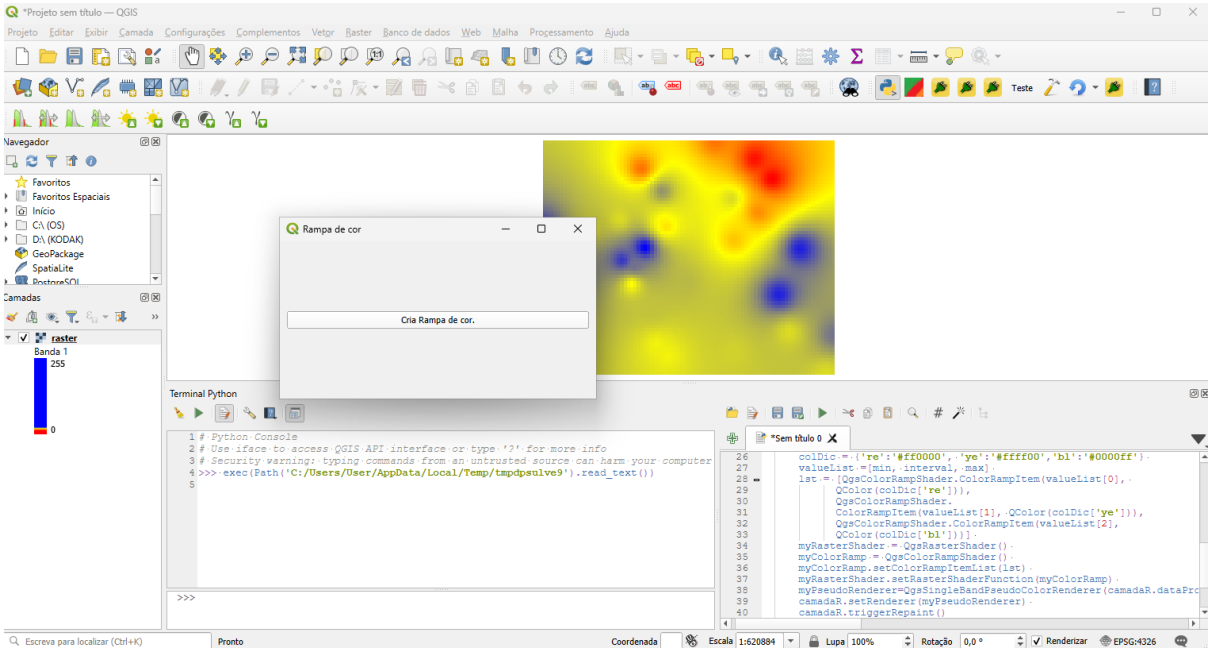


Usando pyQGIS

Classes, Informações Adicionais e Cheat Sheet

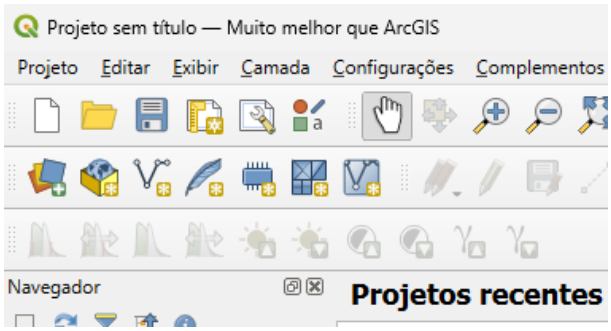
iface - QgisInterface

A classe QgisInterface e iniciada automaticamente quando o QGIS é iniciado e ela é representada pela variável iface (de interface). O iface é o seu programa QGIS e representa a tela principal, os menus, a barra de ferramentas e as demais partes do programa.

No módulo anterior usamos o iface em duas ocasiões, uma para acessar uma ferramenta da barra de ferramenta de raster, e outra para carregar um raster gerado por interpolação. Vamos agora ver outras funções da classe iface.

