The following are *50* code examples for showing how to use *winreg.HKEY\_LOCAL\_MACHINE()*. They are extracted from open source Python projects. You can vote up the examples you like or vote down the exmaples you don't like. You can also save this page to your account.

**+** Save to library

**Example 1**

|  |  |  |
| --- | --- | --- |
| Project: *CodeReader*   Author: *jasonrbr*   File: *iebutton.py*    [(license)](https://github.com/jasonrbr/CodeReader)[*View Source Project*](https://github.com/jasonrbr/CodeReader/tree/master/site-packages/win32com/demos/iebutton.py) | 9 votes | vote downvote up |

def register(classobj):

import winreg

subKeyCLSID = "SOFTWARE\\Microsoft\\Internet Explorer\\Extensions\\%38s" % classobj.\_reg\_clsid\_

try:

hKey = winreg.CreateKey( winreg.HKEY\_LOCAL\_MACHINE, subKeyCLSID )

subKey = winreg.SetValueEx( hKey, "ButtonText", 0, winreg.REG\_SZ, classobj.\_button\_text\_ )

winreg.SetValueEx( hKey, "ClsidExtension", 0, winreg.REG\_SZ, classobj.\_reg\_clsid\_ ) # reg value for calling COM object

winreg.SetValueEx( hKey, "CLSID", 0, winreg.REG\_SZ, "{1FBA04EE-3024-11D2-8F1F-0000F87ABD16}" ) # CLSID for button that sends command to COM object

winreg.SetValueEx( hKey, "Default Visible", 0, winreg.REG\_SZ, "Yes" )

winreg.SetValueEx( hKey, "ToolTip", 0, winreg.REG\_SZ, classobj.\_tool\_tip\_ )

winreg.SetValueEx( hKey, "Icon", 0, winreg.REG\_SZ, classobj.\_icon\_)

winreg.SetValueEx( hKey, "HotIcon", 0, winreg.REG\_SZ, classobj.\_hot\_icon\_)

except WindowsError:

print("Couldn't set standard toolbar reg keys.")

else:

print("Set standard toolbar reg keys.")

**Example 2**

|  |  |  |
| --- | --- | --- |
| Project: *get-youtube-subtitle-url-node*   Author: *joegesualdo*   File: *buildserver.py*    [(license)](https://github.com/joegesualdo/get-youtube-subtitle-url-node)[*View Source Project*](https://github.com/joegesualdo/get-youtube-subtitle-url-node/tree/master/youtube-dl/devscripts/buildserver.py) | 9 votes | vote downvote up |

def \_\_init\_\_(self, \*\*kwargs):

python\_version = kwargs.pop('python', '3.4')

python\_path = None

for node in ('Wow6432Node\\', ''):

try:

key = compat\_winreg.OpenKey(

compat\_winreg.HKEY\_LOCAL\_MACHINE,

r'SOFTWARE\%sPython\PythonCore\%s\InstallPath' % (node, python\_version))

try:

python\_path, \_ = compat\_winreg.QueryValueEx(key, '')

finally:

compat\_winreg.CloseKey(key)

break

except Exception:

pass

if not python\_path:

raise BuildError('No such Python version: %s' % python\_version)

self.pythonPath = python\_path

super(PythonBuilder, self).\_\_init\_\_(\*\*kwargs)

**Example 3**

|  |  |  |
| --- | --- | --- |
| Project: *CodeReader*   Author: *jasonrbr*   File: *iebutton.py*    [(license)](https://github.com/jasonrbr/CodeReader)[*View Source Project*](https://github.com/jasonrbr/CodeReader/tree/master/site-packages/win32com/demos/iebutton.py) | 7 votes | vote downvote up |

def unregister(classobj):

import winreg

subKeyCLSID = "SOFTWARE\\Microsoft\\Internet Explorer\\Extensions\\%38s" % classobj.\_reg\_clsid\_

try:

hKey = winreg.CreateKey( winreg.HKEY\_LOCAL\_MACHINE, subKeyCLSID )

subKey = winreg.DeleteValue( hKey, "ButtonText" )

winreg.DeleteValue( hKey, "ClsidExtension" ) # for calling COM object

winreg.DeleteValue( hKey, "CLSID" )

winreg.DeleteValue( hKey, "Default Visible" )

winreg.DeleteValue( hKey, "ToolTip" )

winreg.DeleteValue( hKey, "Icon" )

winreg.DeleteValue( hKey, "HotIcon" )

winreg.DeleteKey( winreg.HKEY\_LOCAL\_MACHINE, subKeyCLSID )

except WindowsError:

print("Couldn't delete Standard toolbar regkey.")

else:

print("Deleted Standard toolbar regkey.")

#

# test implementation

#

**Example 4**

|  |  |  |
| --- | --- | --- |
| Project: *optimalvibes*   Author: *littlemika*   File: *buildserver.py*    [(license)](https://github.com/littlemika/optimalvibes)[*View Source Project*](https://github.com/littlemika/optimalvibes/tree/master/server/packages/youtube-dl/devscripts/buildserver.py) | 6 votes | vote downvote up |

def \_\_init\_\_(self, \*\*kwargs):

python\_version = kwargs.pop('python', '3.4')

python\_path = None

for node in ('Wow6432Node\\', ''):

try:

key = compat\_winreg.OpenKey(

compat\_winreg.HKEY\_LOCAL\_MACHINE,

r'SOFTWARE\%sPython\PythonCore\%s\InstallPath' % (node, python\_version))

try:

python\_path, \_ = compat\_winreg.QueryValueEx(key, '')

finally:

compat\_winreg.CloseKey(key)

break

except Exception:

pass

if not python\_path:

raise BuildError('No such Python version: %s' % python\_version)

self.pythonPath = python\_path

super(PythonBuilder, self).\_\_init\_\_(\*\*kwargs)

**Example 5**

|  |  |  |
| --- | --- | --- |
| Project: *mbuild*   Author: *intelxed*   File: *msvs.py*    [(license)](https://github.com/intelxed/mbuild)[*View Source Project*](https://github.com/intelxed/mbuild/tree/master/mbuild/msvs.py) | 6 votes | vote downvote up |

def \_find\_msvc\_in\_registry(env,version):

if \_is\_py2:

import \_winreg as winreg

else:

import winreg

vs\_ver = str(version) + '.0'

vs\_key = 'SOFTWARE\\Microsoft\\VisualStudio\\' + vs\_ver + '\\Setup\\VS'

vc\_key = 'SOFTWARE\\Microsoft\\VisualStudio\\' + vs\_ver + '\\Setup\\VC'

vs\_dir = \_read\_registry(winreg.HKEY\_LOCAL\_MACHINE, vs\_key, 'ProductDir')

vc\_dir = \_read\_registry(winreg.HKEY\_LOCAL\_MACHINE, vc\_key, 'ProductDir')

# On a 64-bit host, look for a 32-bit installation

if (not vs\_dir or not vc\_dir):

vs\_key = 'SOFTWARE\\Wow6432Node\\Microsoft\\VisualStudio\\' + \

vs\_ver + '\\Setup\\VS'

vc\_key = 'SOFTWARE\\Wow6432Node\\Microsoft\\VisualStudio\\' + \

vs\_ver + '\\Setup\\VC'

vs\_dir = \_read\_registry(winreg.HKEY\_LOCAL\_MACHINE,

vs\_key, 'ProductDir')

vc\_dir = \_read\_registry(winreg.HKEY\_LOCAL\_MACHINE,

vc\_key, 'ProductDir')

return (vs\_dir,vc\_dir)

