
photoshop Documentation

Release 0.14.0

Oct 22, 2020

CONTENTS

1	Installation	3
1.1	From sources	3
2	Examples	5
2.1	Operation Channels	5
2.2	Fit On Screen	5
2.3	Add Start Application Event	5
2.4	Creating A Layer	6
2.5	Save To Psd	6
2.6	Move To End	7
2.7	Cropping	7
2.8	Compare Colors	7
2.9	New Document	8
2.10	Save As Pdf	8
2.11	Session Smart Sharpen	9
2.12	Copy And Paste	10
2.13	Color	11
2.14	Change Color Of Background And Foreground	11
2.15	Import Image As Layer	12
2.16	Active Layer	12
2.17	Toggle Proof Colors	12
2.18	Session New Document	13
2.19	Replace Images	13
2.20	Add Metadata	13
2.21	Enable Generator	13
2.22	Export Layers As Png	14
2.23	Create New Document	15
2.24	Photoshop Session	15
2.25	Current Tool	15
2.26	Add Slate	16
2.27	Load Selection	16
2.28	Save As Tga	17
2.29	Session Hello World	18
2.30	Session Callback	18
2.31	Operation Layer Set	18
2.32	Rotate Layer	19
2.33	Fill Selection	20
2.34	Open Psd	20
2.35	Link Layer	21
2.36	Eval Javascript	21

2.37	Selection Stroke	21
2.38	Create Thumbnail	23
2.39	Apply Filters	24
2.40	Emboss Action	25
2.41	Session Document Duplicate	26
2.42	Set Active Layer	26
2.43	Hello World	27
2.44	Trim	27
2.45	Apply Crystallize Filter Action	27
2.46	Smart Sharpen	28

The API for using COM (Component Object Model) objects interfaces of Photoshop.

INSTALLATION

1.1 From sources

The sources for photoshop python api can be downloaded from the [Github repo](#).

You can either clone the public repository:

```
$ git clone https://github.com/loonghao/photoshop_python_api.git
```

Or download the [tarball](#):

```
$ python setup.py install
```

Or you can install via pip.

```
$ pip install photoshop_python_api
```


EXAMPLES

2.1 Operation Channels

```
"""A examples to show you how to operation active document channels."""

from photoshop import Session

with Session() as ps:
    doc = ps.active_document
    print(len(doc.channels))
    doc.channels.add()
    doc.channels.removeAll()
    channel = doc.channels.getByName("Red")
    print(channel.name)
    channel.remove()
```

2.2 Fit On Screen

```
"""Let the current document Fit on screen."""

from photoshop import Session

with Session() as ps:
    ps.app.runMenuItem(ps.app.charIDToTypeID("FtOn"))
```

2.3 Add Start Application Event

```
"""Add event for Photoshop start application.

In the current example, every time we start photoshop it will
alert "Start Application Event".

Just like you manually in Script> Script Events Manager to enable the event.

"""

from photoshop import Session
from tempfile import mkdtemp
```

(continues on next page)

(continued from previous page)

```
import os

with Session() as ps:
    root = mkdtemp()
    jsx_file = os.path.join(root, "event.jsx")
    with open(jsx_file, "w") as f:
        f.write('alert("Start Application event.")')
    ps.app.notifiers.add(ps.EventID.Notify, jsx_file)
    print("Add event done.")
```

2.4 Creating A Layer

```
"""
Let's get the current document and create a new layer "Background" and fill
↳ it
with red color. In order to use the Fill tool we will first select the entire
layer and then fill it with a color.

"""

from photoshop import Session

with Session() as ps:
    document = ps.active_document
    # Create color object of color red.
    fillColor = ps.SolidColor()
    fillColor.rgb.red = 222
    fillColor.rgb.green = 0
    fillColor.rgb.blue = 0
    # Add a new layer called Background.
    layer = document.artLayers.add()
    layer.name = "Background"
    # Select the entire layer.
    document.selection.selectAll()
    # Fill the selection with color.
    document.selection.fill(fillColor)
    # Deselect.
    document.selection.deselect()
```

2.5 Save To Psd

```
"""Save your current active document as a .psd file."""
from photoshop import Session

with Session() as ps:
    psd_file = "your/psd/save/file/path.psd"
    doc = ps.active_document
    options = ps.PhotoshopSaveOptions()
    layers = doc.artLayers
    doc.saveAs(psd_file, options, True)
```

(continues on next page)

(continued from previous page)

```
ps.alert("Task done!")
ps.echo(doc.activeLayer)
```

2.6 Move To End

```
import photoshop.api as ps

# Get photoshop instance.
app = ps.Application()

# Add new document and set name to "Example for move to End."
active_document = app.documents.add(name="Example for move to End.")

# Add a new layer set.
group_layer = active_document.layerSets.add()
# Add a layer in the group.
layer = group_layer.artLayers.add()
layer.name = "This is a child layer."
# Add a new layer in this active document top.
top_layer = active_document.artLayers.add()
top_layer.name = "This is a top layer."
top_layer.moveToEnd(group_layer)
```