Mudando o título do nome do programa

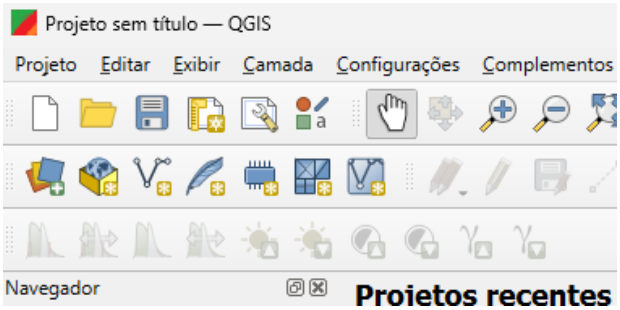
```
titulo = iface.mainWindow().windowTitle()
novo_titulo = titulo.replace('QGIS', 'Muito melhor que ArcGIS')
iface.mainWindow().setWindowTitle(novo_titulo)
```



Mudando o ícone do programa

Mudamos o ícone usando o código abaixo. O **os.path.expanduser('~')** te leva para o pasta raiz de usuário no seu sistema.

```
import os
icone = 'icon.png'# use um arquivo png 64x64 pixels
dire = os.path.join(os.path.expanduser('~'), 'desktop/dash/')
path = os.path.join(dire, icone)
icone2 = QIcon(path)
iface.mainWindow().setWindowIcon(icone2)
```



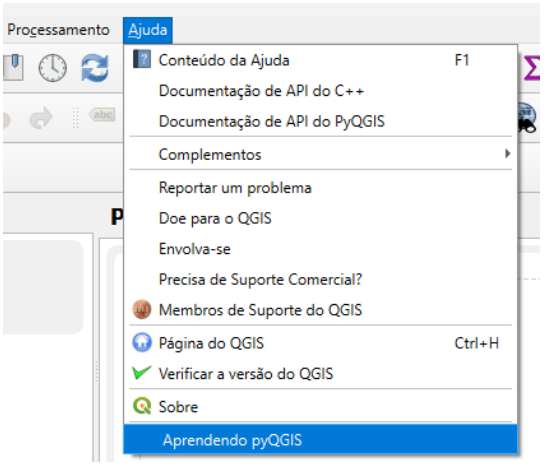
Adicionando um novo item de menu ajuda

Com o seguinte código adicionamos um nove item no menu Ajuda que direciona para a nossa página do Aprendendo pyQGIS.

```
import webbrowser
def abreSite():
```

```
    webbrowser.open('https://gdatasystems.com/pyqgis')

acao = QAction('Aprendendo pyQGIS ')
acao.triggered.connect(abreSite)
iface.helpMenu().addSeparator()
iface.helpMenu().addAction(acao)
```

