Python Generators

Generator

- What is a generator
- Using generators in for loop
- Using generators in expressions
- Terminating generators
- Raising errors
- Closing iterators
- Sending values to generators
- Data pipelines

Generator

- **A function that can be suspended and resumed**

```
# defines a generator function
def get_id():
    value = 1
    while True:
        yield value # yields a value and suspends operation
        value += 1  # resumes operation
```

- **A function that returns iterator**

```
id = get_id() # gets an iterator

print(next(id)) # fetches 1st value
print(next(id)) # fetches 2nd value
print(next(id)) # fetches 3rd value
```

Generator

- **Using iterators in the for loop**

```
id = get_id() # gets an iterator
for number in id:
    if(number < 10):
        print(number)
    else:
        break
```

- **Generator expressions**

```
list = [n for n in range(5)] # returns list
series = (n for n in range(5)) # returns generator
```

Generator

- **Generator function can also terminate**

```
# defines a generator function that terminates
def get_evens(limit):
    value = 2
    count = 0
    while True:
        if count < limit:
            yield value # yields value and suspends operation
            # resumes operation
            value += 2
            count += 1
        else:
            return # terminates
```

- **A terminated generator raises error**

```
StopIteration
```

Generator

- **Iterator can be closed**

```
id = get_id() # gets iterator
next(id) # gets next value
id.close() # closes the iterator, can’t be used anymore
```

- **Iterator can throw exception**

```
id.throw(ValueError) # raises error and closes iterator
```

Generator

- **Generator function can receive input**

```
def get_multiples():
    value = 1
    multiplier = 1
    while True:
        multiplier = 1 if multiplier == None else multiplier
        # yields a value and also receives a value
        multiplier = yield value * multiplier
        value += 1
```

- **Sending value to Generator**

```
series = get_multiples()
print(series.send(None)) # can't send value initially
print(series.send(3)) # can send
print(next(series)) # usual way
```

Generator

- **Generators can use other generators**

```
def get_id():
    value = 1
    while True:
        yield value # yields a value and suspends operation
        value += 1  # resumes operation

def get_doubles(sequence):
    for n in sequence: # gets values from generator
        yield n * 2
```

- **Generators form data pipelines**

```
id = get_id()
double = get_doubles(id)
print(next(double))
print(next(double))
```

Thanks