python chyba: {fc}")
        print(f"    {str(e)}")
        chybnych += 1

```

```

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

return (uspesnych, chybnych)

# Použití
if __name__ == "__main__":
    workspace = r"C:\TEMP\Python_GIS\Lekce_11\cviceni_11.gdb"

    # Buffer pro všechny vrstvy
    batch_buffer(workspace, "100 Meters")

    # Nebo jen pro polygony
    # batch_buffer(workspace, "50 Meters", feature_type="Polygon")

```

---

## 8 Praktická cvičení

### 8.1 Cvičení 1: Inventura geodatabáze

Vytvořte skript, který:

1. Vypíše **všechny** feature classes v geodatabázi
2. Pro každou vrstvu vypíše:
  - Název
  - Typ geometrie
  - Počet atributů
  - Souřadnicový systém
3. Uloží výsledky do textového souboru **inventura.txt**

**Výstup:**

```

Inventura geodatabáze: cviceni_11.gdb
=====

```

1. silnice
    - Typ geometrie: Polyline
    - Počet atributů: 6
    - Souř. systém: S-JTSK\_Krovak\_East\_North (EPSG: 5514)
  
  2. rybníky
    - Typ geometrie: Point
    - Počet atributů: 4
    - Souř. systém: S-JTSK\_Krovak\_East\_North (EPSG: 5514)
- ...

## 8.2 Cvičení 2: Kontrola souřadnicových systémů

Vytvořte skript, který:

1. Zkontroluje **všechny** feature classes
2. Zjistí, zda mají **stejný** souřadnicový systém
3. Vypíše seznam vrstev s **jiným** souřadnicovým systémem

**Hint:**

```
# Referenční souř. systém (první vrstva)
fc_list = arcpy.ListFeatureClasses()
ref_sr = arcpy.Describe(fc_list[0]).spatialReference

# Kontrola ostatních
for fc in fc_list[1:]:
    desc = arcpy.Describe(fc)
    if desc.spatialReference.name != ref_sr.name:
        print(f"POZOR: {fc} má jiný souř. systém!")
```

## 8.3 Cvičení 3: Batch Clip

Vytvořte skript, který:

1. Ořízne **všechny** feature classes pomocí vrstvy **hranice**
2. Výstupy pojmenuje: <název>\_clip
3. Použije try-except pro ošetření chyb
4. Vypíše statistiku (kolik úspěšných/chybných)

Použití nástroje Clip:

```
arcpy.analysis.Clip(  
    in_features="vstup",  
    clip_features="hranice",  
    out_feature_class="vystup"  
)
```

---

## 9 Shrnutí

### 9.1 Co jsme se naučili

**Listing funkce:**

- `ListFeatureClasses()` - seznam vektorových vrstev
- `ListRasters()` - seznam rastrů
- `ListDatasets()` - seznam feature datasetů
- `ListFields()` - seznam atributů

**Describe object:**

- Detailní informace o datech
- Typ geometrie, souřadnicový systém, extent, ...

**Kontrola existence:**

- `arcpy.Exists()` - před použitím zkontovalovat

**Batch processing:**

- Cykly přes seznam feature classes
- Automatizace pro mnoho souborů

**Ošetření chyb:**

- `try-except` pro robustní skripty
- `arcpy.ExecuteError` vs. `Exception`

### 9.2 Kdy použít co?

Úloha	Funkce
“Co je v geodatabázi?”	ListFeatureClasses()
“Jaký typ geometrie má vrstva?”	Describe().shapeType
“Existuje vrstva?”	Exists()
“Zpracovat všechny vrstvy”	for fc in ListFeatureClasses()
“Zachytit chyby”	try-except

### 9.3 Co bude příště?

V příští lekci (Lekce 12):

- **Vektorové analýzy** - Buffer, Clip, Intersect, Union
  - **Geoprocessingové nástroje** - jak je efektivně používat
  - **Parametry nástrojů** - pokročilé nastavení
  - **Prostorové dotazy** - Select by Location
- 

## 10 Domácí úkol

### 10.1 Varianta A (základní)

Vytvořte skript, který:

1. Vypíše názvy **všech** feature classes v geodatabázi **cviceni\_11.gdb**
2. Pro každou vrstvu vypíše **typ geometrie**
3. Spočítá, kolik je bodových, liniových a polygonových vrstev

**Odevzdejte:** Soubor **inventura.py** + screenshot výstupu

### 10.2 Varianta B (střední)

Vytvořte skript, který:

1. Vytvoří buffer 100 m pro **všechny liniové** vrstvy
2. Výstupy pojmenuje: <název>\_buffer100
3. Vypíše, kolik bufferů bylo vytvořeno

**Použijte:**

- `ListFeatureClasses(feature_type="Polyline")`
- Cyklus `for`

**Odevzdějte:** Soubor `batch_buffer_lines.py` + screenshot výsledků v ArcGIS Pro

### 10.3 Varianta C (pokročilá)

Vytvořte skript `batch_clip_robust.py`, který:

1. Ořízne **všechny** feature classes vrstvou `hranice`
2. Před oříznutím zkонтroluje, zda vrstva existuje
3. Použije `try-except` pro ošetření chyb
4. Na konci vypíše:
  - Počet úspěšně oříznutých vrstev
  - Počet chybných vrstev
  - Seznam chybných vrstev

**Struktura:**