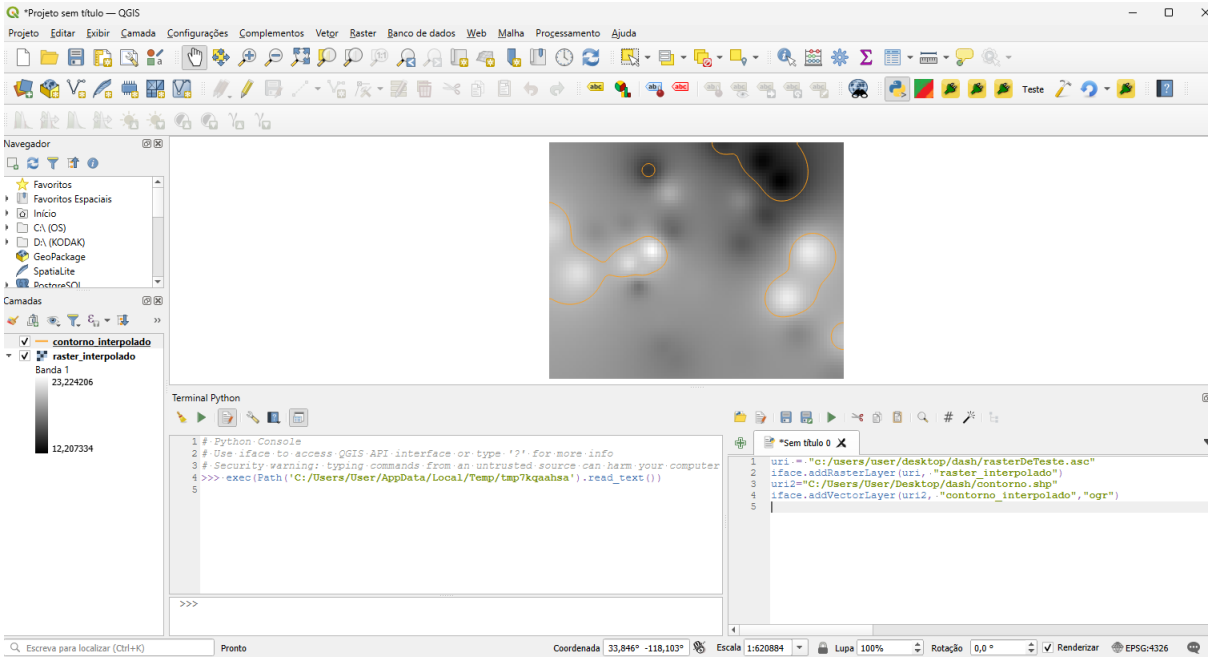


Clicando no item a página vai abrir no navegador.

Adicionando uma nova camada

Como já fizemos no módulo anterior podemos adicionar novas camadas usando o iface.

```
uri = "c:/users/user/desktop/dash/rasterDeTeste.asc"
iface.addRasterLayer(uri, "raster_interpolado")
uri2="C:/Users/User/Desktop/dash/contorno.shp"
iface.addVectorLayer(uri2, "contorno_interpolado","ogr")
```



PyQt

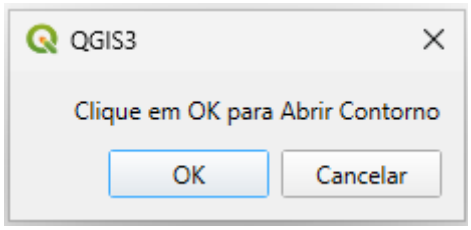
Qt é um conjunto de ferramentas ou plataforma open-source de widget (controles do tipo botões, entrada de texto, rótulos, etc) para criar interfaces gráficas de usuário (GUI) e aplicativos multiplataforma. O QGIS é construído sobre esta plataforma. O PyQt é a interface Python do Qt. O PyQt fornece classes e funções de interação com os widgets QT.

QMessageBox

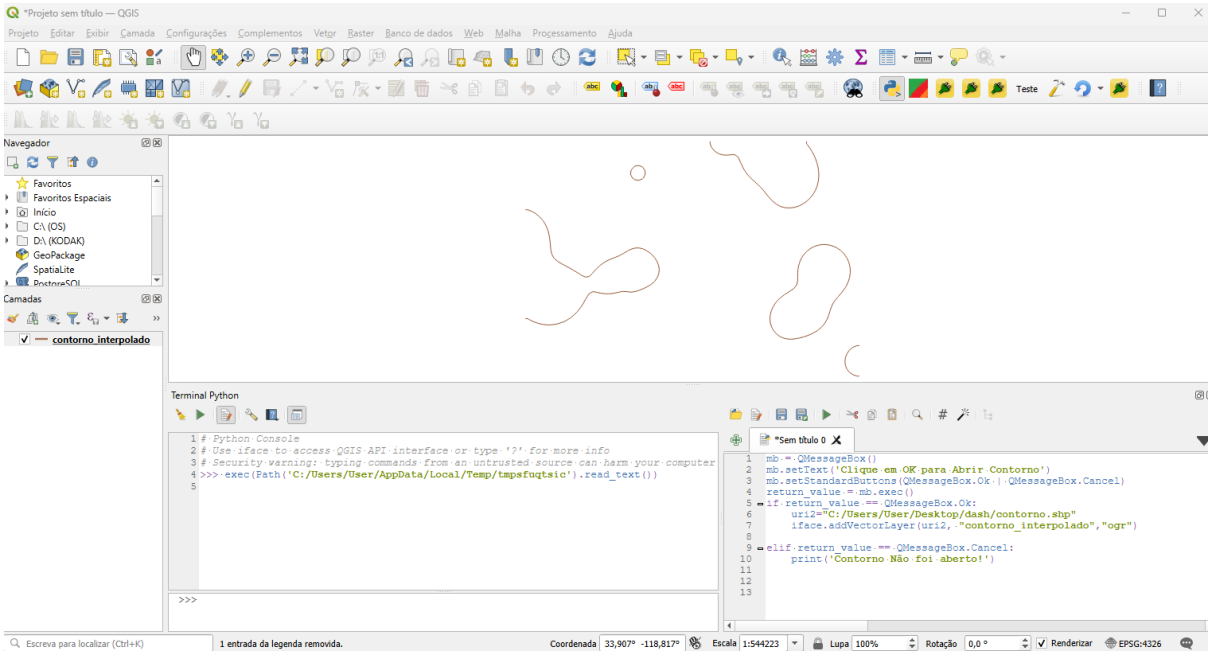
Um exemplo de como usar a classe Qt diretamente no QGIS.

```
mb = QMessageBox()
mb.setText('Clique em OK para Abrir Contorno')
mb.setStandardButtons(QMessageBox.Ok | QMessageBox.Cancel)
return_value = mb.exec()
if return_value == QMessageBox.Ok:
    uri2="C:/Users/User/Desktop/dash/contorno.shp"
    iface.addVectorLayer(uri2, "contorno_interpolado","ogr")

elif return_value == QMessageBox.Cancel:
    print('Contorno Não foi aberto!')
```