2.7 Cropping

```
"""A cropping example."""

from photoshop import Session

with Session(action="new_document") as ps:
    ps.active_document.crop(bounds=[100, 12, 354, 246],
                             width=1920,
                             height=1080)
```

2.8 Compare Colors

```
"""Check whether the foreground is equal to the background color.

References:
    https://github.com/lohriialo/photoshop-scripting-python/blob/master/
    ↪ CompareColors.py

"""

from photoshop import Session

with Session() as ps:
    if ps.app.foregroundColor.isEqual(ps.app.backgroundColor):
        ps.echo("They're Equal.")
```

(continues on next page)

(continued from previous page)

```
else:
    ps.echo("NOT Equal.")
```

2.9 New Document

```
# Create a new Photoshop document with dimensions 4 inches by 4 inches.
import photoshop.api as ps

# Start up Photoshop application
app = ps.Application()

start_ruler_units = app.preferences.rulerUnits

app.preferences.rulerUnits = ps.Units.Pixels

# Create the document
docRef = app.documents.add(1920, 1080, 72.0, "My New Document")

# Make sure to set the ruler units prior to creating the document.
app.preferences.rulerUnits = start_ruler_units
```

2.10 Save As Pdf

```
"""Save current active document as a PDF file."""
import os
from tempfile import mkdtemp

from photoshop import Session

with Session() as ps:
    option = ps.PDFSaveOptions(jpegQuality=12,
                               layers=True,
                               view=True)
    pdf = os.path.join(mkdtemp(), "test.pdf")
    ps.active_document.saveAs(pdf, option)

with Session() as ps:
    option = ps.PDFSaveOptions()
    option.jpegQuality = 12
    option.layers = True
    option.view = True # opens the saved PDF in Acrobat.
    pdf = os.path.join(mkdtemp(), "test.pdf")
    ps.active_document.saveAs(pdf, option)
```

2.11 Session Smart Sharpen

```

"""This script demonstrates how you can use the action manager to execute the
Emboss filter.

References:
    https://github.com/lohriialo/photoshop-scripting-python/blob/master/
    ↪ SmartSharpen.py

"""

from photoshop import Session

import examples.psd_files as psd # Import from examples.

PSD_FILE = psd.get_psd_files()
file_path = PSD_FILE["layer_comps.psd"]

with Session(file_path, action="open") as ps:
    def SmartSharpen(inAmount, inRadius, inNoise):
        idsmart_sharpen_id = ps.app.stringIDToTypeID(ps.EventID.SmartSharpen)
        desc37 = ps.ActionDescriptor()

        idpresetKind = ps.app.stringIDToTypeID(ps.EventID.PresetKind)
        idpresetKindType = ps.app.stringIDToTypeID(ps.EventID.PresetKindType)
        idpresetKindCustom = ps.app.stringIDToTypeID(
            ps.EventID.PresetKindCustom)
        desc37.putEnumerated(idpresetKind, idpresetKindType,
                             idpresetKindCustom)
        idAmnt = ps.app.charIDToTypeID("Amnt")
        idPrc = ps.app.charIDToTypeID("Rds ")
        desc37.putUnitDouble(idAmnt, idPrc, inAmount)

        idRds = ps.app.charIDToTypeID("Rds ")
        idPx1 = ps.app.charIDToTypeID("#Px1")
        desc37.putUnitDouble(idRds, idPx1, inRadius)

        idnoiseReduction = ps.app.stringIDToTypeID("noiseReduction")
        idPrc = ps.app.charIDToTypeID("#Prc")
        desc37.putUnitDouble(idnoiseReduction, idPrc, inNoise)

        idblur = ps.app.charIDToTypeID("blur")
        idblurType = ps.app.stringIDToTypeID("blurType")
        idGsnB = ps.app.charIDToTypeID("GsnB")
        desc37.putEnumerated(idblur, idblurType, idGsnB)

        ps.app.ExecuteAction(idsmart_sharpen_id, desc37)

    docRef = ps.active_document
    nlayerSets = docRef.layerSets
    nArtLayers = docRef.layerSets.item(nlayerSets.length)
    docRef.activeLayer = nArtLayers.artLayers.item(nArtLayers.artLayers.
    ↪length)

    SmartSharpen(300, 2.0, 20)

```

2.12 Copy And Paste

```
"""
References:
    https://github.com/lohriialo/photoshop-scripting-python/blob/master/
    ↪ CopyAndPaste.py
"""

import photoshop.api as ps

app = ps.Application()

startRulerUnits = app.preferences.rulerUnits

app.preferences.rulerUnits = ps.Units.Inches

doc = app.documents.add(
    7, 5, 72, None, ps.NewDocumentMode.NewRGB, ps.DocumentFill.White
)

# Make sure the active layer is not a text layer, which cannot be copied to
# ↪ the
# ↪ clipboard.
if doc.activeLayer.kind != ps.LayerKind.TextLayer:
    # Select the left half of the document. Selections are always expressed
    # in pixels regardless of the current ruler unit type, so we're computing
    # the selection corner points based on the inch unit width and height
    # of the document
    x2 = (doc.width * doc.resolution) / 2
    y2 = doc.height * doc.resolution

    sel_area = ((0, 0), (x2, 0), (x2, y2), (0, y2))
    doc.selection.select(sel_area, ps.SelectionType.ReplaceSelection, 0,
    ↪ False)

    doc.selection.copy()

    # The new doc is created
    # need to change ruler units to pixels because x2 and y2 are pixel units.
    app.preferences.rulerUnits = ps.Units.Pixels
    pasteDoc = app.documents.add(x2, y2, doc.resolution, "Paste Target")
    pasteDoc.paste()
else:
    print("You cannot copy from a text layer")

if startRulerUnits != app.preferences.rulerUnits:
    app.preferences.rulerUnits = startRulerUnits
```

