Intermediate Python

January 6, 2025

1 Intermediate Level Python

1.1 Getting you up to speed

This course assumes that you're at an intermediate level of python. For example, you should have a decent idea what something like this might do:

```
yield from {book.get("author") for book in books if book.get("author")}
```

If not - then you've come to the right place! Welcome to the crash course in intermediate level python. The best way to learn is by doing!

1.2 First: if you need a refresher on the foundations

I'm going to defer to an AI friend for this, because these explanations are so well written with great examples. Copy and paste the code examples into a new cell to give them a try. Pick whichever section(s) you'd like to brush up on.

Python imports:
https://chatgpt.com/share/672f9f31-8114-8012-be09-29ef0d0140fb

Python functions including default arguments:
https://chatgpt.com/share/672f9f99-7060-8012-bfec-46d4cf77d672

Python strings, including slicing, split/join, replace and literals:
https://chatgpt.com/share/672fb526-0aa0-8012-9e00-ad1687c04518

Python f-strings including number and date formatting:
https://chatgpt.com/share/672fa125-0de0-8012-8e35-27918cbb481c

Python lists, dicts and sets, including the `get()` method:
https://chatgpt.com/share/672fa225-3f04-8012-91af-f9c95287da8d

Python files including modes, encoding, context managers, Path, glob.glob:
https://chatgpt.com/share/673b53b2-6d5c-8012-a344-221056c2f960

Python classes:
https://chatgpt.com/share/672fa07a-1014-8012-b2ea-6dc679552715

Pickling Python objects and converting to JSON:
https://chatgpt.com/share/673b553e-9d0c-8012-9919-f3bb5aa23e31

[1]:
```python
# Next let's create some things:

fruits = ["Apples", "Bananas", "Pears"]

book1 = {"title": "Great Expectations", "author": "Charles Dickens"}
book2 = {"title": "Bleak House", "author": "Charles Dickens"}
book3 = {"title": "An Book By No Author"}
book4 = {"title": "Moby Dick", "author": "Herman Melville"}

books = [book1, book2, book3, book4]
```

2 Part 1: List and dict comprehensions

[2]:
```python
# Simple enough to start

for fruit in fruits:
    print(fruit)
```

```
Apples
Bananas
Pears
```

[3]:
```python
# Let's make a new version of fruits

fruits_shouted = []
for fruit in fruits:
    fruits_shouted.append(fruit.upper())

fruits_shouted
```

[3]:
```
['APPLES', 'BANANAS', 'PEARS']
```

[4]:
```python
# You probably already know this
# There's a nice Python construct called "list comprehension" that does this:

fruits_shouted2 = [fruit.upper() for fruit in fruits]
fruits_shouted2
```

[4]:
```
['APPLES', 'BANANAS', 'PEARS']
```

[5]:
```python
# But you may not know that you can do this to create dictionaries, too:

fruit_mapping = {fruit: fruit.upper() for fruit in fruits}
fruit_mapping
```

[5]:
```
{'Apples': 'APPLES', 'Bananas': 'BANANAS', 'Pears': 'PEARS'}
```

```
[6]: # you can also use the if statement to filter the results

     fruits_with_longer_names_shouted = [fruit.upper() for fruit in fruits if␣
      ↪len(fruit)>5]
     fruits_with_longer_names_shouted
```

```
[6]: ['APPLES', 'BANANAS']
```

```
[7]: fruit_mapping_unless_starts_with_a = {fruit: fruit.upper() for fruit in fruits␣
      ↪if not fruit.startswith('A')}
     fruit_mapping_unless_starts_with_a
```

```
[7]: {'Bananas': 'BANANAS', 'Pears': 'PEARS'}
```

```
[8]: # Another comprehension

     [book['title'] for book in books]
```

```
[8]: ['Great Expectations', 'Bleak House', 'An Book By No Author', 'Moby Dick']
```

```
[9]: # This code will fail with an error because one of our books doesn't have an␣
      ↪author

     [book['author'] for book in books]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[9], line 3
      1 # This code will fail with an error because one of our books doesn't␣
  ↪have an author
----> 3 [book['author'] for book in books]

Cell In[9], line 3, in <listcomp>(.0)
      1 # This code will fail with an error because one of our books doesn't␣
  ↪have an author
----> 3 [book['author'] for book in books]

KeyError: 'author'
```

```
[10]: # But this will work, because get() returns None

      [book.get('author') for book in books]
```

```
[10]: ['Charles Dickens', 'Charles Dickens', None, 'Herman Melville']
```

[11]:
```python
# And this variation will filter out the None

[book.get('author') for book in books if book.get('author')]
```

[11]:
```
['Charles Dickens', 'Charles Dickens', 'Herman Melville']
```

[12]:
```python
# And this version will convert it into a set, removing duplicates

set([book.get('author') for book in books if book.get('author')])
```

[12]:
```
{'Charles Dickens', 'Herman Melville'}
```

[13]:
```python
# And finally, this version is even nicer
# curly braces creates a set, so this is a set comprehension

{book.get('author') for book in books if book.get('author')}
```

[13]:
```
{'Charles Dickens', 'Herman Melville'}
```

3 Part 2: Generators

We use Generators in the course because AI models can stream back results.