Ao clicar OK teremos:

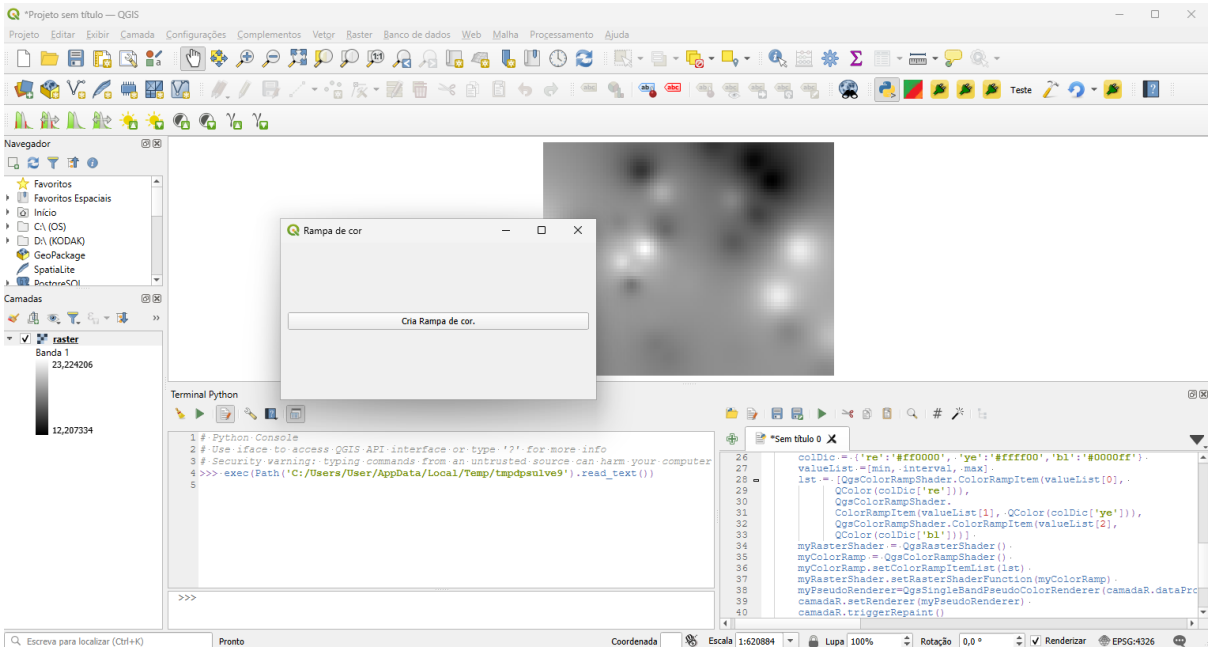


Agora somente um exemplo mais elaborado para ilustrar como a interação entre QGIS e QT pode ser feita, mas não se preocupe, temos os Plugins para facilitar tudo. Isso é somente uma demonstração. Para os pligins usaremos o QtDesigner para criar as Interfaces Gráficas visualmente.