**Example 6**

|  |  |  |
| --- | --- | --- |
| Project: *Packages*   Author: *Keypirinha*   File: *controlpanel.py*    [(license)](https://github.com/Keypirinha/Packages)[*View Source Project*](https://github.com/Keypirinha/Packages/tree/master/ControlPanel/controlpanel.py) | 6 votes | vote downvote up |

def \_list\_items(self):

cpitems = {}

try:

reg\_ns = winreg.OpenKey(

winreg.HKEY\_LOCAL\_MACHINE,

"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Explorer\\ControlPanel\\NameSpace")

reg\_classes = winreg.OpenKey(winreg.HKEY\_CLASSES\_ROOT, "CLSID")

panel\_idx = 0

while 1:

clsid = winreg.EnumKey(reg\_ns, panel\_idx)

panel\_idx += 1

try:

reg\_clsid = winreg.OpenKey(reg\_classes, clsid)

cpitem\_info = self.\_list\_item\_info(clsid, reg\_clsid)

if cpitem\_info:

cpitems[cpitem\_info['clsid']] = cpitem\_info

except OSError:

pass

except OSError:

pass

return cpitems

**Example 7**

|  |  |  |
| --- | --- | --- |
| Project: *lib9*   Author: *Jumpscale*   File: *Windows.py*    [(license)](https://github.com/Jumpscale/lib9)[*View Source Project*](https://github.com/Jumpscale/lib9/tree/master/JumpScale9Lib/sal/windows/Windows.py) | 6 votes | vote downvote up |

def isVistaUACEnabled(self):

"""

Return boolean indicating whether this is a Windows Vista system with

User Account Control enabled.

Warning: If modifies the UAC setting but has not yet rebooted,

this method will return the wrong result.

"""

if self.\_isVistaUACEnabled is not None:

return self.\_isVistaUACEnabled

if self.getWindowsVersion() != self.VERSION\_VISTA:

return False

hkey = reg.HKEY\_LOCAL\_MACHINE

key = 'Software\\Microsoft\\Windows\\CurrentVersion\\Policies\\System'

value = 'EnableLUA'

if not self.registryHasValue(hkey, key, value):

self.\_isVistaUACEnabled = False

elif self.getValueFromRegKey(hkey, key, value) == 0:

self.\_isVistaUACEnabled = False

else:

self.\_isVistaUACEnabled = True

return self.\_isVistaUACEnabled

**Example 8**

|  |  |  |
| --- | --- | --- |
| Project: *lib9*   Author: *Jumpscale*   File: *Windows.py*    [(license)](https://github.com/Jumpscale/lib9)[*View Source Project*](https://github.com/Jumpscale/lib9/tree/master/JumpScale9Lib/sal/windows/Windows.py) | 6 votes | vote downvote up |

def enumRegKeyValues(self, key):

"""List all values of a specified key in the windows registry

@param key: The registry key to check. The key should include the section. Eg. "HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion"

@type key: string

@return: An array of tupples containing the name of each value, the data of the value and it's type

@rtype: tupple(string, WinRegValueType)

"""

hkey, key = self.\_getHiveAndKey(key)

aReg = reg.ConnectRegistry(None, hkey)

aKey = reg.OpenKey(aReg, key)

result = []

index = 0

# The function EnumValue() retrieves the name of one subkey each time it is called.

# It is typically called repeatedly, until an EnvironmentError exception

# is raised, indicating no more values.

while True:

try:

valueName, valueData, valueType = reg.EnumValue(aKey, index)

result.append((valueName, valueData, WinRegValueType.findByIntegerValue(valueType)))

index += 1

except EnvironmentError:

return result

**Example 9**

|  |  |  |
| --- | --- | --- |
| Project: *lib9*   Author: *Jumpscale*   File: *Windows.py*    [(license)](https://github.com/Jumpscale/lib9)[*View Source Project*](https://github.com/Jumpscale/lib9/tree/master/JumpScale9Lib/sal/windows/Windows.py) | 6 votes | vote downvote up |

def enumRegKeySubkeys(self, key):

"""List all sub-keys of a specified key in the windows registry

@param key: The registry key to check. The key should include the section. Eg. "HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion"

@type key: string

"""

hkey, key = self.\_getHiveAndKey(key)

aReg = reg.ConnectRegistry(None, hkey)

aKey = reg.OpenKey(aReg, key)

result = []

index = 0

# The function EnumKey() retrieves the name of one subkey each time it is called.

# It is typically called repeatedly, until an EnvironmentError exception

# is raised, indicating no more values.

while True:

try:

subkey = reg.EnumKey(aKey, index)

result.append(subkey)

index += 1

except EnvironmentError:

return result

**Example 10**

|  |  |  |
| --- | --- | --- |
| Project: *lib9*   Author: *Jumpscale*   File: *Windows.py*    [(license)](https://github.com/Jumpscale/lib9)[*View Source Project*](https://github.com/Jumpscale/lib9/tree/master/JumpScale9Lib/sal/windows/Windows.py) | 6 votes | vote downvote up |

def setValueFromRegKey(self, key, valueName, valueData, valueType):

"""Sets a value in a key

@param key: The registry key that holds the value to set. If the key does not exist, it will be created. The key should include the section. Eg. "HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion"

@type key: string

@param valueName: The name of the value to set

@type valueName: string

@param valueData: The data to assign to the value

@type valueData: string

@param valueType: The type of the value

@type valueType: WinRegValueType

"""

hkey, key = self.\_getHiveAndKey(key)

aReg = reg.ConnectRegistry(None, hkey)

aKey = reg.CreateKey(aReg, key)

reg.SetValueEx(aKey, valueName, 0, valueType.type, valueData)

**Example 11**

|  |  |  |
| --- | --- | --- |
| Project: *sublime-text-3-packages*   Author: *nickjj*   File: *img.py*    [(MIT License)](https://github.com/nickjj/sublime-text-3-packages)[*View Source Project*](https://github.com/nickjj/sublime-text-3-packages/tree/master/Packages/pygments/all/pygments/formatters/img.py) | 5 votes | vote downvote up |

def \_create\_win(self):

try:

key = \_winreg.OpenKey(

\_winreg.HKEY\_LOCAL\_MACHINE,

r'Software\Microsoft\Windows NT\CurrentVersion\Fonts')

except EnvironmentError:

try:

key = \_winreg.OpenKey(

\_winreg.HKEY\_LOCAL\_MACHINE,

r'Software\Microsoft\Windows\CurrentVersion\Fonts')

except EnvironmentError:

raise FontNotFound('Can\'t open Windows font registry key')

try:

path = self.\_lookup\_win(key, self.font\_name, STYLES['NORMAL'], True)

self.fonts['NORMAL'] = ImageFont.truetype(path, self.font\_size)

for style in ('ITALIC', 'BOLD', 'BOLDITALIC'):

path = self.\_lookup\_win(key, self.font\_name, STYLES[style])

if path:

self.fonts[style] = ImageFont.truetype(path, self.font\_size)

else:

if style == 'BOLDITALIC':

self.fonts[style] = self.fonts['BOLD']

else:

self.fonts[style] = self.fonts['NORMAL']

finally:

\_winreg.CloseKey(key)

**Example 12**

|  |  |  |
| --- | --- | --- |
| Project: *devsecops-example-helloworld*   Author: *boozallen*   File: *firefox\_binary.py*    [(license)](https://github.com/boozallen/devsecops-example-helloworld)[*View Source Project*](https://github.com/boozallen/devsecops-example-helloworld/tree/master/webapp/src/test/resources/lib/python2.6/site-packages/selenium/webdriver/firefox/firefox_binary.py) | 5 votes | vote downvote up |

def \_find\_exe\_in\_registry(self):

try:

from \_winreg import OpenKey, QueryValue, HKEY\_LOCAL\_MACHINE, HKEY\_CURRENT\_USER

except ImportError:

from winreg import OpenKey, QueryValue, HKEY\_LOCAL\_MACHINE, HKEY\_CURRENT\_USER

import shlex

keys = (

r"SOFTWARE\Classes\FirefoxHTML\shell\open\command",

r"SOFTWARE\Classes\Applications\firefox.exe\shell\open\command"

)

command = ""

for path in keys:

try:

key = OpenKey(HKEY\_LOCAL\_MACHINE, path)

command = QueryValue(key, "")

break

except OSError:

try:

key = OpenKey(HKEY\_CURRENT\_USER, path)

command = QueryValue(key, "")

break

except OSError:

pass

else:

return ""

if not command:

return ""

return shlex.split(command)[0]

**Example 13**

|  |  |  |
| --- | --- | --- |
| Project: *flasky*   Author: *RoseOu*   File: *firefox\_binary.py*    [(license)](https://github.com/RoseOu/flasky)[*View Source Project*](https://github.com/RoseOu/flasky/tree/master/venv/lib/python2.7/site-packages/selenium/webdriver/firefox/firefox_binary.py) | 5 votes | vote downvote up |

def \_find\_exe\_in\_registry(self):

try:

from \_winreg import OpenKey, QueryValue, HKEY\_LOCAL\_MACHINE, HKEY\_CURRENT\_USER

except ImportError:

from winreg import OpenKey, QueryValue, HKEY\_LOCAL\_MACHINE, HKEY\_CURRENT\_USER

import shlex

keys = (

r"SOFTWARE\Classes\FirefoxHTML\shell\open\command",

r"SOFTWARE\Classes\Applications\firefox.exe\shell\open\command"

)

command = ""

for path in keys:

try:

key = OpenKey(HKEY\_LOCAL\_MACHINE, path)

command = QueryValue(key, "")

break

except OSError:

try:

key = OpenKey(HKEY\_CURRENT\_USER, path)

command = QueryValue(key, "")

break

except OSError:

pass

else:

return ""

if not command:

return ""

return shlex.split(command)[0]

**Example 14**

|  |  |  |
| --- | --- | --- |
| Project: *macos-st-packages*   Author: *zce*   File: *img.py*    [(license)](https://github.com/zce/macos-st-packages)[*View Source Project*](https://github.com/zce/macos-st-packages/tree/master/Packages/pygments/all/pygments/formatters/img.py) | 5 votes | vote downvote up |

def \_create\_win(self):

try:

key = \_winreg.OpenKey(

\_winreg.HKEY\_LOCAL\_MACHINE,

r'Software\Microsoft\Windows NT\CurrentVersion\Fonts')

except EnvironmentError:

try:

key = \_winreg.OpenKey(

\_winreg.HKEY\_LOCAL\_MACHINE,

r'Software\Microsoft\Windows\CurrentVersion\Fonts')

except EnvironmentError:

raise FontNotFound('Can\'t open Windows font registry key')

try:

path = self.\_lookup\_win(key, self.font\_name, STYLES['NORMAL'], True)

self.fonts['NORMAL'] = ImageFont.truetype(path, self.font\_size)

for style in ('ITALIC', 'BOLD', 'BOLDITALIC'):

path = self.\_lookup\_win(key, self.font\_name, STYLES[style])

if path:

self.fonts[style] = ImageFont.truetype(path, self.font\_size)

else:

if style == 'BOLDITALIC':

self.fonts[style] = self.fonts['BOLD']

else:

self.fonts[style] = self.fonts['NORMAL']

finally:

\_winreg.CloseKey(key)

**Example 15**

|  |  |  |
| --- | --- | --- |
| Project: *pyVirtualize*   Author: *rocky1109*   File: *klasses.py*    [(license)](https://github.com/rocky1109/pyVirtualize)[*View Source Project*](https://github.com/rocky1109/pyVirtualize/tree/master/src/pyVirtualize/utils/klasses.py) | 5 votes | vote downvote up |

def \_\_init\_\_(self, scope):

self.scope = scope

if scope == 'user':

self.root = winreg.HKEY\_CURRENT\_USER

self.subkey = 'Environment'

else:

self.root = winreg.HKEY\_LOCAL\_MACHINE

self.subkey = r'SYSTEM\CurrentControlSet\Control\Session Manager\Environment'

**Example 16**

|  |  |  |
| --- | --- | --- |
| Project: *nimp*   Author: *dontnod*   File: *build.py*    [(license)](https://github.com/dontnod/nimp)[*View Source Project*](https://github.com/dontnod/nimp/tree/master/nimp/build.py) | 5 votes | vote downvote up |

def \_find\_devenv\_path(vs\_version):

devenv\_path = None

# First try the registry, because the environment variable is unreliable

# (case of Visual Studio installed on a different drive; it still sets

# the envvar to point to C:\Program Files even if devenv.com is on D:\)

#pylint: disable=import-error

from winreg import OpenKey, QueryValue, HKEY\_LOCAL\_MACHINE

key\_path = 'SOFTWARE\\Classes\\VisualStudio.accessor.' + vs\_version + '.0\\shell\\Open'

try:

with OpenKey(HKEY\_LOCAL\_MACHINE, key\_path) as key:

cmdline = QueryValue(key, 'Command')

if cmdline[:1] == '"':

cmdline = cmdline.split('"')[1]

elif ' ' in cmdline:

cmdline = cmdline.split(' ')[0]

devenv\_path = cmdline.replace('devenv.exe', 'devenv.com')

#pylint: disable=broad-except

except Exception:

pass

# If the registry key is unhelpful, try the environment variable

if not devenv\_path:

vstools\_path = os.getenv('VS' + vs\_version + '0COMNTOOLS')

if vstools\_path is not None:

# Sanitize this because os.path.join sometimes gets confused

if vstools\_path[-1] in [ '/', '\\' ]:

vstools\_path = vstools\_path[:-1]

devenv\_path = os.path.join(vstools\_path, '../../Common7/IDE/devenv.com')

if not devenv\_path or not os.path.exists(devenv\_path):

return None

logging.info("Found Visual Studio at %s", devenv\_path)

return devenv\_path

**Example 17**

|  |  |  |
| --- | --- | --- |
| Project: *leetcode*   Author: *thomasyimgit*   File: *img.py*    [(license)](https://github.com/thomasyimgit/leetcode)[*View Source Project*](https://github.com/thomasyimgit/leetcode/tree/master/env/lib/python3.6/site-packages/pygments/formatters/img.py) | 5 votes | vote downvote up |

def \_create\_win(self):

try:

key = \_winreg.OpenKey(

\_winreg.HKEY\_LOCAL\_MACHINE,

r'Software\Microsoft\Windows NT\CurrentVersion\Fonts')

except EnvironmentError:

try:

key = \_winreg.OpenKey(

\_winreg.HKEY\_LOCAL\_MACHINE,

r'Software\Microsoft\Windows\CurrentVersion\Fonts')

except EnvironmentError:

raise FontNotFound('Can\'t open Windows font registry key')

try:

path = self.\_lookup\_win(key, self.font\_name, STYLES['NORMAL'], True)

self.fonts['NORMAL'] = ImageFont.truetype(path, self.font\_size)

for style in ('ITALIC', 'BOLD', 'BOLDITALIC'):

path = self.\_lookup\_win(key, self.font\_name, STYLES[style])

if path:

self.fonts[style] = ImageFont.truetype(path, self.font\_size)

else:

if style == 'BOLDITALIC':

self.fonts[style] = self.fonts['BOLD']

else:

self.fonts[style] = self.fonts['NORMAL']

finally:

\_winreg.CloseKey(key)

**Example 18**

|  |  |  |
| --- | --- | --- |
| Project: *leetcode*   Author: *thomasyimgit*   File: *firefox\_binary.py*    [(license)](https://github.com/thomasyimgit/leetcode)[*View Source Project*](https://github.com/thomasyimgit/leetcode/tree/master/env/lib/python3.6/site-packages/selenium/webdriver/firefox/firefox_binary.py) | 5 votes | vote downvote up |

def \_find\_exe\_in\_registry(self):

try:

from \_winreg import OpenKey, QueryValue, HKEY\_LOCAL\_MACHINE, HKEY\_CURRENT\_USER

except ImportError:

from winreg import OpenKey, QueryValue, HKEY\_LOCAL\_MACHINE, HKEY\_CURRENT\_USER

import shlex

keys = (r"SOFTWARE\Classes\FirefoxHTML\shell\open\command",

r"SOFTWARE\Classes\Applications\firefox.exe\shell\open\command")

command = ""

for path in keys:

try:

key = OpenKey(HKEY\_LOCAL\_MACHINE, path)

command = QueryValue(key, "")

break

except OSError:

try:

key = OpenKey(HKEY\_CURRENT\_USER, path)

command = QueryValue(key, "")

break

except OSError:

pass

else:

return ""

if not command:

return ""

return shlex.split(command)[0]

**Example 19**

|  |  |  |
| --- | --- | --- |
| Project: *constructor*   Author: *conda*   File: *\_system\_path.py*    [(license)](https://github.com/conda/constructor)[*View Source Project*](https://github.com/conda/constructor/tree/master/constructor/nsis/_system_path.py) | 5 votes | vote downvote up |

def get\_previous\_install\_prefixes(pyversion, arch, allusers=True):

"""Returns a list of prefixes for all old installations of this arch so that

they can be removed from PATH if present. Note, it would be preferable to

uninstall them properly instead.

"""

if allusers:

# All Users

key, subkey = (reg.HKEY\_LOCAL\_MACHINE, r'SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Uninstall\\')

else:

# Just Me

key, subkey = (reg.HKEY\_CURRENT\_USER, r'SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Uninstall\\')

keylist = []

# We ignore pyversion and instead look for any \*conda installations.

regex = re.compile('Python \S+ \(\S+conda[0-9]+ \S+ '+arch+'\)')

\_reg\_query\_sub\_keys(key, subkey, keylist)

results = []

for uninstsubkey in keylist:

final\_part = os.path.basename(uninstsubkey.rstrip('\\'))

if regex.match(final\_part):

try:

with reg.OpenKeyEx(key, uninstsubkey, 0,

reg.KEY\_QUERY\_VALUE) as keyhandle:

reg\_value = reg.QueryValueEx(keyhandle, 'UninstallString')

results.append(os.path.dirname(re.sub(r'^"|"$', '', reg\_value[0])))

except:

pass

return results

**Example 20**

|  |  |  |
| --- | --- | --- |
| Project: *chalktalk\_docs*   Author: *loremIpsum1771*   File: *img.py*    [(license)](https://github.com/loremIpsum1771/chalktalk_docs)[*View Source Project*](https://github.com/loremIpsum1771/chalktalk_docs/tree/master/ctdocs_env/lib/python2.7/site-packages/pygments/formatters/img.py) | 5 votes | vote downvote up |

def \_create\_win(self):

try:

key = \_winreg.OpenKey(

\_winreg.HKEY\_LOCAL\_MACHINE,

r'Software\Microsoft\Windows NT\CurrentVersion\Fonts')

except EnvironmentError:

try:

key = \_winreg.OpenKey(

\_winreg.HKEY\_LOCAL\_MACHINE,

r'Software\Microsoft\Windows\CurrentVersion\Fonts')

except EnvironmentError:

raise FontNotFound('Can\'t open Windows font registry key')

try:

path = self.\_lookup\_win(key, self.font\_name, STYLES['NORMAL'], True)

self.fonts['NORMAL'] = ImageFont.truetype(path, self.font\_size)

for style in ('ITALIC', 'BOLD', 'BOLDITALIC'):

path = self.\_lookup\_win(key, self.font\_name, STYLES[style])

if path:

self.fonts[style] = ImageFont.truetype(path, self.font\_size)

else:

if style == 'BOLDITALIC':

self.fonts[style] = self.fonts['BOLD']

else:

self.fonts[style] = self.fonts['NORMAL']

finally:

\_winreg.CloseKey(key)

**Example 21**

|  |  |  |
| --- | --- | --- |
| Project: *ShuoshuoMonitor*   Author: *aploium*   File: *firefox\_binary.py*    [(license)](https://github.com/aploium/ShuoshuoMonitor)[*View Source Project*](https://github.com/aploium/ShuoshuoMonitor/tree/master/selenium/webdriver/firefox/firefox_binary.py) | 5 votes | vote downvote up |

def \_find\_exe\_in\_registry(self):

try:

from \_winreg import OpenKey, QueryValue, HKEY\_LOCAL\_MACHINE, HKEY\_CURRENT\_USER

except ImportError:

from winreg import OpenKey, QueryValue, HKEY\_LOCAL\_MACHINE, HKEY\_CURRENT\_USER

import shlex

keys = (

r"SOFTWARE\Classes\FirefoxHTML\shell\open\command",

r"SOFTWARE\Classes\Applications\firefox.exe\shell\open\command"

)

command = ""

for path in keys:

try:

key = OpenKey(HKEY\_LOCAL\_MACHINE, path)

command = QueryValue(key, "")

break

except OSError:

try:

key = OpenKey(HKEY\_CURRENT\_USER, path)

command = QueryValue(key, "")

break

except OSError:

pass

else:

return ""

if not command:

return ""

return shlex.split(command)[0]

**Example 22**

|  |  |  |
| --- | --- | --- |
| Project: *rice*   Author: *randy3k*   File: *util.py*    [(license)](https://github.com/randy3k/rice)[*View Source Project*](https://github.com/randy3k/rice/tree/master/rice/util.py) | 5 votes | vote downvote up |

def read\_registry(key, valueex):

reg\_key = OpenKey(HKEY\_LOCAL\_MACHINE, key, 0, KEY\_READ)

return QueryValueEx(reg\_key, valueex)