2.13 Color

```
from photoshop import Session

with Session() as ps:
    doc = ps.active_document
    # Add a solid color.
    textColor = ps.SolidColor()
    textColor.rgb.red = 255.0
    textColor.rgb.green = 197
    textColor.rgb.blue = 255

    # Create empty layer.
    new_text_layer = doc.artLayers.add()
    # Set empty layer type to text layer
    new_text_layer.kind = ps.LayerKind.TextLayer
    # Set current text layer contents to "Hello, World!".
    new_text_layer.textItem.contents = "Hello, World!"
    # Change current text layer position.
    new_text_layer.textItem.position = [160, 167]
    # Change current text layer text size.
    new_text_layer.textItem.size = 36
    # Change current text layer color.
    new_text_layer.textItem.color = textColor
    assert new_text_layer.textItem.color.rgb.red == textColor.rgb.red
```

2.14 Change Color Of Background And Foreground

```
"""Change the color of the background and foreground."""
from photoshop import Session

with Session() as ps:
    foregroundColor = ps.SolidColor()
    foregroundColor.rgb.red = 255
    foregroundColor.rgb.green = 0
    foregroundColor.rgb.blue = 0
    ps.app.foregroundColor = foregroundColor

    backgroundColor = ps.SolidColor()
    backgroundColor.rgb.red = 0
    backgroundColor.rgb.green = 0
    backgroundColor.rgb.blue = 0
    ps.app.backgroundColor = backgroundColor
```

2.15 Import Image As Layer

```
"""Import a image as a artLayer."""

from photoshop import Session

with Session(action="new_document") as ps:
    desc = ps.ActionDescriptor
    desc.putPath(ps.app.charIDToTypeID("null"), "your/image/path")
    event_id = ps.app.charIDToTypeID("Plc ") # `Plc` need one space in here.
    ps.app.executeAction(ps.app.charIDToTypeID("Plc "), desc)
```

2.16 Active Layer

```
# Set the active layer to the last art layer of the active document, or the
# first if the last is already active.

from photoshop import Session

with Session() as ps:
    if len(ps.app.documents) < 1:
        docRef = ps.app.documents.add()
    else:
        docRef = ps.app.activeDocument

    if len(docRef.layers) < 2:
        docRef.artLayers.add()

    ps.echo(docRef.activeLayer.name)
    new_layer = docRef.artLayers.add()
    ps.echo(new_layer.name)
    new_layer.name = "test"
```

2.17 Toggle Proof Colors

```
"""Toggle the proof color.

Like operating in the menu:
**View** > **Proof Colors** (Ctrl + Y)

"""

from photoshop import Session

with Session() as ps:
    ps.app.runMenuItem(ps.app.stringIDToTypeID("toggleProofColors"))
```


2.18 Session New Document

```
"""Action for create new document and print new document name."""
from photoshop import Session

with Session(action="new_document") as ps:
    ps.echo(ps.active_document.name)
```

2.19 Replace Images

```
"""Replace the image of the current active layer with a new image."""

from photoshop import Session

with Session() as ps:
    replace_contents = ps.app.stringIDToTypeID("placedLayerReplaceContents")
    desc = ps.ActionDescriptor
    idnull = ps.app.charIDToTypeID("null")
    desc.putPath(idnull, "your/image/path.jpg")
    ps.app.executeAction(replace_contents, desc)
```

2.20 Add Metadata

```
"""Add metadata to current active document."""

# Import built-in modules
import os

# Import local modules
from photoshop import Session

with Session(action="new_document") as ps:
    doc = ps.active_document
    doc.info.author = os.getenv("USERNAME")
    doc.info.provinceState = "Beijing"
    doc.info.title = "My Demo"
    print("Print all metadata of current active document.")
    ps.echo(doc.info)
```

2.21 Enable Generator

```
"""Enable Generator features."""
from photoshop import Session

with Session() as ps:
    plugin_name = "generator-assets-dummy-menu"
    generatorDesc = ps.ActionDescriptor
```

(continues on next page)

(continued from previous page)

```
generatorDesc.putString(ps.app.stringIDToTypeID("name"), plugin_name)
ps.app.executeAction(ps.app.stringIDToTypeID("generateAssets"),
                    generatorDesc)
```