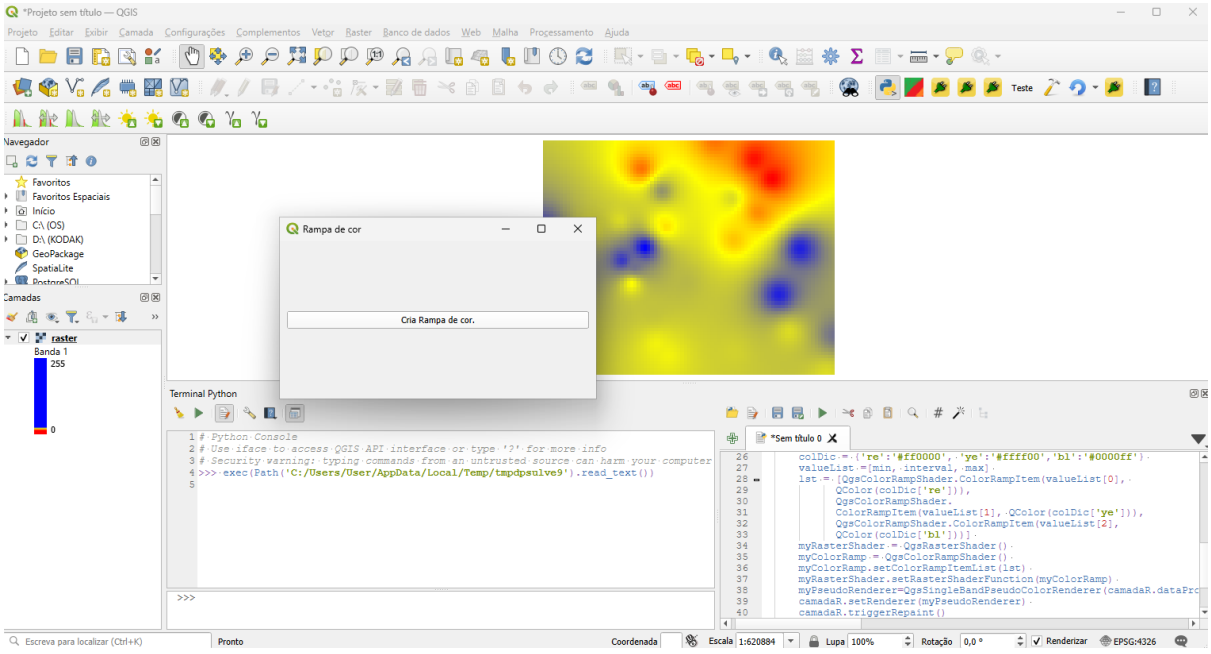
```
class MyWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Rampa de cor")
        self.setGeometry(100, 100, 400, 200)
        central_widget = QWidget()
        self.setCentralWidget(central_widget)
        dialog_button = QPushButton("Cria Rampa de cor.")
        dialog_button.clicked.connect(self.rampa)
        layout = QVBoxLayout()
        layout.addWidget(dialog_button)
        central_widget.setLayout(layout)

    def rampa(self):
        camadaR = iface.activeLayer()
        renderer = camadaR.renderer()
        provider = camadaR.dataProvider()
        rect = camadaR.extent()
        stats = provider.bandStatistics(1,
                QgsRasterBandStats.All,rect, 0)
        min= stats.minimumValue
        max = stats.maximumValue
        range = max - min
        add = range//2
        interval = min + add
        colDic = {'re':'#ff0000', 'ye':'#ffff00','bl':'#0000ff'}
        valueList =[min, interval, max]
        lst = [QgsColorRampShader.ColorRampItem(valueList[0],
              QColor(colDic['re'])),
              QgsColorRampShader.
              ColorRampItem(valueList[1], QColor(colDic['ye'])),
              QgsColorRampShader.ColorRampItem(valueList[2],
              QColor(colDic['bl']))]
        myRasterShader = QgsRasterShader()
        myColorRamp = QgsColorRampShader()
        myColorRamp.setColorRampItemList(lst)
        myRasterShader.setRasterShaderFunction(myColorRamp)
        myPseudoRenderer=
            QgsSingleBandPseudoColorRenderer(camadaR.dataProvider(),
            camadaR.type(), myRasterShader)
        camadaR.setRenderer(myPseudoRenderer)
        camadaR.triggerRepaint()
arquivo = "c:/users/user/desktop/dash/rasterDeTeste.asc"
iface.addRasterLayer(arquivo, "raster")
window = MyWindow()
window.show()
```

Ao executar teremos:



E clicando no botão teremos:



Executando um script pyQGS fora do Qgis

Podemos executar processamentos do QGIS sem iniciar a interface gráfica, usando somente scripts. Para fazer isso temos que criar um script com a seguinte estrutura mínima.

```
from qgis.core import *
# indique onde o programa qgis está localizado no seu computador
QgsApplication.setPrefixPath("/usr", True)
# Crie uma referência à QgsApplication. Usando False como segundo
argumento
# para não ativar a interface gráfica.
qgs = QgsApplication([], False)
# inicie o qgis
qgs.initQgis()
############################################
# Escreva o código de processamento aqui###
############################################
# finalize o script usando:
qgs.exitQgis()
```

Vamos mostrar como isso funciona criando um script que criará um grid raster a partir de um arquivo texto CSV com alguns pontos. O mesmo procedimento do último exemplo do mõdulo anterior. Nomeie o arquivo de **criaRaster.py**. Substitua users/user/desktop/dash apropriadamente para o seu sistema.

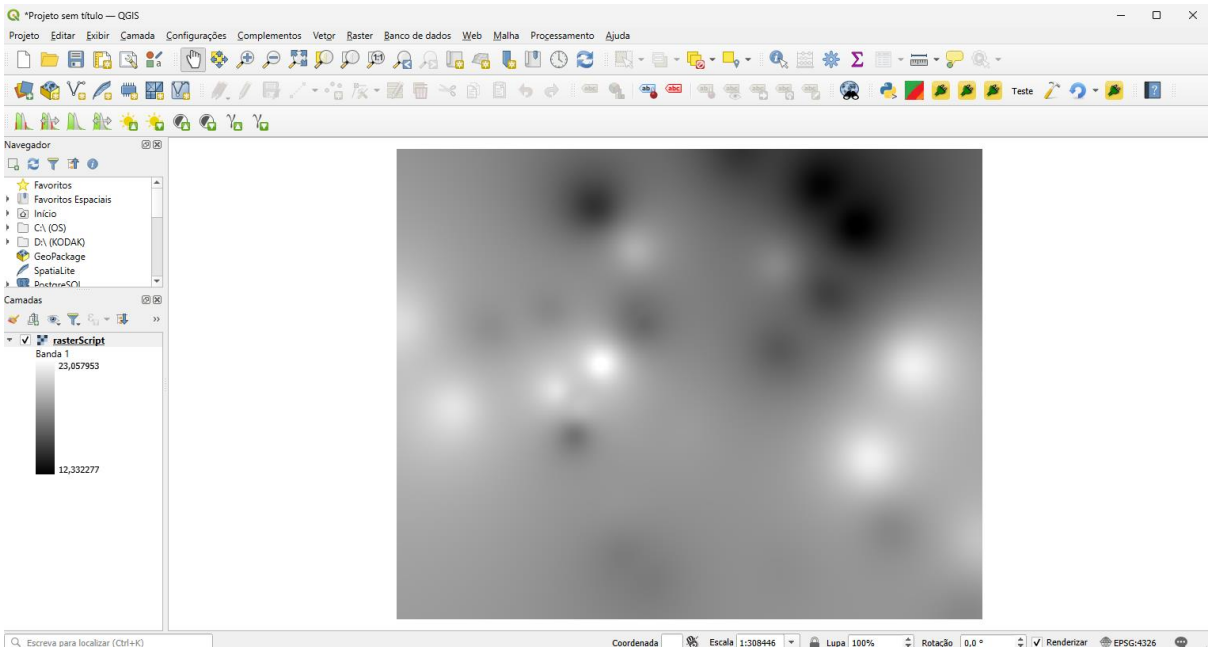
```
from qgis.core import *
from qgis.analysis import *
import os
QgsApplication.setPrefixPath("C:/ProgramFiles/QGIS3.34.1/bin/",
True)
qgs = QgsApplication([], False)
qgs.initQgis()
uri="file:///users/user/desktop/dash/tfinal.csv?type=csv&xField=LONG
ITUDE&yField=LATITUDE&crs=epsg:4326"
camada = QgsVectorLayer(uri, 'Converte', "delimitedtext")
if not camada.isValid():
    print("A camada não carregou apropriadamente!")
else:
    c_data = QgsInterpolator.LayerData()
    c_data.source = camada
    c_data.zCoordInterpolation = False
    c_data.interpolationAttribute = 6
    c_data.sourceType = QgsInterpolator.SourcePoints
    interpolado = QgsIDWInterpolator([c_data])
    interpolado.setDistanceCoefficient(2)
    arquivo = os.path.expanduser('~')+"/desktop/dash/rasterScript.asc"
    rect = camada.extent()
    res = 0.001
    ncol = int( ( rect.xMaximum() - rect.xMinimum() ) / res )
    nrows = int( (rect.yMaximum() - rect.yMinimum() ) / res)
    saida = QgsGridFileWriter(interpolado,arquivo,rect,ncol,nrows)
    saida.writeFile()
```

```
qgs.exitQgis()
```

Abrir o shell OSGeo4W, navegar até a pasta do script **criaRaster.py** e executar usando:

```
python-qgis criaRaster.py
```

Ao abrirmos o raster rasterScript.asc criado pelo script acima no QGIS veremos o seguinte.



