### Iterar informações sobre todos os filhos findChildren(QObject)

# Suponha que self.window seja sua janela principal (QQuickWindow ou similar)

from PySide6.QtCore import QObject

children = self.window.findChildren(QObject)

for child in children:

print("Objeto:", child)

print(" Classe:", child.metaObject().className())

print(" ID:", child.objectName())

print(" Properties:", child.dynamicPropertyNames())

print("---")

* findChildren(QObject) retorna todos os objetos filhos recursivamente.
* child.metaObject().className() mostra o tipo QML (ex: Rectangle, Text, MouseArea).
* child.objectName() mostra o objectName definido em QML, se houver.
* child.dynamicPropertyNames() lista todas as property definidas via setProperty ou property no QML.

Só botões personalizados:

botões = self.window.findChildren(QObject, options=Qt.FindChildrenRecursively)

for botão in botões:

if botão.metaObject().className() == "BotaoEstilizado":

print("Botão:", botão.objectName())

### Outras informações

Além de metaObject().className(), você pode acessar muitas outras informações úteis sobre os objetos QML no Python via QObject ou QQuickItem.

**Informações básicas do objeto:**

Propriedade Python Descrição

obj.objectName() Nome do objeto (objectName definido em QML)

obj.metaObject().className() Nome da classe real do QML (ex: Rectangle, Text, MouseArea)

obj.property("foo") Valor de uma propriedade chamada "foo"

obj.dynamicPropertyNames() Lista de nomes de propriedades definidas dinamicamente

obj.parent() Pai do objeto (outro QObject)

obj.children() Filhos diretos do objeto

**Métodos do metaobjeto:**

Através de obj.metaObject(), você também pode acessar:

meta = obj.metaObject()

for i in range(meta.propertyCount()):

prop = meta.property(i)

print(f"Property: {prop.name()} = {obj.property(prop.name())}")

**Verificando sinais, slots e propriedades:**

meta = obj.metaObject()

print("Propriedades:")

for i in range(meta.propertyCount()):

prop = meta.property(i)

print(" ", prop.name())

print("Sinais:")

for i in range(meta.methodCount()):

method = meta.method(i)

if method.methodType() == method.Signal:

print(" ", method.name())

print("Slots:")

for i in range(meta.methodCount()):

method = meta.method(i)

if method.methodType() == method.Slot:

print(" ", method.name())

**Para QQuickItem especificamente:**

Se o objeto for um item visual (ex: Rectangle, Text, etc.), ele herda de QQuickItem, então:

from PySide6.QtQuick import QQuickItem

if isinstance(obj, QQuickItem):

print("x:", obj.x(), "y:", obj.y())

print("width:", obj.width(), "height:", obj.height())

print("visible:", obj.isVisible())