2.22 Export Layers As Png

```
"""Export every layer as a .png file."""
import os

from photoshop import Session

import examples.psd_files as psd # Import from examples.

PSD_FILE = psd.get_psd_files()

def hide_all_layers(layers):
    for layer in layers:
        layer.visible = False

def main():
    psd_file = PSD_FILE["export_layers_as_png.psd"]
    with Session(psd_file, action="open") as ps:
        doc = ps.active_document
        options = ps.PNGSaveOptions()
        layers = doc.artLayers
        for layer in layers:
            hide_all_layers(layers)
            layer.visible = True
            layer_path = os.path.join(doc.path, layer.name)
            print(layer_path)
            if not os.path.exists(layer_path):
                os.makedirs(layer_path)
            image_path = os.path.join(layer_path, f"{layer.name}.png")
            doc.saveAs(image_path, options, True)
        ps.alert("Task done!")
        ps.echo(doc.activeLayer)

if __name__ == "__main__":
    main()
```

2.23 Create New Document

```
"""Create a new document."""

from photoshop import Session

with Session() as ps:
    ps.app.preferences.rulerUnits = ps.Units.Pixels
    ps.app.documents.add(1920, 1080, name="my_new_document")
```

2.24 Photoshop Session

```
"""Add slate information dynamically."""

import os
from datetime import datetime
from tempfile import mkdtemp

import examples.psd_files as psd # Import from examples.
from photoshop import Session

PSD_FILE = psd.get_psd_files()
file_path = PSD_FILE["slate_template.psd"]

with Session(file_path, action="open", auto_close=True) as ps:
    layer_set = ps.active_document.layerSets.getByName("template")
    data = {
        "project_name": "test_project",
        "datetime": datetime.today().strftime("%Y-%m-%d"),
    }
    for layer in layer_set.layers:
        if layer.kind == ps.LayerKind.TextLayer:
            layer.textItem.contents = data[layer.textItem.contents.strip()]

    jpg_file = os.path.join(mkdtemp("photoshop-python-api"), "slate.jpg")
    ps.active_document.saveAs(jpg_file, ps.JPEGSaveOptions())
    os.startfile(jpg_file)
```

2.25 Current Tool

```
from photoshop import Session

with Session() as ps:
    # Print the current tool.
    ps.echo(ps.app.currentTool)

    # Set current tool to `typeCreateOrEditTool`.
    ps.app.currentTool = "typeCreateOrEditTool"
```

2.26 Add Slate

```
"""Add slate information dynamically.

- Open template.
- Update info.
- Save as jpg.
- Close current document.

"""

import os
from datetime import datetime
from tempfile import mkdtemp

from photoshop import Session

import examples._psd_files as psd # Import from examples.

PSD_FILE = psd.get_psd_files()
slate_template = PSD_FILE["slate_template.psd"]
with Session(slate_template, action="open", auto_close=True) as ps:
    layer_set = ps.active_document.layerSets.getByName("template")

    data = {
        "project name": "test_project",
        "datetime": datetime.today().strftime("%Y-%m-%d"),
    }
    for layer in layer_set.layers:
        if layer.kind == ps.LayerKind.TextLayer:
            layer.textItem.contents = data[layer.textItem.contents.strip()]

    jpg_file = os.path.join(mkdtemp("photoshop-python-api"), "slate.jpg")
    ps.active_document.saveAs(jpg_file, ps.JPEGSaveOptions())
    print(f"Save jpg to {jpg_file}")
    os.startfile(jpg_file)
```

2.27 Load Selection

```
# This script will demonstrate how to load a selection from a saved alpha
# channel.

from photoshop import Session

with Session() as ps:
    doc_ref = ps.app.documents.add(320, 240)
    start_ruler_units = ps.app.preferences.rulerUnits
    if start_ruler_units is not ps.Units.Pixels:
        ps.app.Preferences.RulerUnits = ps.Units.Pixels
    # Save a rectangular selection area offset by 50 pixels from the image
    # border into an alpha channel.
    offset = 50
    selBounds1 = (
        (offset, offset),
```

(continues on next page)

(continued from previous page)

```

        (doc_ref.Width - offset, offset),
        (doc_ref.Width - offset, doc_ref.Height - offset),
        (offset, doc_ref.Height - offset),
    )
    doc_ref.selection.select(selBounds1)
    selAlpha = doc_ref.channels.Add()
    doc_ref.selection.store(selAlpha)

    # Now create a second wider but less tall selection.
    selBounds2 = ((0, 75), (doc_ref.Width, 75), (doc_ref.Width, 150), (0,
→150))
    doc_ref.selection.select(selBounds2)

    # Load the selection from the just saved alpha channel, combining it with
    # the active selection.
    doc_ref.selection.load(selAlpha, ps.SelectionType.ExtendSelection, False)

    # Set ruler back to where it was.
    ps.app.Preferences.RulerUnits = start_ruler_units

