Curso Práctico de Python: Creación de un CRUD

Artículo

Context managers

David Aroesti 17 de Octubre de 2018

Los context managers son objetos de Python que proveen información contextual adicional al bloque de código. Esta información consiste en correr una función (o cualquier callable) cuando se inicia el contexto con el keyword with; al igual que correr otra función cuando el código dentro del bloque with concluye. Por ejemplo:

```
with open('some_file.txt') as f:
    lines = f.readlines()
```

Si estás familiarizado con este patrón, sabes que llamar la función open de esta manera, garantiza que el archivo se cierre con posterioridad. Esto disminuye la cantidad de información que el programador debe manejar directamente y facilita la lectura del código.

Existen dos formas de implementar un context manager: con una clase o con un generador. Vamos a implementar la funcionalidad anterior para ilustrar el punto:

```
class CustomOpen(object):
    def __init__(self, filename):
        self.file = open(filename)

    def __enter__(self):
        return self.file
```

```
with CustomOpen('file') as f:
    contents = f.read()
```

sta es simplemente una clase de Python con dos métodos adicionales: **enter** y **xit**. Estos métodos son utilizados por el keyword with para determinar las cciones de inicialización, entrada y salida del contexto.

mismo código puede implementarse utilizando el módulo contextlib que forma arte de la librería estándar de Python.

```
from contextlib import contextmanager

@contextmanager
def custom_open(filename):
    f = open(filename)
    try:
        yield f
    finally:
        f.close()

with custom_open('file') as f:
    contents = f.read()
```

código anterior funciona exactamente igual que cuando lo escribimos con una ase. La diferencia es que el código se ejecuta al inicializarse el contexto y retorna control cuando el keyword yield regresa un valor. Una vez que termina el bloque ith, el context manager toma de nueva cuenta el control y ejecuta el código de mpieza.

Escribe aquí tu pregunta +2

wilantury Estudiante · hace 18 días

Documentación:
- https://docs.python.org/3/library/contextlib.html?highlight=contex manager

Aplicaciones de Python en el ...

Edwin Jesset Barrientos Gonzales Estudiante • hace 2 meses

En otras palabras, el protocolo de gestor de contextos definidos en el PEP 343 permite la extracción de la parte aburrida de la estructura try...except...finally en una clase separada manteniendo solo el bloque de interés do_something.
Primero se llama al método **enter**. Puede devolver un valor que será asignado a var. La parte as es opcional: si no está presente, el valor devuelto por **enter** será ignorado.
El bloque de código bajo with se ejecutará. Como si fueran condiciones try. Se podrá ejecutar con éxito hasta el final o, si algo falla, puede break, continue` o return o lanzar una excepción. Pase lo que pase, una vez que el bloque ha finalizado, se producirá una llamada al método **exit**. Si se lanzó una excepción, la información se manda a **exit**, el cual se describe en la siguiente subsección. En el caso normal, las excepciones podrían ser ignoradas como si fueran una condición finally y serían relanzadas después de que finalice **exit**.

2

Rubén Maier Enzler Estudiante • hace 4 meses

se explica en algún momento de este curso o del curso de python el uso de yield?

1

Jorge Mayorga • hace 4 meses

yield es cuando se retorna una secuencia de valores y no un valor final:
Mira aqui

5

Alejo RM • hace 4 meses

Si, en este mismo curso: https://platzi.com/clases/1378-python-practico/14334-iterators-and-generators/

5

Ver más respuestas

sergio-beltran Estudiante • hace 7 meses

creo que comprendi, tengo que practicarlo mas

1

Aplicaciones de Python en el ...

1

Facundo Nicolás García Martoni • hace 6 meses

¡Hola @SanGeeky! Te dejo este link de un artículo muy bueno sobre los **Context Managers** en Python.
A mi me sirvió mucho para comprender los diferentes casos de uso 😃

1

Andrés Mauricio Montoya Sánchez Estudiante • el año pasado

Los métodos **init**, **enter** y **exit** los entendí como los React lifecycles jajaa

1

Victor Daniel Aguirre Gil Estudiante • el año pasado

Python esta lleno de trucos que casi parecen mágia.

5

MatiasBZ Estudiante • el año pasado

Les dejo un enlace donde se explica muy bien
http://book.pythontips.com/en/latest/context_managers.html

9

elbarbero400 Estudiante • el año pasado

```
from contextlib import contextmanager

classCustomOpen(object):
  def__init__(self, filename):
    self.file = open(filename)

  def__enter__(self):
    return self.file

  def__exit__(self, ctx_type, ctx_value, ctx_traceback):
    self.file.close()
```

Aplicaciones de Python en el ...

```
@contextmanager
defcustom_open(filename):
  f = open(filename)
  try:
    yield f
  finally:
    f.close()


if __name__ == '__main__':
  ## custom = CustomOpen('operaciones_con_listas.py')
  custom_open('operaciones_con_listas.py')
  print(read_file('operaciones_con_listas.py'))
  print('-'*30)
```

3

ennma5 Estudiante • el año pasado

Hola estimado David (@jdaroesti), me encanta la forma en la que explicas y enseñas, desde el curso pasado soy tu fan.
Sin embargo encontré en el presente artículo algo que podría mejorar:
En el primer ejemplo sobre el uso de with pegaste with con open, lo que ocasionará en la práctica un error de sintaxis.

```
withopen(‘some_file.txt’) as f:
    lines = f.readlines()
```

Siento que fue un error de dedo, pero que podría confundir a las personas que comienzan con el lenguaje.
La forma correcta del primer ejemplo sería:

```
withopen(‘some_file.txt’) as f:
    lines = f.readlines()
```

El mismo detalle ocurre en el ejemplo siguiente en la línea:

```
classCustomOpen(object):def__init__(self, filename):
```

Aplicaciones de Python en el ...

Ojalá no me vea muy chocoso con éste comentario, pero siento que tu calidad como profesional de python, es muy alta y éste pequeño error podría quitar coherencia a tu imagen.
Excelente curso, ojalá fueras mentor de otros cursos de programación como javascript.
Saludos.

3

cmorales • el año pasado

Como que le quedó igual el código estimado colega. Al parecer es problema del editor de platzi.

10

Nestor Cepeda • el año pasado

El editor tiende a pegar las palabras, si le das copy - paste debes desagruparlas.

2

ennma5 • el año pasado

Que mal plan con el editor de platzi, ojalá en el futuro logren corregir el detalle.

4

David Daniel Castillo Nava • el año pasado

jajaja me dio un poco de risa :3 por que fue el editor siempre hace eso!

3

Facundo Nicolás García Martoni • hace 6 meses

Un pequeño aporte a este comentario; si ustedes colocan el nombre del lenguaje que están usando después de las primeras tres comillas invertidas --> Así (```python) <-- Van a tener un remarcado de sintaxis correcto por parte del editor. Los invito a probarlo 😃

2

Ver más respuestas