3.9.5. Callable

Fig. 3.9.5.1 Photo by Taylor Grote on Unsplash

Note

Outline

Note

Roadmap

1. This topic: TryExcept

> myBlock
>
> | Exception Handling | | | | |
> |---|---|---|---|---|
> | try Statement | except Statement | finally Statement | Error Type | with Statement |

2. Course: Python 1
3. Subject: Programming
4. Field
    a. Software Engineering (SE)
    b. Computer Science and Information Engineering (CSIE)
    c. Electrical/Electronics Engineering (EE)

3.9.5.1. Introduction

1. The callable() function in Python returns True if the object passed appears callable i.e. can be called like a function.
2. It returns False otherwise.
3. Some key notes on callable().
    a. Returns True for functions, methods, classes.
    b. Returns False for non-callables like class instances.
    c. Can check if something is callable before calling it.
    d. Helps avoid TypeErrors.
    e. Prefer over calling isinstance(x, collections.Callable).

3.9.5.2. Examples

3.9.5.2.1. Ex1: callable()

1. Code+Output

> Code Output
>
> a. getattr() takes the object and attribute name as strings.
> b. Line 12: gets 'gender', if not existed, returns 'Male' as default.
>
> **Listing 3.9.5.2.1.1 /src/ExceptionHandling/Callable/callable.py**

```
'''
author: cph
since:  20230805
'''
def add(x, y):
  return x + y

class Person:
    def __init__(self, name):
        self.name = name

if __name__ =='__main__':
    print(f'callable(add): {callable(add)}')
    print(f'callable(len): {callable(len)}')

    oC = [(1), (2, ), [3], {4}, {5: 'a'}]
    for i in oC:
        print(f'callable({i}): {callable(oC)}')

    p = Person('Andy')
    print(f'callable(p): {callable(p)}')
```

3.9.5.2.2. Ex2: callable()

1. Code+Output

Code | Output

Listing 3.9.5.2.2.1 /src/ExceptionHandling/Callable/callable2.py

```
'''
author: cph
since:  20230805
'''
def execute(callback):
    if callable(callback):  # 2nd check
        callback()

def callback():
    print(f'This is a callback function...')

if __name__ =='__main__':
    if callable(execute):   # 1st check
        execute(callback)
```

Important

Summary: Callable

1. callable() checks if an object appears callable i.e. can be called like a function.
2. Useful for validating callbacks.

Important

Summary: Exception Handling

1. Here is a summary of built-in Python functions related to try/except error handling.

Build-in	Description
try	Defines a block of code to test for errors.
except	Defines a block of code to execute if an error occurs in the try block. Can specify the error type(s) to handle.
else	Optional block executed if no error occurred in try block.
finally	Optional block that executes after try/except regardless of error.
raise	Manually raises an exception.
assert	Verify condition is true, raise AssertionError if not.
hasattr()	Check if an object has a given attribute.
getattr()	Get attribute from an object if present.
callable()	Check if an object is callable like a function.
repr()	Return printable representation of object for debugging.

2. The try/except construct allows gracefully handling exceptions in Python.
3. finally always runs after try/except to release resources.
4. Built-ins like hasattr(), repr() are useful in try/except for introspection and diagnostics.
5. Proper use of try/except/finally blocks is important for robust error handling.

3.9.5.3. Homework

3.9.5.3.1. Hw1

Question Code

```
請使用例外處理，判斷讀入檔案(file.txt)是否成功。
當讀檔發生錯誤時，輸出"讀檔錯誤"；
可以順利執行則輸出"檔案已讀入"。
```

Note

1. Start: 20170719
2. System Environment:

Listing 3.9.5.3.1.1 requirements.txt

```
sphinx==7.1.2                           # Sphinx
graphviz>=0.20.1                        # Graphviz
sphinxbootstrap4theme>=0.6.0            # Theme: Bootstrap
sphinx-material>=0.0.35                 # Theme: Material
sphinxcontrib-plantuml>=0.25            # PlantUML
sphinxcontrib.bibtex>=2.5.0             # Bibliography
sphinx-autorun>=1.1.1                   # ExecCode: pycon
sphinx-execute-code-python3>=0.3        # ExecCode
btd.sphinx.inheritance-diagram>=2.3.1   # Diagram
sphinx-copybutton>=0.5.1                # Copy button
sphinx_code_tabs>=0.5.3                 # Tabs
sphinx-immaterial>=0.11.3               # Tabs

#-----------------------------------------------------
#-- Library Upgrade Error by Library Itself
#   >> It needs to fix by library owner
#   >> After fixed, we need to try it later
#-----------------------------------------------------
pydantic==1.10.10                       # 2.0: sphinx compiler error, 20230701

#-----------------------------------------------------
#-- Minor Extension
#-----------------------------------------------------
sphinxcontrib.httpdomain>=1.8.1         # HTTP API

#sphinxcontrib-blockdiag>=3.0.0         # Diagram: block
#sphinxcontrib-actdiag>=3.0.0           # Diagram: activity
#sphinxcontrib-nwdiag>=2.0.0            # Diagram: network
#sphinxcontrib-seqdiag>=3.0.0           # Diagram: sequence

#-----------------------------------------------------
#-- Still Wait For Upgrading Version
#-----------------------------------------------------

#-----------------------------------------------------
#-- Still Under Testing
#-----------------------------------------------------
#numpy>=1.24.2                          # Figure: numpy

#-----------------------------------------------------
#-- NOT Workable
#-----------------------------------------------------
#sphinxcontrib.jsdemo==0.1.4    # ExecCode: Need replace add_js_file()
#jupyter-sphinx==0.4.0          # ExecCode: Need gcc compiler
#sphinxcontrib.slide==1.0.0     # Slide: Slideshare
#hieroglyph==2.1.0              # Slide: make slides
#matplotlib>=3.7.1              # Plot: Need Python >= v3.8
#manim==0.17.2                  # Diagram: scipy, numpy need gcc
#sphinx_diagrams==0.4.0         # Diagram: Need GKE access
#sphinx-tabs>=3.4.1                     # Tabs: Conflict w/ sphinx-material
```