**Example 23**

|  |  |  |
| --- | --- | --- |
| Project: *sublimeTextConfig*   Author: *luoye-fe*   File: *img.py*    [(license)](https://github.com/luoye-fe/sublimeTextConfig)[*View Source Project*](https://github.com/luoye-fe/sublimeTextConfig/tree/master/pygments/all/pygments/formatters/img.py) | 5 votes | vote downvote up |

def \_create\_win(self):

try:

key = \_winreg.OpenKey(

\_winreg.HKEY\_LOCAL\_MACHINE,

r'Software\Microsoft\Windows NT\CurrentVersion\Fonts')

except EnvironmentError:

try:

key = \_winreg.OpenKey(

\_winreg.HKEY\_LOCAL\_MACHINE,

r'Software\Microsoft\Windows\CurrentVersion\Fonts')

except EnvironmentError:

raise FontNotFound('Can\'t open Windows font registry key')

try:

path = self.\_lookup\_win(key, self.font\_name, STYLES['NORMAL'], True)

self.fonts['NORMAL'] = ImageFont.truetype(path, self.font\_size)

for style in ('ITALIC', 'BOLD', 'BOLDITALIC'):

path = self.\_lookup\_win(key, self.font\_name, STYLES[style])

if path:

self.fonts[style] = ImageFont.truetype(path, self.font\_size)

else:

if style == 'BOLDITALIC':

self.fonts[style] = self.fonts['BOLD']

else:

self.fonts[style] = self.fonts['NORMAL']

finally:

\_winreg.CloseKey(key)

**Example 24**

|  |  |  |
| --- | --- | --- |
| Project: *CodeReader*   Author: *jasonrbr*   File: *win32timezone.py*    [(license)](https://github.com/jasonrbr/CodeReader)[*View Source Project*](https://github.com/jasonrbr/CodeReader/tree/master/site-packages/win32/lib/win32timezone.py) | 5 votes | vote downvote up |

def \_FindTimeZoneKey(self):

"""Find the registry key for the time zone name (self.timeZoneName)."""

# for multi-language compatability, match the time zone name in the

# "Std" key of the time zone key.

zoneNames = dict(self.\_get\_indexed\_time\_zone\_keys('Std'))

# Also match the time zone key name itself, to be compatible with

# English-based hard-coded time zones.

timeZoneName = zoneNames.get(self.timeZoneName, self.timeZoneName)

key = \_RegKeyDict.open(winreg.HKEY\_LOCAL\_MACHINE, self.tzRegKey)

try:

result = key.subkey(timeZoneName)

except:

raise ValueError('Timezone Name %s not found.' % timeZoneName)

return result

**Example 25**

|  |  |  |
| --- | --- | --- |
| Project: *CodeReader*   Author: *jasonrbr*   File: *win32timezone.py*    [(license)](https://github.com/jasonrbr/CodeReader)[*View Source Project*](https://github.com/jasonrbr/CodeReader/tree/master/site-packages/win32/lib/win32timezone.py) | 5 votes | vote downvote up |

def \_get\_time\_zone\_key(subkey=None):

"Return the registry key that stores time zone details"

key = \_RegKeyDict.open(winreg.HKEY\_LOCAL\_MACHINE, TimeZoneInfo.tzRegKey)

if subkey:

key = key.subkey(subkey)

return key

**Example 26**

|  |  |  |
| --- | --- | --- |
| Project: *CodeReader*   Author: *jasonrbr*   File: *ietoolbar.py*    [(license)](https://github.com/jasonrbr/CodeReader)[*View Source Project*](https://github.com/jasonrbr/CodeReader/tree/master/site-packages/win32com/demos/ietoolbar.py) | 5 votes | vote downvote up |

def DllRegisterServer():

comclass = IEToolbar

# register toolbar with IE

try:

print("Trying to register Toolbar.\n")

hkey = winreg.CreateKey( winreg.HKEY\_LOCAL\_MACHINE, "SOFTWARE\\Microsoft\\Internet Explorer\\Toolbar" )

subKey = winreg.SetValueEx( hkey, comclass.\_reg\_clsid\_, 0, winreg.REG\_BINARY, "\0" )

except WindowsError:

print("Couldn't set registry value.\nhkey: %d\tCLSID: %s\n" % ( hkey, comclass.\_reg\_clsid\_ ))

else:

print("Set registry value.\nhkey: %d\tCLSID: %s\n" % ( hkey, comclass.\_reg\_clsid\_ ))

# TODO: implement reg settings for standard toolbar button

# unregister plugin

**Example 27**

|  |  |  |
| --- | --- | --- |
| Project: *amazon\_order\_history\_scraper*   Author: *drewctate*   File: *firefox\_binary.py*    [(license)](https://github.com/drewctate/amazon_order_history_scraper)[*View Source Project*](https://github.com/drewctate/amazon_order_history_scraper/tree/master/selenium/webdriver/firefox/firefox_binary.py) | 5 votes | vote downvote up |

def \_find\_exe\_in\_registry(self):

try:

from \_winreg import OpenKey, QueryValue, HKEY\_LOCAL\_MACHINE, HKEY\_CURRENT\_USER

except ImportError:

from winreg import OpenKey, QueryValue, HKEY\_LOCAL\_MACHINE, HKEY\_CURRENT\_USER

import shlex

keys = (r"SOFTWARE\Classes\FirefoxHTML\shell\open\command",

r"SOFTWARE\Classes\Applications\firefox.exe\shell\open\command")

command = ""

for path in keys:

try:

key = OpenKey(HKEY\_LOCAL\_MACHINE, path)

command = QueryValue(key, "")

break

except OSError:

try:

key = OpenKey(HKEY\_CURRENT\_USER, path)

command = QueryValue(key, "")

break

except OSError:

pass

else:

return ""

if not command:

return ""

return shlex.split(command)[0]

**Example 28**

|  |  |  |
| --- | --- | --- |
| Project: *Repobot*   Author: *Desgard*   File: *img.py*    [(license)](https://github.com/Desgard/Repobot)[*View Source Project*](https://github.com/Desgard/Repobot/tree/master/repobot-lenv/lib/python3.6/site-packages/pygments/formatters/img.py) | 5 votes | vote downvote up |

def \_create\_win(self):

try:

key = \_winreg.OpenKey(

\_winreg.HKEY\_LOCAL\_MACHINE,

r'Software\Microsoft\Windows NT\CurrentVersion\Fonts')

except EnvironmentError:

try:

key = \_winreg.OpenKey(

\_winreg.HKEY\_LOCAL\_MACHINE,

r'Software\Microsoft\Windows\CurrentVersion\Fonts')

except EnvironmentError:

raise FontNotFound('Can\'t open Windows font registry key')

try:

path = self.\_lookup\_win(key, self.font\_name, STYLES['NORMAL'], True)

self.fonts['NORMAL'] = ImageFont.truetype(path, self.font\_size)

for style in ('ITALIC', 'BOLD', 'BOLDITALIC'):

path = self.\_lookup\_win(key, self.font\_name, STYLES[style])

if path:

self.fonts[style] = ImageFont.truetype(path, self.font\_size)

else:

if style == 'BOLDITALIC':

self.fonts[style] = self.fonts['BOLD']

else:

self.fonts[style] = self.fonts['NORMAL']

finally:

\_winreg.CloseKey(key)

