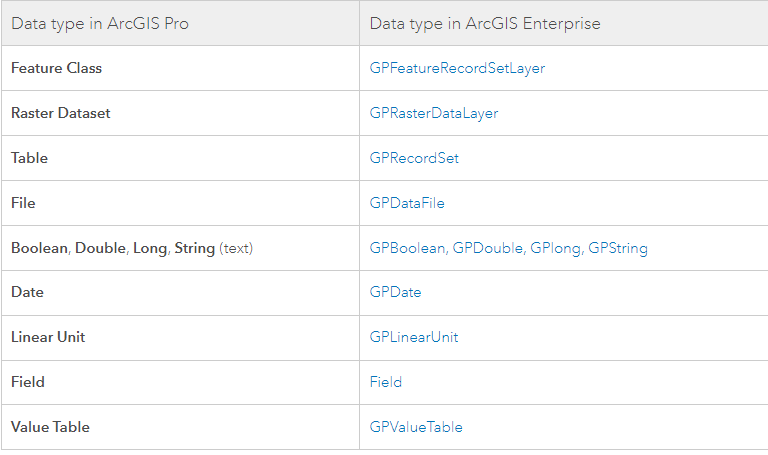
Detalhes Pertinentes

Datatypes:



1. A função arcpy.AddMessage() não funciona, sendo necessário jogar todo “feedback” e log de texto para um arquivo texto e destina-lo como output, foi criada a função portalAddMessage para encurtar essa tarefa:

|  |
| --- |
| def portalAddMessage(paramOutputIndex, outputFileName, output):  with open(arcpy.env.scratchFolder + '\\{0}.txt'.format(outputFileName), 'w') as f:  f.write(str(output))  arcpy.SetParameter(paramOutputIndex, arcpy.env.scratchFolder + '\\{0}.txt'.format(outputFileName)) |

Exemplo de utilização:

|  |
| --- |
| param\_input = arcpy.Parameter(  name="input",  displayName="Input Features",  direction="Input",  parameterType="Required",  datatype="GPFeatureRecordSetLayer"  )  param\_output = arcpy.Parameter(  name="output",  displayName="Output Features",  direction="Output",  parameterType="Derived",  datatype="DEFile"  )  result = ‘abcde’  portalAddMessage(1, 'result', result) |

1. Para conseguir inserir uma camada no input de uma web tool, o datatype do parametro deve ser “GPFeatureRecordSetLayer”.  
   Exemplo:

|  |
| --- |
| param\_input = arcpy.Parameter(  name="in\_features",  displayName="Input Features",  direction="Input",  parameterType="Required",  datatype="GPFeatureRecordSetLayer"  ) |

O tipo da camada que será aceita como input (linha, ponto, polígono) será atrelada ao tipo da camada que foi passada como parâmetro no momento que a tool é transferida para o portal, ou seja, se uma tool, no momento que é transferida pro portal é executada utilizando uma camada de linhas, a tool do portal só aceitará camadas de linhas.

3. layer.value no ArcGIS Pro retorna o type <class 'MappingLayerObject'> enquanto no WebApp retorna o type <class 'geoprocessing record set object'>

4. Manipulação de arquivos para output devem ser todos trabalhados utilizando a Scratch Folder ou in\_memory (ambos apontam para scratch folder)

Exemplo:

|  |
| --- |
| f = open(arcpy.env.scratchFolder+'/testfile.txt', "a")  f.write("isso é um exemplo")  f.close()  arcpy.SetParameter(1, arcpy.env.scratchFolder+'/testfile.txt') |

5. Para editar um campo de uma camada do mapa é possível passa-la como parametro e utilizar tanto UpdateCursor quanto CalculateField para realizar a edição do campo