```

2.28 Save As Tga

```

import os
from tempfile import mkdtemp

from photoshop import Session

with Session(action="new_document") as ps:
    doc = ps.active_document
    text_color = ps.SolidColor()
    text_color.rgb.green = 255
    text_color.rgb.red = 0
    text_color.rgb.blue = 0
    new_text_layer = doc.artLayers.add()
    new_text_layer.kind = ps.LayerKind.TextLayer
    new_text_layer.textItem.contents = "Hello, World!"
    new_text_layer.textItem.position = [160, 167]
    new_text_layer.textItem.size = 40
    new_text_layer.textItem.color = text_color
    tga_file = os.path.join(mkdtemp("photoshop-python-api"), "test.tga")
    doc.saveAs(tga_file, ps.TargaSaveOptions(), asCopy=True)
    os.startfile(tga_file)

```

2.29 Session Hello World

```
"""Add slate information dynamically."""

import os
from tempfile import mkdtemp

from photoshop import Session

with Session() as adobe:
    doc = adobe.app.documents.add(2000, 2000)
    text_color = adobe.SolidColor()
    text_color.rgb.red = 255
    new_text_layer = doc.artLayers.add()
    new_text_layer.kind = adobe.LayerKind.TextLayer
    new_text_layer.textItem.contents = "Hello, World!"
    new_text_layer.textItem.position = [160, 167]
    new_text_layer.textItem.size = 40
    new_text_layer.textItem.color = text_color
    options = adobe.JPEGSaveOptions(quality=1)
    jpg_file = os.path.join(mkdtemp("photoshop-python-api"), "hello_world.jpg
↪")
    doc.saveAs(jpg_file, options, asCopy=True)
    adobe.app.doJavaScript(f'alert("save to jpg: {jpg_file}")')
```

2.30 Session Callback

```
from photoshop import Session

def do_something(photoshop_api):
    print(photoshop_api.active_document)
    print("Do something.")

with Session(callback=do_something) as ps:
    ps.echo(ps.active_document.name)
    ps.alert(ps.active_document.name)
```

2.31 Operation Layer Set

```
"""A examples to show you how to operation layerSet."""

from photoshop import Session

with Session(action="new_document") as ps:
    docRef = ps.active_document
    # Add a new layerSet.
    new_layer_set = docRef.layerSets.add()
    # Print the layerSet count.
    ps.echo(docRef.layerSets.length)
```

(continues on next page)

(continued from previous page)

```

ps.echo(len(docRef.layerSets))
# Rename the layerSet.
docRef.layerSets[0].name = "New Name"
ps.echo(new_layer_set.name)

# Change the layerSet opacity
new_layer_set.opacity = 90
ps.echo(new_layer_set.opacity)

# Duplicate the layerSet.
duplicate_layer_set = new_layer_set.duplicate()
# Add a new artLayer in current active document.
layer = docRef.artLayers.add()
# Move the artLayer under the duplicate layerSet.
layer.move(duplicate_layer_set, ps.ElementPlacement.PlaceInside)
# Merge the layerSet.
merged_layer = duplicate_layer_set.merge()
ps.echo(merged_layer.name)

# Set visible.
new_layer_set.visible = False

merged_layer.remove()

```

2.32 Rotate Layer

```

"""This scripts demonstrates how to rotate a layer 45 degrees clockwise.

References:
    https://github.com/lohriialo/photoshop-scripting-python/blob/master/
    ↪RotateLayer.py

"""

import photoshop.api as ps

app = ps.Application()

if len(app.documents) > 0:
    print(app.activeDocument.activeLayer.typename)
    if not app.activeDocument.activeLayer.isBackgroundLayer:
        docRef = app.activeDocument
        layerRef = docRef.layers[0]
        layerRef.rotate(45.0)
    else:
        print("Operation cannot be performed on background layer")
else:
    print("You must have at least one open document to run this script!")

```

2.33 Fill Selection

```
# Fill the current selection with an RGB color.

from photoshop import Session

with Session() as ps:
    start_ruler_units = ps.app.Preferences.RulerUnits

    if len(ps.app.documents) < 1:
        if start_ruler_units is not ps.Units.Pixels:
            ps.app.Preferences.RulerUnits = ps.Units.Pixels
        docRef = ps.app.documents.add(
            320, 240, 72, None, ps.NewDocumentMode.NewRGB,
            ps.DocumentFill.White
        )
        docRef.artLayers.add()
        ps.app.preferences.rulerUnits = start_ruler_units

    if not ps.active_document.activeLayer.isBackgroundLayer:
        selRef = ps.active_document.selection
        fillColor = ps.SolidColor()
        fillColor.rgb.red = 225
        fillColor.rgb.green = 0
        fillColor.rgb.blue = 0
        selRef.fill(fillColor, ps.ColorBlendMode.NormalBlendColor, 25, False)
    else:
        ps.echo("Can't perform operation on background layer.")
```

2.34 Open Psd

```
import photoshop.api as ps
from photoshop import Session

# style 1
app = ps.Application()
app.load("your/psd/or/psb/file_path.psd")

# style 2
with Session("your/psd/or/psb/file_path.psd", action="open") as ps:
    ps.echo(ps.active_document.name)
```