**Example 29**

|  |  |  |
| --- | --- | --- |
| Project: *webapp*   Author: *superchilli*   File: *firefox\_binary.py*    [(license)](https://github.com/superchilli/webapp)[*View Source Project*](https://github.com/superchilli/webapp/tree/master/venv/lib/python2.7/site-packages/selenium/webdriver/firefox/firefox_binary.py) | 5 votes | vote downvote up |

def \_find\_exe\_in\_registry(self):

try:

from \_winreg import OpenKey, QueryValue, HKEY\_LOCAL\_MACHINE, HKEY\_CURRENT\_USER

except ImportError:

from winreg import OpenKey, QueryValue, HKEY\_LOCAL\_MACHINE, HKEY\_CURRENT\_USER

import shlex

keys = (

r"SOFTWARE\Classes\FirefoxHTML\shell\open\command",

r"SOFTWARE\Classes\Applications\firefox.exe\shell\open\command"

)

command = ""

for path in keys:

try:

key = OpenKey(HKEY\_LOCAL\_MACHINE, path)

command = QueryValue(key, "")

break

except OSError:

try:

key = OpenKey(HKEY\_CURRENT\_USER, path)

command = QueryValue(key, "")

break

except OSError:

pass

else:

return ""

if not command:

return ""

return shlex.split(command)[0]

**Example 30**

|  |  |  |
| --- | --- | --- |
| Project: *nsf2x*   Author: *adb014*   File: *mapiex.py*    [(license)](https://github.com/adb014/nsf2x)[*View Source Project*](https://github.com/adb014/nsf2x/tree/master/mapiex.py) | 5 votes | vote downvote up |

def MimeToMapi (self, eml, m, flag = 0) :

if self.converter == None :

clsid = win32com.mapi.mapi.CLSID\_IConverterSession

iid = win32com.mapi.mapi.IID\_IConverterSession

try:

tmp = pythoncom.CoCreateInstance (clsid, None, pythoncom.CLSCTX\_INPROC\_SERVER, pythoncom.IID\_IUnknown)

self.converter = tmp.QueryInterface (iid)

except :

# Test for ClickToRun version of Outlook and manually load library and create instance

for iconvpath in ("", "16.0", "15.0") :

regpath = os.path.join ("Software","Microsoft","Office",iconvpath,"ClickToRun","Registry","Machine","Software","Classes")

if platform.machine() == "AMD64" and platform.architecture()[0] == "32bit":

# 32bit application on 64bit platform

regpath = os.path.join (regpath,"Wow6432Node")

regpath = os.path.join (regpath,"CLSID", str(clsid),"InprocServer32")

self.converter = self.CoCreateInstanceC2R (winreg.HKEY\_LOCAL\_MACHINE, regpath, clsid, iid)

if self.converter != None :

break

if self.converter == None :

NameError("mapi:MimeToMapi : Can not create IConverterSession instance")

# Open file as IStream. Don't use win32com.mapi.mapi.OpenStreamOnFile as it doesn't

# handle Unicode file names

f = open(eml, "rb")

Istrm = util.wrap (FileStream(f), pythoncom.IID\_IUnknown, None, True)

self.converter.MIMEToMAPI(Istrm, m, flag)

**Example 31**

|  |  |  |
| --- | --- | --- |
| Project: *nsf2x*   Author: *adb014*   File: *nsf2x.py*    [(license)](https://github.com/adb014/nsf2x)[*View Source Project*](https://github.com/adb014/nsf2x/tree/master/nsf2x.py) | 5 votes | vote downvote up |

def OutlookPath():

"""Function to retrieve the path to Outlook from the registry"""

aReg = winreg.ConnectRegistry(None, winreg.HKEY\_LOCAL\_MACHINE)

aKey = winreg.OpenKey(aReg, r"SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths\OUTLOOK.EXE")

# prepend unused variables with "dummy\_" to keep PyLint happy

dummy\_n, v, dummy\_t = winreg.EnumValue(aKey, 0)

winreg.CloseKey(aKey)

winreg.CloseKey(aReg)

return v

**Example 32**

|  |  |  |
| --- | --- | --- |
| Project: *teleport*   Author: *eomsoft*   File: *env.py*    [(license)](https://github.com/eomsoft/teleport)[*View Source Project*](https://github.com/eomsoft/teleport/tree/master/dist/server/script/core/env.py) | 5 votes | vote downvote up |

def \_get\_msbuild(self):

# 14.0 = VS2015

# 12.0 = VS2012

# 4.0 = VS2008

chk = ['14.0', '12.0', '4.0']

p = None

for c in chk:

p = self.\_winreg\_read(winreg.HKEY\_LOCAL\_MACHINE, r'SOFTWARE\Microsoft\MSBuild\ToolsVersions\{}'.format(c), r'MSBuildToolsPath')

if p is not None:

break

return os.path.join(p[0], 'MSBuild.exe') if p is not None else None

**Example 33**

|  |  |  |
| --- | --- | --- |
| Project: *teleport*   Author: *eomsoft*   File: *env.py*    [(license)](https://github.com/eomsoft/teleport)[*View Source Project*](https://github.com/eomsoft/teleport/tree/master/dist/server/script/core/env.py) | 5 votes | vote downvote up |

def \_get\_visual\_studio\_path(self):

chk = ['14.0', '12.0', '4.0']

p = None

for c in chk:

p = self.\_winreg\_read(winreg.HKEY\_LOCAL\_MACHINE, r'SOFTWARE\Microsoft\VisualStudio\{}'.format(c), r'ShellFolder')

if p is not None:

break

return p[0] if p is not None else None

**Example 34**

|  |  |  |
| --- | --- | --- |
| Project: *teleport*   Author: *eomsoft*   File: *env.py*    [(license)](https://github.com/eomsoft/teleport)[*View Source Project*](https://github.com/eomsoft/teleport/tree/master/dist/server/script/core/env.py) | 5 votes | vote downvote up |

def \_get\_perl(self):

p = self.\_winreg\_read(winreg.HKEY\_LOCAL\_MACHINE, r'SOFTWARE\perl', 'BinDir')

return p[0] if p is not None else None

**Example 35**

|  |  |  |
| --- | --- | --- |
| Project: *teleport*   Author: *eomsoft*   File: *env.py*    [(license)](https://github.com/eomsoft/teleport)[*View Source Project*](https://github.com/eomsoft/teleport/tree/master/dist/server/script/core/env.py) | 5 votes | vote downvote up |

def \_get\_nsis(self):

p = self.\_winreg\_read(winreg.HKEY\_LOCAL\_MACHINE, r'SOFTWARE\NSIS\Unicode', '')

if p is None:

p = self.\_winreg\_read(winreg.HKEY\_LOCAL\_MACHINE, r'SOFTWARE\NSIS', '')

return os.path.join(p[0], 'makensis.exe') if p is not None else None

**Example 36**

|  |  |  |
| --- | --- | --- |
| Project: *teleport*   Author: *eomsoft*   File: *env.py*    [(license)](https://github.com/eomsoft/teleport)[*View Source Project*](https://github.com/eomsoft/teleport/tree/master/build/builder/core/env.py) | 5 votes | vote downvote up |

def \_get\_msbuild(self):