Colinha do pyQGS
A seguir temos uma cola (cheat sheet) das classes principais para uma rápida referência. Em azul temos as bibliotecas a serem carregadas no caso de usar em Plugins ou em scripts fora do QGIS.

QgsInterface() iface

Aparência
```
from qgis.PyQt.QtWidgets import QApplication

app = QApplication.instance()
app.setStyleSheet(".QWidget {color:blue;background-color:yellow;}")
```

Você pode carregar o stylesheet de um arquivo com
```
from qgis.PyQt.QtWidgets import QApplication
app = QApplication.instance()
with  open("testdata/file.qss") as qss_file_content:
    app.setStyleSheet(qss_file_content.read()
```

Mudando o ícone e título do programa
```
from qgis.PyQt.QtGui import QIcon
icon = QIcon("/caminho/para/ele/icon.png")
iface.mainWindow().setWindowIcon(icon)
iface.mainWindow().setWindowTitle("ArcGIS só que não....")
```

Barra de Ferramentas
Remove e adiciona barra de ferramenta
```
toolbar = iface.helpToolBar()
parent = toolbar.parentWidget()
parent.removeToolBar(toolbar)
parent.addToolBar(toolbar)
```

Remove ações da barra de ferramenta
```
actions = iface.attributesToolBar().actions()
iface.attributesToolBar().clear()
iface.attributesToolBar().addAction(actions[4])
iface.attributesToolBar().addAction(actions[3])
```

Menus
Remove e adiciona Menu
```
menu = iface.helpMenu()
menubar = menu.parentWidget()
menubar.removeAction(menu.menuAction())
menubar.addAction(menu.menuAction())
```

Canvas (tela do mapa)
Acessando o canvas
```
canvas = iface.mapCanvas()
```

Mudando a cor do canvas

```
from qgis.PyQt.QtCore import Qt
iface.mapCanvas().setCanvasColor(Qt.black)
iface.mapCanvas().refresh()
```

Mudando intervalo de atualização do mapa

```
from qgis.core import QgsSettings
# para 150 milisegundos
QgsSettings().setValue("/qgis/map_update_interval", 150)
```

Camadas

Adicionando uma camada vetor do tipo shapefile

```
camada = iface.addVectorLayer("caminho/aoi.shp", "Camada", "ogr")
if not camada or not camada.isValid():
    print("Falha ao carregar camada!")
```

Adicionando uma camada raster do tipo tif

```
from qgis.core import QgsRasterLayer
arquivoR = "caminho/srtm.tif"
camada = QgsRasterLayer(arquivo, "Raster")
if not camada.isValid():
    print("Falha ao carregar camada!")
```

Obtendo a camada ativa

```
camada = iface.activeLayer()
```

Listando todas as camadas

```
from qgis.core import QgsProject
QgsProject.instance().mapLayers().values()
```

Obtendo o nome das camadas

```
from qgis.core import QgsProject
nomes = [layer.name() for layer in QgsProject.instance().mapLayers().values()]
print("Camadas = {}".format(nomes))
```

Obtendo camada pelo seu nome e fazendo ela ativa

```
from qgis.core import QgsProject
camada = QgsProject.instance().mapLayersByName("NomeCamada")[0]
iface.setActiveLayer(layer)
```

Adicionando um novo elemento

```
from qgis.core import QgsGeometry, QgsPointXY, QgsFeature
pr = camada.dataProvider()
feat = QgsFeature()
feat.setGeometry(QgsGeometry.fromPointXY(QgsPointXY(10,10)))
pr.addFeatures([feat])
```

Obtendo os elementos

```
for f in camada.getFeatures():
    print (f)
```

Obtendo elementos selecionados

```
for f in camada.selectedFeatures():
    print (f)
```