```
import arcpy

arcpy.env.workspace = r"C:\TEMP\Python_GIS\Lekce_11\cviceni_11.gdb"
arcpy.env.overwriteOutput = True

clip_layer = "hranice"

# Kontrola existence hranice
if not arcpy.Exists(clip_layer):
    print(f"CHYBA: '{clip_layer}' neexistuje!")
else:
    fc_list = arcpy.ListFeatureClasses()

    uspesnych = 0
    chybnych = 0
    chybe_vrstvy = []

    for fc in fc_list:
        # Přeskočit samotnou vrstvu hranice
        if fc == clip_layer:
            continue

        try:
```

```

# ... váš kód ...

except:
    # ... váš kód ...

# Výpis statistik
print(f"\nStatistika:")
# ...

```

**Odevzdejte:** Soubor batch\_clip\_robust.py + textový soubor s výstupem skriptu

## 10.4 Varianta D (výzva)

Vytvořte funkci `analyzuj_geodatabazi()`, která:

1. Analyzuje geodatabázi a vrátí **slovník** se statistikami:

```

{
    "pocet_fc": 15,
    "pocet_rasteru": 3,
    "typy_geometrie": {
        "Point": 5,
        "Polyline": 7,
        "Polygon": 3
    },
    "souradnicove_systemy": {
        "S-JTSK": 12,
        "WGS84": 3
    }
}

```

2. Vytvoří HTML report s přehledem (bonus: s grafy!)
3. Uloží report jako `report_geodatabaze.html`

**Odevzdejte:** Soubor analyze\_gdb.py + vygenerovaný HTML report

## 11 Cheatsheet

```
# === LISTING FUNKCÍ ===
import arcpy

arcpy.env.workspace = r"C:\Data\Projekt.gdb"

# Seznam feature classes
fc_list = arcpy.ListFeatureClasses()
fc_list = arcpy.ListFeatureClasses("s*") # Začíná na "s"
fc_list = arcpy.ListFeatureClasses(feature_type="Polygon") # Jen polygony

# Seznam rastrů
raster_list = arcpy.ListRasters()
raster_list = arcpy.ListRasters("dem*", "TIF") # TIFF rastry začínající "dem"

# Seznam feature datasetů
dataset_list = arcpy.ListDatasets()

# Seznam atributů
fields = arcpy.ListFields("silnice")
for field in fields:
    print(f"{field.name} ({field.type})")

# === DESCRIBE OBJECT ===
desc = arcpy.Describe("silnice")

# Základní info
desc.name          # Název
desc.dataType       # Typ dat (FeatureClass, RasterDataset, ...)
desc.catalogPath   # Plná cesta

# Feature class specifické
desc.shapeType      # Typ geometrie (Point, Polyline, Polygon)
desc.hasM            # Má M hodnoty?
desc.hasZ            # Má Z hodnoty?

# Souřadnicový systém
sr = desc.spatialReference
sr.name             # Název souř. systému
sr.factoryCode      # EPSG kód
```

```

# Extent (rozsah)
extent = desc.extent
extent.XMin, extent.XMax, extent.YMin, extent.YMax

# === KONTROLA EXISTENCE ===
if arcpy.Exists("silnice"):
    print("Vrstva existuje")

# Před vytvořením zkontrolovat výstup
if not arcpy.Exists("vystup"):
    arcpy.analysis.Buffer("vstup", "vystup", "100 Meters")

# === BATCH PROCESSING ===
fc_list = arcpy.ListFeatureClasses()

for fc in fc_list:
    vystup = f"{fc}_buffer"
    arcpy.analysis.Buffer(fc, vystup, "100 Meters")

# Jen pro určitý typ
lines = arcpy.ListFeatureClasses(feature_type="Polyline")
for line in lines:
    arcpy.analysis.Buffer(line, f"{line}_buffer", "50 Meters")

# === TRY-EXCEPT ===
try:
    arcpy.analysis.Buffer("vstup", "vystup", "100 Meters")
    print("Hotovo")
except arcpy.ExecuteError:
    # Chyba ArcGIS nástroje
    print("Chyba ArcGIS:")
    print(arcpy.GetMessages())
except Exception as e:
    # Jiná Python chyba
    print(f"Python chyba: {str(e)}")

# Batch s ošetřením chyb
uspesnych = 0
chybnych = 0

for fc in fc_list:
    try:

```

```
    arcpy.analysis.Buffer(fc, f"{fc}_buffer", "100 Meters")
    uspesnych += 1
except:
    print(f"Chyba u {fc}")
    chybnych += 1

print(f"Úspěch: {uspesnych}, Chyba: {chybnych}")