# 14.0 = VS2015

# 12.0 = VS2012

# 4.0 = VS2008

chk = ['14.0', '12.0', '4.0']

p = None

for c in chk:

p = self.\_winreg\_read(winreg.HKEY\_LOCAL\_MACHINE, r'SOFTWARE\Microsoft\MSBuild\ToolsVersions\{}'.format(c), r'MSBuildToolsPath')

if p is not None:

break

return os.path.join(p[0], 'MSBuild.exe') if p is not None else None

**Example 37**

|  |  |  |
| --- | --- | --- |
| Project: *teleport*   Author: *eomsoft*   File: *env.py*    [(license)](https://github.com/eomsoft/teleport)[*View Source Project*](https://github.com/eomsoft/teleport/tree/master/build/builder/core/env.py) | 5 votes | vote downvote up |

def \_get\_perl(self):

p = self.\_winreg\_read(winreg.HKEY\_LOCAL\_MACHINE, r'SOFTWARE\perl', 'BinDir')

return p[0] if p is not None else None

**Example 38**

|  |  |  |
| --- | --- | --- |
| Project: *teleport*   Author: *eomsoft*   File: *env.py*    [(license)](https://github.com/eomsoft/teleport)[*View Source Project*](https://github.com/eomsoft/teleport/tree/master/build/builder/core/env.py) | 5 votes | vote downvote up |

def \_get\_nsis(self):

p = self.\_winreg\_read(winreg.HKEY\_LOCAL\_MACHINE, r'SOFTWARE\NSIS\Unicode', '')

if p is None:

p = self.\_winreg\_read(winreg.HKEY\_LOCAL\_MACHINE, r'SOFTWARE\NSIS', '')

return os.path.join(p[0], 'makensis.exe') if p is not None else None

**Example 39**

|  |  |  |
| --- | --- | --- |
| Project: *enkiWS*   Author: *juliettef*   File: *img.py*    [(license)](https://github.com/juliettef/enkiWS)[*View Source Project*](https://github.com/juliettef/enkiWS/tree/master/pygments/formatters/img.py) | 5 votes | vote downvote up |

def \_create\_win(self):

try:

key = \_winreg.OpenKey(

\_winreg.HKEY\_LOCAL\_MACHINE,

r'Software\Microsoft\Windows NT\CurrentVersion\Fonts')

except EnvironmentError:

try:

key = \_winreg.OpenKey(

\_winreg.HKEY\_LOCAL\_MACHINE,

r'Software\Microsoft\Windows\CurrentVersion\Fonts')

except EnvironmentError:

raise FontNotFound('Can\'t open Windows font registry key')

try:

path = self.\_lookup\_win(key, self.font\_name, STYLES['NORMAL'], True)

self.fonts['NORMAL'] = ImageFont.truetype(path, self.font\_size)

for style in ('ITALIC', 'BOLD', 'BOLDITALIC'):

path = self.\_lookup\_win(key, self.font\_name, STYLES[style])

if path:

self.fonts[style] = ImageFont.truetype(path, self.font\_size)

else:

if style == 'BOLDITALIC':

self.fonts[style] = self.fonts['BOLD']

else:

self.fonts[style] = self.fonts['NORMAL']

finally:

\_winreg.CloseKey(key)

**Example 40**

|  |  |  |
| --- | --- | --- |
| Project: *kbe\_server*   Author: *xiaohaoppy*   File: *installer.py*    [(license)](https://github.com/xiaohaoppy/kbe_server)[*View Source Project*](https://github.com/xiaohaoppy/kbe_server/tree/master/kbengine/kbe/tools/server/install/installer.py) | 5 votes | vote downvote up |

def getWindowsEnvironmentKey(scope):

assert scope in ('user', 'system')

root = winreg.HKEY\_CURRENT\_USER

subkey = 'Environment'

if scope != 'user':

root = winreg.HKEY\_LOCAL\_MACHINE

subkey = r'SYSTEM\CurrentControlSet\Control\Session Manager\Environment'

return (root, subkey)

**Example 41**

|  |  |  |
| --- | --- | --- |
| Project: *python-flask-security*   Author: *weinbergdavid*   File: *img.py*    [(license)](https://github.com/weinbergdavid/python-flask-security)[*View Source Project*](https://github.com/weinbergdavid/python-flask-security/tree/master/env/Lib/site-packages/pygments/formatters/img.py) | 5 votes | vote downvote up |

def \_create\_win(self):

try:

key = \_winreg.OpenKey(

\_winreg.HKEY\_LOCAL\_MACHINE,

r'Software\Microsoft\Windows NT\CurrentVersion\Fonts')

except EnvironmentError:

try:

key = \_winreg.OpenKey(

\_winreg.HKEY\_LOCAL\_MACHINE,

r'Software\Microsoft\Windows\CurrentVersion\Fonts')

except EnvironmentError:

raise FontNotFound('Can\'t open Windows font registry key')

try:

path = self.\_lookup\_win(key, self.font\_name, STYLES['NORMAL'], True)

self.fonts['NORMAL'] = ImageFont.truetype(path, self.font\_size)

for style in ('ITALIC', 'BOLD', 'BOLDITALIC'):

path = self.\_lookup\_win(key, self.font\_name, STYLES[style])

if path:

self.fonts[style] = ImageFont.truetype(path, self.font\_size)

else:

if style == 'BOLDITALIC':

self.fonts[style] = self.fonts['BOLD']

else:

self.fonts[style] = self.fonts['NORMAL']

finally:

\_winreg.CloseKey(key)

**Example 42**

|  |  |  |
| --- | --- | --- |
| Project: *blender*   Author: *gastrodia*   File: *img.py*    [(license)](https://github.com/gastrodia/blender)[*View Source Project*](https://github.com/gastrodia/blender/tree/master/library/site-packages/pygments/formatters/img.py) | 5 votes | vote downvote up |

def \_create\_win(self):

try:

key = \_winreg.OpenKey(

\_winreg.HKEY\_LOCAL\_MACHINE,

r'Software\Microsoft\Windows NT\CurrentVersion\Fonts')

except EnvironmentError:

try:

key = \_winreg.OpenKey(

\_winreg.HKEY\_LOCAL\_MACHINE,

r'Software\Microsoft\Windows\CurrentVersion\Fonts')

except EnvironmentError:

raise FontNotFound('Can\'t open Windows font registry key')

try:

path = self.\_lookup\_win(key, self.font\_name, STYLES['NORMAL'], True)

self.fonts['NORMAL'] = ImageFont.truetype(path, self.font\_size)

for style in ('ITALIC', 'BOLD', 'BOLDITALIC'):

path = self.\_lookup\_win(key, self.font\_name, STYLES[style])

if path:

self.fonts[style] = ImageFont.truetype(path, self.font\_size)

else:

if style == 'BOLDITALIC':

self.fonts[style] = self.fonts['BOLD']

else:

self.fonts[style] = self.fonts['NORMAL']

finally:

\_winreg.CloseKey(key)

**Example 43**

|  |  |  |
| --- | --- | --- |
| Project: *WindowsEnumeration*   Author: *tdmathison*   File: *WinEnum.py*    [(license)](https://github.com/tdmathison/WindowsEnumeration)[*View Source Project*](https://github.com/tdmathison/WindowsEnumeration/tree/master/WinEnum.py) | 5 votes | vote downvote up |

def walk\_registry(hkey, path, access\_flags, keywords, onerror=None):