Obtendo o ID dos elementos selecionados

```
selected_ids = camada.selectedFeatureIds()
print(selected_ids)
```

Criando uma camada na memória com os elementos selecionados

```
from qgis.core import QgsFeatureRequest
camadam= layer.materialize(QgsFeatureRequest().setFilterFids(layer.selectedFeatureIds()))
QgsProject.instance().addMapLayer(camadam)
```

Obtendo a geometria da camada

```
# Camada tipo Point
for f in camada.getFeatures():
    geom = f.geometry()
    print ('%f, %f' % (geom.asPoint().y(), geom.asPoint().x()))
```

Transladando uma geometria

```
from qgis.core import QgsFeature, QgsGeometry
f = QgsFeature()
geom = QgsGeometry.fromWkt("POINT(7 45)")
geom.translate(1, 1)
f.setGeometry(geom)
print(f.geometry())
```

Assinalando o Sistema de referência de coordenadas (CRS) de uma camada

```
from qgis.core import QgsProject, QgsCoordinateReferenceSystem
for camada in QgsProject.instance().mapLayers().values():
    camada.setCrs(QgsCoordinateReferenceSystem('EPSG:4326'))
```

Checando o CRS de uma camada

```
from qgis.core import QgsProject
for camada in QgsProject.instance().mapLayers().values():
    crs = camada.crs().authid()
    camada.setName('{} ({})'.format(camada.name(), crs))
```

Uma função para ocultar uma coluna de atributo

```
from qgis.core import QgsEditorWidgetSetup
def fieldVisibility (camada,fname):
    setup = QgsEditorWidgetSetup('Hidden', {})
    for i, column in enumerate(layer.fields()):
        if column.name()==fname:
            camada.setEditorWidgetSetup(idx, setup)
            break
        else:
            continue
```

Criando camada na memória a partir de WKT

```
from qgis.core import QgsVectorLayer,QgsFeature,QgsGeometry,QgsProject

camada = QgsVectorLayer('Polygon?crs=epsg:4326', 'MS', 'memory')
pr = camada.dataProvider()
```

```
f = QgsFeature()
geom = QgsGeometry.fromWkt("POLYGON ((-88.82 34.99,-88.09 34.89,-
88.39 30.34,-89.57 30.18,-89.73 31,-91.63 30.99,-90.87 32.37,-91.23
33.44,-90.93 34.23,-90.30 34.99,-88.82 34.99))")
poly.setGeometry(geom)
pr.addFeatures([f])
camada.updateExtents()
QgsProject.instance().addMapLayers([camada])
```

Lendo as camadas de um GeoPackage de vetores

```
from qgis.core import QgsDataProvider
arquivo = "caminho/seupacote.gpkg"
camada = QgsVectorLayer(arquivo, "teste", "ogr")
subCamadass = camada.dataProvider().subLayers()

for subCamada in subCamadass:
    nome = subCamada.split(QgsDataProvider.SUBLAYER_SEPARATOR)[1]
    uri = "%s|layername=%s" % (arquivo, nome,)
    # Cria a Camada
    subCamadav = QgsVectorLayer(uri, nome, 'ogr')
    # Adiciona a camada ao mapa
    QgsProject.instance().addMapLayer(subVamadav)
```

Adicionando uma camada Tile (Camada XYZ)

```
from qgis.core import QgsRasterLayer, QgsProject
def loadXYZ(url, name):
    rasterLyr = QgsRasterLayer("type=xyz&url=" + url, name, "wms")
    QgsProject.instance().addMapLayer(rasterLyr)

url= 'https://tile.openstreetmap.org/{z}/{x}/{y}.png&zmax=19&zmin=0'
loadXYZ(url, 'OpenStreetMap')
```

Adicionando uma camada de um banco de dados Postgis

```
from qgis.core import QgsVectorLayer, QgsProject, QgsDataSourceUri
uri = QgsDataSourceUri()
uri.setConnection("servidorDNS","5432","dbase","user", "pswrd")
uri.setDataSource("schema", "tablename", "geom")
camada = QgsVectorLayer(uri.uri(False), "Remota", "postgres")
QgsProject.instance().addMapLayer(rasterLyr)
```

Removendo todas as camadas do Projeto

```
QgsProject.instance().removeAllMapLayers()
```

Removendo tudo do projeto

```
QgsProject.instance().clear()
```