Typing and Type Annotations (Type Hints) Practice

Useful sources are

Python typing package on docs.python.org

Mypy cheat sheet [https://mypy.readthedocs.io/en/stable/cheat\_sheet\_py3.html](./Useful%20sources%20%20are%0A%20%20Python%20typing%20package%20on%20docs.python.org%0A%20%20Mypy%20cheat%20sheet%20https://mypy.readthedocs.io/en/stable/cheat_sheet_py3.html)

1. Complete this table.

In the "Example use" column, assume that **x** refers to an object that provides the Type in the left column.   
As an example, for Sized type:

# string is a Sized type

**x = "strings have length"**

|  |  |  |
| --- | --- | --- |
| Type | Provides methods | Example use (\*) |
|  | \_\_call\_\_() | x = MyCallable()  x() |
| Sized |  | len(x) |
|  | \_\_next\_\_() | while True:  print(next(x)) |
|  | \_\_iter\_\_() | # 2 typical uses that do not  # explicitly call iter() |
|  |  | "apple" in x |
|  | combines 3 types: | "apple" in x # True or False  len(x)  [print(item) for item in x] |
|  | \_\_getitem\_\_()  \_\_len\_\_() | x[2]  x["foo"]  *Name the most basic type that specifies this behavior* |

2. We have a Scorecard class that creates an *iterator.* Add type hints to specify that Scorecard creates an iterator always returns float values?

class Scorecard( ):

def \_\_init\_\_(self, name):

self.name = name

self.scores = []

3. Fill in the blanks with correct types. Use the most specific type hint

today: = datetime.today()

weekend: = today.isoweekday()==0 or today.isoweekday()==6

# Define "Number" as a type that can be either float or int

Number = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Write type hints for average to make it as general as possible.

That means, you want to specify what the parameter should be but don't put unnecessary restrictions on the parameter type.

def average(items: ) -> :

return sum(items)/max(1, len(items))

5. Add type hints to the code below.

**class Product:**

"""A kind of item that the store sells, e.g Nescafe Ice Coffee."""

def \_\_init\_\_(self, product\_id: \_\_\_\_\_\_\_\_\_,

description: \_\_\_\_\_\_\_\_,

price: \_\_\_\_\_\_\_\_\_\_\_):

self.id: str = product\_id

self.description = description

self.unit\_price = price

**class LineItem:**

"""LineItem represents the purchase of a product, with a quantity"""

def \_\_init\_\_(self, product: \_\_\_\_\_\_\_, quantity: \_\_\_\_\_\_ = 1):

self.product = product

self.quantity = quantity

def get\_total(self) \_\_\_\_\_\_\_\_\_\_\_\_:

return self.product.unit\_price \* self.quantity

def \_\_str\_\_(self) \_\_\_\_\_\_\_\_\_\_\_\_:

return self.product.description

**class Sale(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_):**

"""A sale of a collection of LineItems"""

def \_\_init\_\_(self):

self.items: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ = []

def add\_item(self, item: \_\_\_\_\_\_\_\_\_\_\_\_):

"""Add a LineItem to this sale"""

self.items.append(item)

def total(self) \_\_\_\_\_\_\_\_\_\_\_\_:

total\_price = sum( item.get\_total() for item in self.items )

tax = TaxCalc.get\_tax(total\_price)

return total\_price + tax

def \_\_iter\_\_(self):

return iter(self.items)

def \_\_len\_\_(self):

return len(self.items)

**class TaxCalc:**

# tax rate is a static (class) value

TAX\_RATE = 0.07

@classmethod

def get\_tax(cls, amount: \_\_\_\_\_\_\_\_\_\_) \_\_\_\_\_\_\_\_\_\_\_\_\_\_:

"""compute the tax on given amount"""

return cls.TAX\_RATE \* amount

6. Write the type annotation that matches each of these examples.

|  |  |
| --- | --- |
| Example | Type Annotation |
| [1, 2, 3] or [1, 2.2] or [0.0, 1.5] |  |
| {'one': 1, 'two': 2} but not {'one': '1', 'two': '2'} |  |
| lambda x: math.sqrt(x) |  |
| lambda s: s.isalpha() |  |
| In Django: |  |
| Question.objects.filter(  question\_text\_\_contains="the") |  |

7. For each of these type annotations, give an example object that would pass the `mypy` check for that type. For example, if the type annotation was used on a function and you invoke the function using your example object, mypy would not issue any warnings.

Example:

Type annotation

|  |  |
| --- | --- |
| Type Annotation | Example Conforming Object |
| dict[str, dict[str, Any]] | {'1': {'question\_id': 1,  'question\_text': "How old are you?",  'answer': None  }  } |
| Iterable[tuple[float, float]] |  |
| Sequence[int | float]  but the example object is not:  Sequence[int] or Sequence[float] |  |
|  |  |