""" Walks all keys of the registry searching for values that match any of the provided 'keywords'. """

try:

key = winreg.OpenKey(hkey, path, access\_flags)

except OSError as e:

if onerror is not None:

onerror(e)

return

i = 0

sub\_keys = []

with key:

while True:

try:

sub\_keys.append(winreg.EnumKey(key, i))

except OSError:

break

i += 1

i = 0

while True:

try:

data = winreg.EnumValue(key, i)

i += 1

for keyword in keywords:

if keyword.lower() in str(data[0]).lower():

if hkey == winreg.HKEY\_LOCAL\_MACHINE:

hive = 'HKLM\\'

else:

hive = 'HKCU\\'

print('{0}\\{1}\\{2} = {3}'.format(hive, path, data[0], data[1]))

except OSError:

break

for key in sub\_keys:

next\_path = os.path.join(path, key)

for item in walk\_registry(hkey, next\_path, access\_flags, keywords, onerror):

yield item

**Example 44**

|  |  |  |
| --- | --- | --- |
| Project: *yatta\_reader*   Author: *sound88*   File: *img.py*    [(license)](https://github.com/sound88/yatta_reader)[*View Source Project*](https://github.com/sound88/yatta_reader/tree/master/yatta/lib/python3.6/site-packages/pygments/formatters/img.py) | 5 votes | vote downvote up |

def \_create\_win(self):

try:

key = \_winreg.OpenKey(

\_winreg.HKEY\_LOCAL\_MACHINE,

r'Software\Microsoft\Windows NT\CurrentVersion\Fonts')

except EnvironmentError:

try:

key = \_winreg.OpenKey(

\_winreg.HKEY\_LOCAL\_MACHINE,

r'Software\Microsoft\Windows\CurrentVersion\Fonts')

except EnvironmentError:

raise FontNotFound('Can\'t open Windows font registry key')

try:

path = self.\_lookup\_win(key, self.font\_name, STYLES['NORMAL'], True)

self.fonts['NORMAL'] = ImageFont.truetype(path, self.font\_size)

for style in ('ITALIC', 'BOLD', 'BOLDITALIC'):

path = self.\_lookup\_win(key, self.font\_name, STYLES[style])

if path:

self.fonts[style] = ImageFont.truetype(path, self.font\_size)

else:

if style == 'BOLDITALIC':

self.fonts[style] = self.fonts['BOLD']

else:

self.fonts[style] = self.fonts['NORMAL']

finally:

\_winreg.CloseKey(key)

**Example 45**

|  |  |  |
| --- | --- | --- |
| Project: *Packages*   Author: *Keypirinha*   File: *putty.py*    [(license)](https://github.com/Keypirinha/Packages)[*View Source Project*](https://github.com/Keypirinha/Packages/tree/master/PuTTY/putty.py) | 5 votes | vote downvote up |

def \_autodetect\_official\_installreg(self):

try:

key = winreg.OpenKey(

winreg.HKEY\_LOCAL\_MACHINE,

"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Uninstall\\PUTTY\_is1",

access=winreg.KEY\_READ | winreg.KEY\_WOW64\_32KEY)

value = winreg.QueryValueEx(key, "InstallLocation")[0]

winreg.CloseKey(key)

exe\_file = os.path.join(value, self.EXE\_NAME\_OFFICIAL)

if os.path.exists(exe\_file):

return exe\_file

except:

pass

return None

**Example 46**

|  |  |  |
| --- | --- | --- |
| Project: *Packages*   Author: *Keypirinha*   File: *filezilla.py*    [(license)](https://github.com/Keypirinha/Packages)[*View Source Project*](https://github.com/Keypirinha/Packages/tree/master/FileZilla/filezilla.py) | 5 votes | vote downvote up |

def \_autodetect\_official\_installreg(self):

try:

key = winreg.OpenKey(

winreg.HKEY\_LOCAL\_MACHINE,

"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Uninstall\\FileZilla Client",

access=winreg.KEY\_READ | winreg.KEY\_WOW64\_32KEY)

value = winreg.QueryValueEx(key, "InstallLocation")[0]

winreg.CloseKey(key)

exe\_file = os.path.join(value, self.EXE\_NAME\_OFFICIAL)

if os.path.exists(exe\_file):

return exe\_file

except:

pass

return None

**Example 47**

|  |  |  |
| --- | --- | --- |
| Project: *Packages*   Author: *Keypirinha*   File: *winscp.py*    [(license)](https://github.com/Keypirinha/Packages)[*View Source Project*](https://github.com/Keypirinha/Packages/tree/master/WinSCP/winscp.py) | 5 votes | vote downvote up |

def \_autodetect\_official\_installreg(self):

try:

key = winreg.OpenKey(

winreg.HKEY\_LOCAL\_MACHINE,

"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Uninstall\\winscp3\_is1",

access=winreg.KEY\_READ | winreg.KEY\_WOW64\_32KEY)

value = winreg.QueryValueEx(key, "InstallLocation")[0]

winreg.CloseKey(key)

exe\_file = os.path.join(value, self.EXE\_NAME\_OFFICIAL)

if os.path.exists(exe\_file):

return exe\_file

except:

pass

return None

**Example 48**

|  |  |  |
| --- | --- | --- |
| Project: *lib9*   Author: *Jumpscale*   File: *Windows.py*    [(license)](https://github.com/Jumpscale/lib9)[*View Source Project*](https://github.com/Jumpscale/lib9/tree/master/JumpScale9Lib/sal/windows/Windows.py) | 5 votes | vote downvote up |

def \_getHiveAndKey(self, fullKey):

'''Split a windows registry key in two parts: the hive (hkey) and the registry key

Eg: "HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion" will return: (\_winreg.HKEY\_LOCAL\_MACHINE, "SOFTWARE\Microsoft\Windows\CurrentVersion")

'''

str\_hkey, str\_key = fullKey.split('\\', 1)

hiveType = WinRegHiveType.getByName(str\_hkey.lower())

return hiveType.hive, str\_key

**Example 49**

|  |  |  |
| --- | --- | --- |
| Project: *lib9*   Author: *Jumpscale*   File: *Windows.py*    [(license)](https://github.com/Jumpscale/lib9)[*View Source Project*](https://github.com/Jumpscale/lib9/tree/master/JumpScale9Lib/sal/windows/Windows.py) | 5 votes | vote downvote up |

def exportRegKeysToString(self, key):

"""Exports Windows registry key to a string

This function exports a Windows registry key to a string (ini-file format).

@param key: The registry key to export. The key should include the section. Eg. "HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion"

@type key: string

"""

strBuffer = StringIO()

regfile = IniFile(strBuffer)

self.\_addValuesRecursively(regfile, key)

return regfile.getContent()

**Example 50**

|  |  |  |
| --- | --- | --- |
| Project: *lib9*   Author: *Jumpscale*   File: *Windows.py*    [(license)](https://github.com/Jumpscale/lib9)[*View Source Project*](https://github.com/Jumpscale/lib9/tree/master/JumpScale9Lib/sal/windows/Windows.py) | 5 votes | vote downvote up |

def exportRegKeysToFile(self, key, path):

"""Exports Windows registry key to a file

This function exports a Windows registry key to an ini-file.

@param key: The registry key to export. The key should include the section. Eg. "HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion"

@type key: string

@param path: The path of the file to export to

@type path: string

"""

j.sal.fs.writeFile(path, self.exportRegKeysToString(key))