6. As funções arcpy.management.Copy / CopyFeatures não funcionam – Talvez funcionem em versões mais recentes do portal.

7. Parâmetros do tipo “Field” não são até então suportados

8. Quando é passado as seleções de uma camada como parâmetro de entrada é criada uma cópia in\_memory apenas com as feições selecionadas, impossibilitando edição da camada original. [CASO AS FEIÇÕES SEJAM SELECIONADAS COM O “WIDGET SELECIONAR” A TOOL USARÁ OS CAMPOS DA CAMADA DEFAULT (A CAMADA PASSADA NA EXECUÇÃO DA TOOL DENTRO DO PRO), DEVERÁ SER USADO A FERRAMENTA DE SELEÇÃO PRESENTE NO WIDGET DE GEOPROCESSAMENTO]

9. Ao colocar um DEWorkspace como parâmetro da tool, o parâmetro fica como uma constante, será sempre o valor selecionado no ArcGIS Pro anterior ao compartilhamento da tool para o portal, o usuário não é capaz de selecionar um workspace de destino.

10. InsertCursor funciona normalmente

12. Output a partir de pdf aparentemente funciona, porém necessita da utilização de uma biblioteca para edição/criação de pdfs

13. Para manipulação de itens contidos no portal é necessário utilizar a classe “GIS” para fazer essa conexão com itens do portal, a instanciação do objeto deve ser feita com o parâmetro “verify\_cert”=False

Segue exemplo:

|  |
| --- |
| import arcgis.gis  gis = arcgis.gis.GIS('home', verify\_cert=False) |

14.Não é possível pegar o objeto de relacionamento entre duas camadas, porém é possível pegar suas camadas relacionadas pelo id do item no portal, segue exemplo:

|  |
| --- |
| gis = arcgis.gis.GIS('home', verify\_cert=False)  item = gis.content.get(ID do item no portal) |

Com isso é possível percorrer o objeto item:

|  |
| --- |
| for i in item.layers:  i.properties |

Também possível realizar queries em cima de cada objeto dentro do objeto item

Ex. de query:

|  |
| --- |
| query\_result1 = feature\_layer.query(where='POP2010>1000000', out\_fields='WHITE, BLACK, MULT\_RACE, HISPANIC')  len(query\_result1.features)  query\_result1.fields |

Output:

|  |
| --- |
| 9  [{'name': 'FID',  'type': 'esriFieldTypeOID',  'alias': 'FID',  'sqlType': 'sqlTypeInteger',  'domain': None,  'defaultValue': None},  {'name': 'WHITE',  'type': 'esriFieldTypeInteger',  'alias': 'WHITE',  'sqlType': 'sqlTypeInteger',  'domain': None,  'defaultValue': None},  {'name': 'BLACK',  'type': 'esriFieldTypeInteger',  'alias': 'BLACK',  'sqlType': 'sqlTypeInteger',  'domain': None,  'defaultValue': None},  {'name': 'MULT\_RACE',  'type': 'esriFieldTypeInteger',  'alias': 'MULT\_RACE',  'sqlType': 'sqlTypeInteger',  'domain': None,  'defaultValue': None},  {'name': 'HISPANIC',  'type': 'esriFieldTypeInteger',  'alias': 'HISPANIC',  'sqlType': 'sqlTypeInteger',  'domain': None,  'defaultValue': None}] |

Documentação completa de manipulação de feature layers no portal: <https://developers.arcgis.com/python/guide/working-with-feature-layers-and-features/>

14. O “widget de filtro” serve somente para visualização dos valores filtrados da camada, não possibilita a seleção dos dados para manipulação.

15. Para abrir uma “edit session” deverá ser feita com os seguintes parâmetros:

edit = arcpy.da.Editor(arcpy.**env**.scratchGDB)

edit.startEditing(***with\_undo***=False, ***multiuser\_mode***=False)

edit.startOperation()

....... [TRECHO DE CÓDIGO] ........

edit.stopOperation()

edit.stopEditing(***save\_changes***=True)

Possibilidade de criação das tools

* 1. Exportar/fazer download de um arquivo .txt com "Hello World"

É possível, basta definir um parâmetro de output com o datatype “DEFile”, criar o arquivo com “open()” no diretório scratchFolder ou in\_memory e direcionar esse arquivo para o parâmetro de output usando SetParameter.