2.35 Link Layer

```
import photoshop.api as ps

app = ps.Application()

start_ruler_units = app.preferences.rulerUnits

if len(app.documents) < 1:
    if start_ruler_units is not ps.Units.Pixels:
        app.preferences.rulerUnits = ps.Units.Pixels
    docRef = app.documents.add(
        320, 240, 72, None,
        ps.NewDocumentMode.NewRGB,
        ps.DocumentFill.BackgroundColor,
    )
else:
    docRef = app.activeDocument

layerRef = docRef.artLayers.add()
layerRef2 = docRef.artLayers.add()
layerRef.link(layerRef2)

# Set the ruler back to where it was
app.preferences.rulerUnits = start_ruler_units
```

2.36 Eval Javascript

```
import photoshop.api as ps

app = ps.Application()
jsx = r"""
var doc = app.activeDocument;
var orig_name = doc.name;
alert(orig_name);
"""
app.doJavaScript(jsx)

# Print name of current active document.
print(app.doJavaScript("app.activeDocument.name"))
```

2.37 Selection Stroke

```
"""Create a stroke around the current selection, Set the stroke color and
width of the new stroke.

References:
    https://github.com/lohriialo/photoshop-scripting-python/blob/master/
    ↪ SelectionStroke.py
```

(continues on next page)

(continued from previous page)

```
"""

import photoshop.api as ps

app = ps.Application()

if len(list((i, x) for i, x in enumerate(app.documents, 1))) > 0:
    if not app.activeDocument.activeLayer.isBackgroundLayer:
        psPixels = 1
        start_ruler_units = app.Preferences.RulerUnits
        app.preferences.rulerUnits = ps.Units.Pixels

        selRef = app.activeDocument.selection
        offset = 10
        selBounds = (
            (offset, offset),
            (app.activeDocument.width - offset, offset),
            (app.activeDocument.width - offset,
             app.activeDocument.height - offset),
            (offset, app.activeDocument.height - offset),
        )

        selRef.select(selBounds)
        selRef.selectBorder(5)

        # create text color properties
        strokeColor = ps.SolidColor()

        strokeColor.cmyk.cyan = 58
        strokeColor.cmyk.magenta = 0
        strokeColor.cmyk.yellow = 70
        strokeColor.cmyk.black = 0
        app.displayDialogs = ps.DialogModes.DisplayNoDialogs
        selRef.stroke(
            strokeColor,
            2,
            ps.StrokeLocation.OutsideStroke,
            ps.ColorBlendMode.ColorBlendMode,
            75,
            True,
        )

        # Set ruler units back the way we found it.
        app.preferences.rulerUnits = start_ruler_units
    else:
        print("Operation cannot be performed on background layer")
else:
    print("Create a document with an active selection before running this "
          "script!")
```

2.38 Create Thumbnail

```

"""Create a thumbnail image for currently active document.

You can use the thumbnail image to upload to Shotgun or Ftrack.

"""

# Import built-in modules
import os
from tempfile import mkdtemp

# Import local modules
from photoshop import Session

def create_thumbnail(output_path=None, max_resolution=512):
    """Create a thumbnail image for currently active document.

    Args:
        output_path (str): The absolute output path of the thumbnail image.
            The default is to output to a temporary folder.
        max_resolution (int): The max resolution of the thumbnail. The_
↪default
            is `512`.

    Returns:
        str: The absolute output path of the thumbnail image.

    """
    output_path = output_path or os.path.join(mkdtemp(), "thumb.jpg")

    with Session(auto_close=True) as ps:
        orig_name = ps.active_document.name
        width_str = ps.active_document.width
        height_str = ps.active_document.height
        thumb_name = f"{orig_name}_thumb"

        max_resolution = width_str / max_resolution
        thumb_width = int(width_str / max_resolution)
        thumb_height = int(height_str / max_resolution)

        thumb_doc = ps.active_document.duplicate(thumb_name)
        thumb_doc.resizeImage(thumb_width, thumb_height - 100)
        thumb_doc.saveAs(output_path, ps.JPEGSaveOptions(), asCopy=True)
        thumb_doc.close()
        return output_path

if __name__ == "__main__":
    thumb_file = create_thumbnail()
    print(f"Save thumbnail file to {thumb_file}.")

```

2.39 Apply Filters

```

"""
References:
    https://github.com/lohriialo/photoshop-scripting-python/blob/master/
    ↪ApplyFilters.py

"""
import os

# selections in the open document.
import photoshop.api as ps

import examples._psd_files as psd # Import from examples.

PSD_FILE = psd.get_psd_files()

# Start up Photoshop application
app = ps.Application()

# We don't want any Photoshop dialogs displayed during automated execution
app.displayDialogs = ps.DialogModes.DisplayNoDialogs

psPixels = 1
start_ruler_units = app.preferences.rulerUnits
if start_ruler_units is not psPixels:
    app.preferences.rulerUnits = psPixels

fileName = PSD_FILE["layer_comps.psd"]
docRef = app.open(fileName)
nLayerSets = len(list((i, x) for i, x in enumerate(docRef.layerSets))) - 1
nArtLayers = len(
    list((i, x) for i, x in enumerate(docRef.layerSets[nLayerSets].
    ↪artLayers)),
)

active_layer = docRef.activeLayer = docRef.layerSets[nLayerSets].artLayers[
    nArtLayers]
sel_area = ((0, 212), (300, 212), (300, 300), (0, 300))
docRef.selection.select(sel_area, ps.SelectionType.ReplaceSelection, 20,
    ↪True)
print(f"Current active layer: {active_layer.name}")
active_layer.applyAddNoise(15, ps.NoiseDistribution.GaussianNoise, False)

backColor = ps.SolidColor()
backColor.hsb.hue = 0
backColor.hsb.saturation = 0
backColor.hsb.brightness = 100
app.backgroundColor = backColor

sel_area2 = ((120, 20), (210, 20), (210, 110), (120, 110))
docRef.selection.select(sel_area2, ps.SelectionType.ReplaceSelection, 25,
    False)
active_layer.applyDiffuseGlow(9, 12, 15)
active_layer.applyGlassEffect(
    7, 3, 7, False, ps.TextureType.TinyLensTexture, None,
)