# === PRAKTICKÉ KOMBINACE ===
# Najít všechny polygony a vytvořit buffer
polygons = arcpy.ListFeatureClasses(feature_type="Polygon")
for poly in polygons:
    if arcpy.Exists(poly):
        try:
            desc = arcpy.Describe(poly)
            print(f"Zpracovávám: {poly} ({desc.shapeType})")
            arcpy.analysis.Buffer(poly, f"{poly}_buffer", "50 Meters")
        except:
            print(f" Chyba u {poly}")
```

## 12 Poznámky pro vyučujícího

### 12.1 Běžné chyby studentů

```
# 1. Zapomínají nastavit workspace
fc_list = arcpy.ListFeatureClasses() # Hledá v aktuálním adresáři!
# Správně:
arcpy.env.workspace = r"C:\Data\Projekt.gdb"
fc_list = arcpy.ListFeatureClasses()

# 2. Neošetří prázdný seznam
fc_list = arcpy.ListFeatureClasses("neexistující") # []
for fc in fc_list: # Neprojde ani jednou - žádná chyba, ale nic se nestane
    ...
# Řešení: zkontrolovat if fc_list:

# 3. Zapomínají přeskočit vrstvu při batch clip
for fc in fc_list:
    arcpy.analysis.Clip(fc, "hranice", f"{fc}_clip")
    # Pokusí se oříznout i "hranice" samy sebou!
# Řešení: if fc != "hranice": continue

# 4. Nepochopí rozdíl mezi Describe a ListFields
desc = arcpy.Describe("silnice")
desc.fields # CHYBA - Describe nemá .fields
# Správně:
fields = arcpy.ListFields("silnice")

# 5. Try-except zachytí chybu, ale skript pokračuje
for fc in fc_list:
    try:
        arcpy.Buffer_analysis(fc, f"{fc}_buffer", "100")
        # Zapomněli "Meters" - chyba!
    except:
        pass # Tichá smrt - žádný výpis!
# Řešení: vždy vypsat chybu v except bloku

# 6. Špatný formát vzdálenosti
arcpy.analysis.Buffer("vstup", "vystup", 100) # CHYBA - musí být string!
# Správně:
arcpy.analysis.Buffer("vstup", "vystup", "100 Meters")
```

## 12.2 Časový plán (90 min)

Čas	Obsah
0-10 min	Motivace, proč listing a describe
10-30 min	Listing funkce (demo + studenti zkouší)
30-45 min	Describe object, kontrola existence
45-65 min	Batch processing - společné příklady
65-80 min	Try-except, robustní skripty
80-90 min	Praktické cvičení, shrnutí, zadání DÚ

## 12.3 Klíčové momenty

### 12.3.1 Listing funkce (10-30 min):

- **UKÁZAT** všechny na projektoru v Python Window
- Nechat studenty vyzkoušet s jejich daty
- Zdůraznit: "Tohle je základ pro automatizaci!"

### 12.3.2 Describe (30-45 min):

- Prakticky ukázat shapeType, spatialReference
- Říct: "Describe = detektiv - zjistí vše o datech"
- **POZOR:** Studenti pletou Describe a ListFields

### 12.3.3 Batch processing (45-65 min):

- Začít jednoduchým příkladem (buffer pro všechny)
- Postupně přidat Describe (různé vzdálenosti podle typu)
- **DŮLEŽITÉ:** Projít krok za krokem, nechat je psát

### 12.3.4 Try-except (65-80 min):

- Vysvětlit: "Co když jeden soubor je poškozený?"
- Ukázat skript bez try-except (spadne)
- Ukázat stejný skript s try-except (pokračuje)
- **KRITICKÉ:** Vždy něco vypsat v except bloku!

## 12.4 Rizika

1. Listing může být matoucí (10-30 min → 35 min)
  - Řešení: Praktické ukázky v Python Window
  - Mít připravená data s různými typy vrstev
2. Describe vs. ListFields zmatek
  - Řešení: Jasně rozlišit na projektoru
  - Describe = info o vrstvě, ListFields = seznam atributů
3. Try-except studenti nepochopí
  - Řešení: Nejdřív ukázat skript BEZ try-except (spadne)
  - Pak STEJNÝ skript S try-except (pokračuje)
  - Zdůraznit: "Takhle budete psát produkční skripty!"
4. Batch processing nestihnou
  - Řešení: První příklad společně, druhý jako DÚ

## 12.5 Tipy

- Mějte připravenou geodatabázi s různými typy vrstev
- Na projektoru ukazujte **Python Window** + ArcGIS Pro vedle sebe
- Postupná eskalace složitosti (jednoduchý batch → s filtrem → s try-except)
- Zdůrazněte: "Describe a Exists = základ každého robustního skriptu"
- Cheatsheet vytiskněte - studenti ho budou často používat
- Ukažte **GetMessages()** pro zjištění, co se pokazilo

## 12.6 Materiály k přípravě

- Geodatabáze **cviceni\_11.gdb** s 10+ feature classes různých typů
- Ukázková vrstva **hranice** pro clip
- Příklady kódu pro všechny listing funkce
- Vzorový robustní batch processing skript
- Řešení všech domácích úkolů (A, B, C, D)
- Diagram: Kdy použít co? (flowchart)