Type Checking

Type Checking

Verify that the rules for using different types are obeyed, and that the correct types are used in function calls, assignments, and other program elements.

```
Example:
lst = ["cat", "dog", "rat"]
sum( lst )                # type error
for x in range(1.0,4.0):
    # type error: int required
    print(x)
```

Static versus Dynamic

static - fixed, unchanging, immobile

In computer programming:

anything that is done or known before run-time.

"static content" - fixed content in a web application, such as images, fonts, CSS files, fixed web pages.

"static type checking" - type checking done before the program is run.

- may be done by compiler or static type checking tool.

Static versus Dynamic

dynamic - characterized by change or activity

In computer programming:

anything that is done, created, or known only when the code is run.

"dynamic content" - web pages generated at run-time from a template. Content that changes over time.

"dynamic type checking" - verify type rules while the program is running.

Does Python Do Dynamic Type Checking?

Answer is not obvious.

Consider this:

```
# what type is required for x and y?
def join(x, y):
    return x + y


join(2, 3)
join("hi", "bye")
join(Fraction(1,2), Fraction(2,3))
```

What Some People Say

Python does dynamic type checking.

Python associates types with *values* rather than *variables*.

Type checking is done on values.

Static versus Dynamic Binding

"Binding" refers to association of names with particular pieces of code.

Static Binding - a name is "bound" to particular code in an unchanging (static) way.

Dynamic Binding - a name is "bound" to code in a dynamic, changing way (at run-time).

@staticmethod

```
class Fraction:
    @staticmethod
    def gcd( m, n):
        """greatest common divisor"""
        # use Euclid's algorithm
```

gcd is statically bound. We know exactly what code will be invoked even before the program is run!

```
x = Fraction.gcd(60, 75)
```

Dynamic binding

```
lst = [ Fraction(2,3), "hello",
        datetime.now() ]
for x in lst:
    print( str(x) )
2/3
'hello'
2019-11-17 15:50:34
```

`str(x)` is dynamically bound to the __str__() method of a particular class (Fraction, string, datetime).

We don't know until run-time what kind of object x refers to, or which class's __str__() method should be invoked.

Static Checking & Software Correctness

We want our software to be correct (of course).

Static type checking finds some programming errors before the program is run.

Some type errors may also indicate *logic errors*.

Example

```
class Scorecard:
    """Accumulate scores and compute their average."""
    def __init__(self):
        self.scores = []

    def add_score(self, score):
        self.scores.append(score)

    def average(self):
        """return average of all scores"""
        return sum(self.scores)/len(self.scores)

if __name__ == "__main__":
    scores = Scorecard()
    n = input("input a score: ")
    scores.add_score(n)
    n = input("input another score: ")
    scores.add_score(n)
    print("The average is " + scores.average())
```

This simple code contains 2 distinct errors. As written, most IDE won't detect them.

Exercise

1. Download scorecard.py to an empty directory.
2. Open in your favorite IDE.
3. Does the IDE flag any errors?
4. Add *type hints* ... one at a time.

   Please don't add them all at once.

   "hint" the parameter to addScore. What happens?

   "hint" the scores List (from typing import List).

   What happens?

For VS Code, if you don't have Pyright extension, then install it. See any difference?

Typing and Encapsulation

In Scorecard, the scores are assumed to be numbers.

Can we allow scores to be objects?

quiz_score = Score("Quiz 1", 10.0)

In Scorecard we could write:

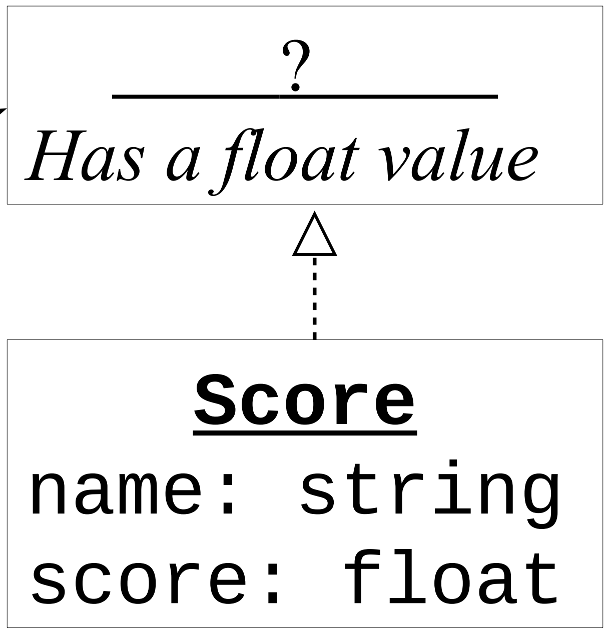
```
def average(self):
    # add the values of the score objects
    total = sum(float(x) for x in self.scores)
    # don't divide by zero if no scores
    return total/max(1, len(self.scores))
```

Typing and Encapsulation

What is the ***required behavior*** of a score object,
so that Scorecard can compute the average score?

```
def add_score(self, score: ____?____):
```

Is there a "type" we can use that means "this object has a float value, call float(x) to get it"?



Resources

Mai's write-up on "type hinting" in ISP19/problems Github repository.