```

(continues on next page)

(continued from previous page)

```
docRef.selection.deselect()

# Set ruler units back the way we found it.
if start_ruler_units is not psPixels:
    app.Preferences.RulerUnits = start_ruler_units
```

2.40 Emboss Action

```
from photoshop import Session

with Session() as ps:
    app = ps.app
    for index, x in enumerate(range(50)):
        # Execute an existing action from action palette.
        idPly = app.charIDToTypeID("Ply ")
        desc8 = ps.ActionDescriptor()
        idnull = app.charIDToTypeID("null")
        ref3 = ps.ActionReference()
        idActn = app.charIDToTypeID("Actn")
        ref3.putName(idActn, "Sepia Toning (layer)")
        idASet = app.charIDToTypeID("ASet")
        ref3.putName(idASet, "Default Actions")
        desc8.putReference(idnull, ref3)
        app.executeAction(idPly, desc8, ps.DialogModes.DisplayNoDialogs)

        # Create solid color fill layer.
        idMk = app.charIDToTypeID("Mk ")
        desc21 = ps.ActionDescriptor()
        idNull = app.charIDToTypeID("null")
        ref12 = ps.ActionReference()
        idContentLayer1 = app.stringIDToTypeID("contentLayer")
        ref12.putClass(idContentLayer1)
        desc21.putReference(idNull, ref12)
        idUsng = app.charIDToTypeID("Usng")
        desc22 = ps.ActionDescriptor()
        idType = app.charIDToTypeID("Type")
        desc23 = ps.ActionDescriptor()
        idClr = app.charIDToTypeID("Clr ")
        desc24 = ps.ActionDescriptor()
        idRd = app.charIDToTypeID("Rd ")
        desc24.putDouble(idRd, index)
        idGrn = app.charIDToTypeID("Grn ")
        desc24.putDouble(idGrn, index)
        idBl = app.charIDToTypeID("Bl ")
        desc24.putDouble(idBl, index)
        idRGBC = app.charIDToTypeID("RGBC")
        desc23.putObject(idClr, idRGBC, desc24)
        idSolidColorLayer = app.stringIDToTypeID("solidColorLayer")
        desc22.putObject(idType, idSolidColorLayer, desc23)
        idContentLayer2 = app.stringIDToTypeID("contentLayer")
        desc21.putObject(idUsng, idContentLayer2, desc22)
        app.executeAction(idMk, desc21, ps.DialogModes.DisplayNoDialogs)

        # Select mask.
```

(continues on next page)

(continued from previous page)

```
idSlct = app.charIDToTypeID("slct")
desc38 = ps.ActionDescriptor()
idNull1 = app.charIDToTypeID("null")
ref20 = ps.ActionReference()
idChnl1 = app.charIDToTypeID("Chnl")
idChnl2 = app.charIDToTypeID("Chnl")
idMsk = app.charIDToTypeID("Msk ")
ref20.putEnumerated(idChnl1, idChnl2, idMsk)
desc38.putReference(idNull1, ref20)
idMkVs = app.charIDToTypeID("MkVs")
desc38.putBoolean(idMkVs, False)
app.executeAction(idSlct, desc38, ps.DialogModes.DisplayNoDialogs)

app.activeDocument.activeLayer.invert()
```

2.41 Session Document Duplicate

```
"""Action for duplicate current active document."""
from photoshop import Session

with Session(action="document_duplicate") as ps:
    ps.echo(ps.active_document.name)
```

2.42 Set Active Layer

```
"""
References:
    https://github.com/lohriialo/photoshop-scripting-python/blob/master/
    ↪ActiveLayer.py

"""
import photoshop.api as ps

app = ps.Application()

if app.documents.length < 1:
    docRef = app.documents.add()
else:
    docRef = app.activeDocument

if docRef.layers.length < 2:
    docRef.artLayers.add()

activeLayerName = docRef.activeLayer.name
if docRef.activeLayer.name != docRef.layers.item(docRef.layers.length).name:
    docRef.activeLayer = docRef.layers.item(docRef.layers.length)
else:
    docRef.activeLayer = docRef.layers.item(1)
```