If you've not used Generators before, please start with this excellent intro from ChatGPT:

https://chatgpt.com/share/672faa6e-7dd0-8012-aae5-44fc0d0ec218

Try pasting some of its examples into a cell.

[14]:
```python
# First define a generator; it looks like a function, but it has yield instead␣
↪of return

import time

def come_up_with_fruit_names():
    for fruit in fruits:
        time.sleep(1) # thinking of a fruit
        yield fruit
```

[15]:
```python
# Then use it

for fruit in come_up_with_fruit_names():
    print(fruit)
```

```
Apples
Bananas
Pears
```

[16]:
```
# Here's another one

def authors_generator():
    for book in books:
        if book.get("author"):
            yield book.get("author")
```

[17]:
```
# Use it

for author in authors_generator():
    print(author)
```

```
Charles Dickens
Charles Dickens
Herman Melville
```

[18]:
```
# Here's the same thing written with list comprehension

def authors_generator():
    for author in [book.get("author") for book in books if book.get("author")]:
        yield author
```

[19]:
```
# Use it

for author in authors_generator():
    print(author)
```

```
Charles Dickens
Charles Dickens
Herman Melville
```

[20]:
```
# Here's a nice shortcut
# You can use "yield from" to yield each item of an iterable

def authors_generator():
    yield from [book.get("author") for book in books if book.get("author")]
```

[21]:
```
# Use it

for author in authors_generator():
    print(author)
```

```
Charles Dickens
Charles Dickens
Herman Melville
```

[22]:
```
# And finally - we can replace the list comprehension with a set comprehension

def unique_authors_generator():
    yield from {book.get("author") for book in books if book.get("author")}
```

[23]:
```
# Use it

for author in unique_authors_generator():
    print(author)
```

```
Herman Melville
Charles Dickens
```

[24]:
```
# And for some fun - press the stop button in the toolbar when bored!
# It's like we've made our own Large Language Model... although not␣
 ↪particularly large..
# See if you understand why it prints a letter at a time, instead of a word at␣
 ↪a time. If you're unsure, try removing the keyword "from" everywhere in the␣
 ↪code.

import random
import time

pronouns = ["I", "You", "We", "They"]
verbs = ["eat", "detest", "bathe in", "deny the existence of", "resent",␣
 ↪"pontificate about", "juggle", "impersonate", "worship", "misplace",␣
 ↪"conspire with", "philosophize about", "tap dance on", "dramatically␣
 ↪renounce", "secretly collect"]
adjectives = ["turqoise", "smelly", "arrogant", "festering", "pleasing",␣
 ↪"whimsical", "disheveled", "pretentious", "wobbly", "melodramatic",␣
 ↪"pompous", "fluorescent", "bewildered", "suspicious", "overripe"]
nouns = ["turnips", "rodents", "eels", "walruses", "kumquats", "monocles",␣
 ↪"spreadsheets", "bagpipes", "wombats", "accordions", "mustaches",␣
 ↪"calculators", "jellyfish", "thermostats"]

def infinite_random_sentences():
    while True:
        yield from random.choice(pronouns)
        yield " "
        yield from random.choice(verbs)
        yield " "
        yield from random.choice(adjectives)
        yield " "
        yield from random.choice(nouns)
        yield ". "

for letter in infinite_random_sentences():
    print(letter, end="", flush=True)
    time.sleep(0.02)
```

```
You pontificate about bewildered calculators. We dramatically renounce
```

```
pretentious kumquats. I conspire with suspicious monocles. They eat wobbly
mustaches. We juggle pretentious monocles. They pontificate about pleasing
spreadsheets. You dramatically renounce festering kumquats. They eat disheveled
calculators. We impersonate smelly mustaches. We dramatically renounce festerin
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[24], line 26
     24 for letter in infinite_random_sentences():
     25     print(letter, end="", flush=True)
---> 26     time.sleep(0.02)

KeyboardInterrupt:
```

4 Exercise

Write some python classes for the books example.

Write a Book class with a title and author. Include a method has_author()

Write a BookShelf class with a list of books. Include a generator method unique_authors()

5 Finally

Here are some intermediate level details of Classes from our AI friend, including use of type hints, inheritance and class methods. This includes a Book example.

https://chatgpt.com/share/67348aca-65fc-8012-a4a9-fd1b8f04ba59