CSC110 Lecture 8: Function Specification and Property-Based Testing

Print this handout

Exercise 1: Reviewing preconditions and type annotations

1. What is the relationship between a function's *parameter type annotations* and a function's *preconditions*?

2. The following function calculates the pay for an employee who worked for a given time period (e.g., 10am–4pm) at a given hourly pay rate (e.g., $15/hour). Write Python expressions for preconditions to express the constraints on the function inputs described in the docstring.

```
def calculate_pay(start: int, end: int, pay_rate: float) -> float:
    """Return the pay of an employee who worked for the given time at the given pay

    start and end represent the hour (from 0 to 23 inclusive) that the employee
    started and ended their work.

    pay_rate is the hourly pay rate and must be >= 15.0 (the minimum wage).

    Preconditions:









    >>> calculate_pay(3, 5, 15.5)
    31.0
    >>> calculate_pay(9, 21, 22.0)
    264.0
    """
    return (end - start) * pay_rate
```

3. For each of the following Python values, write down the most specific type annotation for that value.

Python value	Type annotation
`[1, 2, 3]`	
`{'hi', 'bye', 'haha'}`	
`{1.5: True, 3.6: False, -1.0: True}`	
`{1.5: [1, 2, 3], 3.6: [4, 5, 6]}`	

4. For each of the following descriptions of (collection) data, write the most specific type annotation for that data.

Description of Data	Type annotation
A study music playlist (song names)	
A colour in the RGB24 model (as a list!)	
David's grocery list (food names and associated quantities)	
An unordered collection of distinct names	

5. When would we use the type annotation `list` instead of `list[...]` (with a type in the square brackets)?

Exercise 2: Property-based testing

1. Consider the following function, which is a generalization of `is_even` that checks whether one number is divisible by another. (We'll discuss this implementation, and the "`d == 0`" case, more in tomorrow's lecture.)

```
def divides(d: int, n: int) -> bool:
    """Return whether d divides n.

    >>> divides(3, 9)   # Is 9 divisible by 3? (Yes)
    True
    >>> divides(3, 10)  # Is 10 divisible by 3? (No)
    False
    """
    if d == 0:
        return n == 0
    else:
        return n % d == 0
```

There are many different properties of divisibility from mathematics that we can use to express property-based tests. For each of the properties below, translate them into a property-based test. We've started the first one for you (you can copy-and-paste the template and use it for each one; you don't need to repeat the `import` statements).

   a. For all integers $n \in \mathbb{Z}$, 2 divides $2 \times n$. (This is very similar to our `is_even` example.)

```
from hypothesis import given
from hypothesis.strategies import integers


@given(n=integers())
def test_a(n: int) -> None:
    """Test the following property of the divides function:

    For all integers n, 2 divides (2 * n).
    (This is very similar to our `is_even` example.)
    """
```

   b. $\forall n, d \in \mathbb{Z},\ d \mid d \times n$

      Hint: you can use the syntax `@given(n=..., d=...)` to tell hypothesis to generate values for multiple funtion parameters.

   c. $\forall n, d \in \mathbb{Z}^+,\ d \mid n \Rightarrow d \leq n$

      Notes:

      - Use `integers(min_value=1)` instead of `integers()` to return a strategy that generates only positive integer values for n and d.
      - Recall that $p \Rightarrow q$ is equivalent to $\neg p \vee q$.

Additional exercises

1. *Preconditions and if statements.* Implement each of the two functions below. Note that these functions only differ in how they handle the "special case" when `age < 0`.

```
def ticket_price_v1(age: int) -> float:
    """Return the ticket price for a person who is age years old.

    Seniors 65 and over pay 4.75, kids 12 and under pay 4.25, and
    everyone else pays 7.50.

    Preconditions:
      - age >= 0

    >>> ticket_price_v1(7)
    4.25
    >>> ticket_price_v1(21)
    7.5
    >>> ticket_price_v1(101)
    4.75
    """


def ticket_price_v2(age: int) -> float:
    """Return the ticket price for a person who is age years old.

    Seniors 65 and over pay 4.75, kids 12 and under pay 4.25, and
    everyone else pays 7.50.

    If the given age is negative, then the ticket_price is 0.0.

    >>> ticket_price_v2(7)
    4.25
    >>> ticket_price_v2(21)
    7.5
    >>> ticket_price_v2(101)
    4.75
    """
```

2. *More property-based testing practice.*

   Using the same `divides` function as in Exercise 2, write property-based tests to represent each of the following mathematical properties of divisibility.

   a. For all integers d, d divides itself.

   b. For all integers n, 1 divides n.

   c. $\forall n, d \in \mathbb{Z},\ d \mid n \Rightarrow d \mid n + d$.

   d. $\forall d, n, m, a, b \in \mathbb{Z}, d \mid n \wedge d \mid m \Rightarrow d \mid a \times n + b \times m$.