É muito útil para realização de log ou saídas de texto, visto que na web tool não tem um meio de saída para outputs texto, tornando funções como “AddMessage()” inutilizáveis. Segue mais informações na seção “detalhes pertinentes” item.1

Ex:

f = open(arcpy.env.scratchFolder + ‘arquivo.txt’, ‘w’)

f.write(‘aaaa’)

f.close()

arcpy.SetParameter(0, f.name)

Verificar arquivo “export\_txt\_file.pyt” para exemplo.

* 2. Exportar/fazer download de um arquivo .csv com qualquer dado.

É possível, basta definir um parâmetro de output com o datatype “DEFile”, criar o arquivo com “open()” no diretório scratchFolder ou in\_memory e direcionar esse arquivo para o parâmetro de output usando SetParameter.

Ex:

f = open(arcpy.env.scratchFolder + ‘arquivo.csv’, ‘w’)

f.write(‘campo1;campo2;campo3;campo4’)

f.close()

arcpy.SetParameter(0, f.name)

Verificar arquivo “export\_csv\_file.pyt” para exemplo.

* 3. Permitir o usuário selecionar uma camada do mapa e exportar um arquivo com todos os dados da camada

Pode ser feito utilizando SearchCursor em cima da entrada da ferramenta de seleção do webapp, o qual passa uma cópia da layer apenas com os registros selecionados, e criando/formatando a exportação pro tipo de arquivo desejado.

Ex arquivo csv:

layer = parameters[0]

f=open(arcpy.env.scratchFolder + ‘arquivo.csv’, ‘w’)

with arcpy.da.SearchCursor(layer, ‘\*’) as cursor:

for row in cursor:

i=0

for col in row:

f.write(str(col))

if i!=len(row)-1:

f.write(‘;’)

i+=1

f.write(‘\n’)

Verificar aquivo “export\_layer\_data.pyt” para exemplo

* 4. Criar uma camada no mapa com informações de alguma outra camada no mapa.

Basta criar uma Feature com os dados da camada de origem, utilizando SearchCursor e InsertCursor e redireciona-la para output com arcpy.SetParameter()

Verificar arquivo “layer\_to\_layer.pyt” para exemplo

* 5. Alimentar uma camada no mapa com alguns registros.

Pode ser realizado normalmente com InsertCursor

Verificar arquivo “layer\_to\_layer.pyt” para exemplo

* 6. Editar um registro específico (digamos, de objectid 57) de alguma camada do mapa.

Pode ser feito normalmente utilizando as ferramentas de edição UpdateCursor ou CalculateField. Não é possível editar uma camada passando seleções como parâmetro, por exemplo com o widget Selecionar, pois seleções são passadas como cópia.

Verificar arquivo “update\_field.pyt” para exemplo

* 7. Permitir o usuário selecionar uma camada do mapa e exportar um .csv apenas com as feições que estiverem selecionadas pelo widget "Selecionar**".**

Necessário apenas percorrer o input da tool com SearchCursor e redirecionar os dados para um csv, já que o input utilizando o widget “Selecionar” cria uma cópia da layer apenas com aqueles registros.

Verificar arquivo “export\_layer\_data.pyt” para exemplo

* 8. Exportar um pdf com dados de alguma camada

É possível exportar um pdf, porém necessita de um módulo de edição de pdfs em python para realizar a formatação correta do pdf.

* 9. Identificar os *relationships* entre camadas no mapa e apresentar uma texto com a estrutura de relacionamentos.

Não é possível pegar o objeto de relacionamento entre duas camadas, porém é possível obter os registros relacionados (tendo conhecimento prévio de seu relacionamento), utilizando o id do item no portal é possível pegar as layers que se relacionam, em cima disso é possível executar querys para puxar os registros desejados.

Verificar arquivo “relationships.pyt” para exemplo