2.43 Hello World

```
import os
from tempfile import mkdtemp

import photoshop.api as ps

def hello_world():
    app = ps.Application()
    doc = app.documents.add()
    text_color = ps.SolidColor()
    text_color.rgb.green = 255
    new_text_layer = doc.artLayers.add()
    new_text_layer.kind = ps.LayerKind.TextLayer
    new_text_layer.textItem.contents = "Hello, World!"
    new_text_layer.textItem.position = [160, 167]
    new_text_layer.textItem.size = 40
    new_text_layer.textItem.color = text_color
    options = ps.JPEGSaveOptions(quality=5)
    jpg_file = os.path.join(mkdtemp("photoshop-python-api"), "hello_world.jpg")
    doc.saveAs(jpg_file, options, asCopy=True)
    os.startfile(jpg_file)

if __name__ == "__main__":
    hello_world()
```

2.44 Trim

```
"""A trim example."""

import examples.psd_files as psd # Import from examples.
from photoshop import Session

PSD_FILE = psd.get_psd_files()
example_file = PSD_FILE["trim.psd"]
with Session(example_file, action="open") as ps:
    ps.active_document.trim(ps.TrimType.TopLeftPixel,
                           True, True, True, True)
```

2.45 Apply Crystallize Filter Action

```
""" This script demonstrates how you can use the action manager
to execute the Crystallize filter.
In order to find all the IDs, see https://helpx.adobe.com/photoshop/kb/downloadable-plugins-and-content.html#ScriptingListenerplugin
This blog here explains what a script listener is http://blogs.adobe.com/crawlspace/2006/05/installing\_and\_1.html
```

(continues on next page)

(continued from previous page)

```

References:
    https://github.com/lohriialo/photoshop-scripting-python/blob/master/
    ↪ApplyCrystallizeFilterAction.py

"""

from photoshop import Session

import examples._psd_files as psd # Import from examples.

PSD_FILE = psd.get_psd_files()

with Session(PSD_FILE["layer_comps.psd"], "open") as ps:
    active_document = ps.active_document
    nLayerSets = active_document.layerSets
    print(f"The total amount of current layerSet (Group) is "
          f"{len(nLayerSets)}.")
    nArtLayers = active_document.layerSets.item(len(nLayerSets)).artLayers

    # get the last layer in LayerSets
    active_document.activeLayer = active_document.layerSets.item(
        len(nLayerSets)).artLayers.item(len(nArtLayers))

    def applyCrystallize(cellSize):
        cellSizeID = ps.app.CharIDToTypeID("ClSz")
        eventCrystallizeID = ps.app.CharIDToTypeID("Crst")

        filterDescriptor = ps.ActionDescriptor
        filterDescriptor.putInteger(cellSizeID, cellSize)

        ps.app.executeAction(eventCrystallizeID, filterDescriptor)

    applyCrystallize(25)
    print("Apply crystallize done.")

```

2.46 Smart Sharpen

```

"""This script demonstrates how you can use the action manager to execute the
Emboss filter.

References:
    https://github.com/lohriialo/photoshop-scripting-python/blob/master/
    ↪SmartSharpen.py

"""

import photoshop.api as ps

import examples._psd_files as psd # Import from examples.

app = ps.Application()

PSD_FILE = psd.get_psd_files()

```

(continues on next page)

(continued from previous page)

```

file_path = PSD_FILE["layer_comps.psd"]
docRef = app.open(file_path)

nlayerSets = docRef.layerSets
nArtLayers = docRef.layerSets.item(nlayerSets.length)
docRef.activeLayer = nArtLayers.artLayers.item(nArtLayers.artLayers.length)

def SmartSharpen(inAmount, inRadius, inNoise):
    idsmart_sharpen_id = app.stringIDToTypeID(ps.EventID.SmartSharpen)
    desc37 = ps.ActionDescriptor()

    idpresetKind = app.stringIDToTypeID(ps.EventID.PresetKind)
    idpresetKindType = app.stringIDToTypeID(ps.EventID.PresetKindType)
    idpresetKindCustom = app.stringIDToTypeID(ps.EventID.PresetKindCustom)
    desc37.putEnumerated(idpresetKind, idpresetKindType, idpresetKindCustom)

    idAmnt = app.charIDToTypeID("Amnt")
    idPrc = app.charIDToTypeID("Rds ")
    desc37.putUnitDouble(idAmnt, idPrc, inAmount)

    idRds = app.charIDToTypeID("Rds ")
    idPxl = app.charIDToTypeID("#Pxl")
    desc37.putUnitDouble(idRds, idPxl, inRadius)

    idnoiseReduction = app.stringIDToTypeID("noiseReduction")
    idPrc = app.charIDToTypeID("#Prc")
    desc37.putUnitDouble(idnoiseReduction, idPrc, inNoise)

    idblur = app.charIDToTypeID("blur")
    idblurType = app.stringIDToTypeID("blurType")
    idGsnB = app.charIDToTypeID("GsnB")
    desc37.putEnumerated(idblur, idblurType, idGsnB)

    app.ExecuteAction(idsmart_sharpen_id, desc37)

SmartSharpen(300, 2.0, 20)

```