Python Culture

and other topics

John Hanley

17th September 2024

supermoon!

partial eclipse

Agenda:

- talk
- demo

python syntax – *args, tuple unpack

```
def elevation(lat, lon):
    return ...

sf = (37.8, -122.4)
lat, lon = sf
```

These are the same:

```
print(elevation(lat, lon))

print(elevation(*sf))
```

syntax – **kwargs

```
loc = {"lat": 37.8,
       "lon": -122.4}

def elevation_api(location: dict[str, float]) -> float:
```

These are the same:

```
    return elevation(location["lat"],
                     location["lon"])

    return elevation(lat=location["lat"],
                     lon=location["lon"])

    return elevation(**location)
```

syntax – walrus operator

```python
import re

line = 'On 2024-09-15 "Shogun" won 18 Emmys'

if match := re.search(r"\d{4}-\d{2}-\d{2}", line):
    ymd = match[0]
```

same as:

```python
match = re.search(r"\d{4}-\d{2}-\d{2}", line)
if match:
    ymd = match[0]
```

syntax – ellipsis

Yes, it's an object. It's a singleton.

```
>>> ... is None
False
>>>
>>> ... is ...
True
```

syntax – ellipsis

These are the same. Kind of.

```
class FaceUnrecognizedError(ValueError):
    pass

class FaceUnrecognizedError(ValueError):
    ...

class FaceUnrecognizedError(ValueError):
    a = 1
    a
```

syntax – `@` decorator

```python
from time import sleep, time

def timed(fn):
    def wrapper(*args, **kw):
        start = time()
        result = fn(*args, **kw)
        print(f"{fn.__name__} took {time() - start} seconds")
        return result
    return wrapper

@timed
def slow(n: float) -> None:
    sleep(2 * n)

slow(4)
```

numeric tower

see PEP-484

> *when an argument is annotated as having type* `float`*, an argument of type* `int` *is acceptable*

list vs. tuple

Both are sequences. (One is mutable, the other: immutable.)

We use a list for an arbitrary number of "same thing", e.g. a list of `names = ["Alice", "Bob"]`.

We use a tuple for a fixed number of things where position changes meaning, e.g. a `(lat, lon)` pair.

list vs. tuple

Here's an analogy: a pythonista uses a tuple where a C programmer would use a `struct`.

Also, consider naming the tuple elements.

```
from collections import namedtuple
from typing_extensions import NamedTuple

Location = namedtuple("Location", "lat, lon")

class Location(NamedTuple):
    lat: float
    lon: float
```

list vs. tuple

Dataclasses are mutable, but they may also be a good fit for the "struct" use case.

```
from dataclasses import dataclass

@dataclass
class Location:
    lat: float
    lon: float
```

exceptions

EAFP – Easier to Ask Forgiveness than Permission

LBYL – Look Before You Leap

In some environments we try hard to avoid triggering exceptions.

In a python context, we often prefer to let exceptions happen, and deal with the blowback.

exceptions

Non-pythonic versus pythonic:

```
import re

def convert_to_celsius(f_temperature: str) -> float:
    if match := re.search(r"^\d+(\.\d+)?$", f_temperature):
        return (float(match[0]) - 32) * 5 / 9
    return float("NaN")


def convert_to_celsius(f_temperature: str) -> float:
    try:
        return (float(f_temperature) - 32) * 5 / 9
    except ValueError:
        return float("NaN")
```

testing – more flexible

Run with

- `$ python -m unittest *.py`, or ...
- `$ pytest *.py`

```
import unittest
from math import isnan

class CelsiusTest(unittest.TestCase):

    def test_convert_to_celsius(self):
        self.assertEqual(0.0, convert_to_celsius("32"))
        self.assertEqual(100.0, convert_to_celsius("212"))
        self.assertTrue(isnan(convert_to_celsius("Brrr, cold!")))
```

testing – less flexible

Run with

- `$ pytest *.py`

```
def test_convert_to_celsius():
    assert 0.0 == convert_to_celsius("32")
    assert 100.0 == convert_to_celsius("212")
    assert isnan(convert_to_celsius("hot and humid"))
```

main guard

Protect your code from being executed during an `import`.

```
if __name__ == "__main__":
    report()
```

Why? So another module can `import` your module without side effects. No delay, no print(), no fail if host or file not found, none of that.

And there *will* be another importing module, because someone *will* write a test suite. Even if you didn't.

numbers

- `float` – 53-bits of significand
- `complex` – a vector of two floats

```
>>> a = 3 + 4j
>>> b = 1 + 2j
>>>
>>> a + b
(4+6j)
>>>
>>> (a + b).real
4.0
>>> (a + b).imag
6.0
```

real numbers

- `int` – these will never overflow
- `float` – 53-bits of significand
- `Decimal` – scaled integers, for financial figures
- `Fraction` – a rational number $\frac{p}{q}$, both of them `int`s

“real” numbers

```
>>> from decimal import Decimal
>>> from fractions import Fraction

>>> .10 + .20
0.30000000000000004
>>>
>>> Decimal("0.1") + Decimal("0.2")
Decimal('0.3')
>>>
>>>
>>> Fraction(1, 10) + Fraction(1, 5)
Fraction(3, 10)
>>>
>>> Fraction(2, 10) == Fraction(1, 5)
True
```

bare except

Avoid this.

```
try:
    ...
except:
    ...
```

Prefer to name an exception.

```
try:
    ...
except Exception:
    ...
```

re-throwing

Preserve diagnostic information, such as source line numbers.

```
try:
    ...
except ValueError as e:
    ...
    raise MyAppError("ouch!") from e
```

iterating

Avoid this:

```
for i in range(len(names)):
    print(names[i])
```

Prefer one of these:

```
for name in names:
    print(name)

for i, name in enumerate(names):
    print(i, name)
```

parallel lists

Avoid this:

```
assert len(names) == len(ages)

for i in range(len(names)):
    print(names[i], ages[i])
```

Prefer this:

```
for name, age in zip(names, ages):
    print(name, age)
```

Or create a named tuple. Or a dataclass.

next time... pull requests

the PR process

addendum – presenters who are new to the Dojo

Checklist:

- ☐ Visit the Dojo on a Tuesday night before your talk.
- ☐ Speak with Peter or John about the proposed topic.
- ☐ Practice discord VoiceBox screen sharing with Peter or John.
- ☐ Bring an HDMI video adapter to the Dojo, and try it out.
- ☐ Bring your own laptop,
- ☐ with power supply, so it won't die part way through.
- ☐ Arrive before 6:30pm, so you can test the screen share setup.
- ☐ Enable your microphone.
- ☐ Request a mic check in the python channel.
- ☐ Verify that both browser and bash prompt are visible in discord.