CSC108H Winter 2024 Worksheet 05 : Function Definitions

1. **Function Definitions**

   (a) Function `double` takes a number and returns twice its value. Finish the examples by writing the returned value. Then, write a `return` statement to complete the function definition:

```
def double(num: float) -> float:
    """Return twice the value of num.

    >>> double(7.0)
     14.0
    >>> double(5.7)
     11.4
    """
     return num * 2
```

   (b) Function `our_maximum` takes two numbers and returns the larger of the two. Finish the examples by writing the returned value. Then, complete the function definition:

```
def our_maximum(num1: float, num2: float) -> float:
    """Return the larger of num1 and num2.

    >>> our_maximum(1.5, 2.5)
     2.5
    >>> our_maximum(4.0, 3.7)
     4.0
    """
     return max(num1, num2)
```

   (c) Function `max_of_min` takes four values, `num1`, `num2`, `value1`, and `value2`, determines the minimum of `num1` and `num2`, the minimum of `value1` and `value2`, and returns the maximum of those minimums.

   What value does `max_of_min(4.0, 3.7, 6.0, 3.5)` produce? 3.7

   What value does `max_of_min(1.0, 1.7, 4.5, 3.0)` produce? 3.0

   Use your responses above to fill in two examples in the docstring. Then, complete the function definition. If you like, you can use several statements.

```
def max_of_min(num1: float, num2: float, value1: float, value2: float) -> float:
    """Return the maximum of the minimums of the pairs num1 and num2,
    and value1 and value2.

    >>> max_of_min(4.0, 3.7, 6.0, 3.5)
    3.7
    >>> max_of_min(1.0, 1.7, 4.5, 3.0)
    3.0
    """
    return max(min(num1, num2), min(value1, value2))
```

CSC108H Winter 2024 Worksheet 06 : Function Design Recipe

1. **Function Design Recipe**

   Following the Function Design Recipe, write a function that satisfies this description:

   This function returns a string containing a given word repeated a given number of times. For example, someone should be able to call the function to repeat `"Marcia "` three times and the function should return `"Marcia Marcia Marcia "`, or call the function to repeat `"Buffalo "` eight times and have it return `"Buffalo Buffalo Buffalo Buffalo Buffalo Buffalo Buffalo Buffalo "`.

   Examples

   ```
   """

   >>> repeat_word('Marcia ', 3)
   'Marcia Marcia Marcia '

   >>> repeat_word('Buffalo ', 8)
   'Buffalo Buffalo Buffalo Buffalo Buffalo Buffalo Buffalo Buffalo '
   """
   ```

   Header

   ```
   def repeat_word(word: str, repeat_count: int) -> str:
   ```

   Description

   ```
   Returns the string word repeated repeat_count times.

   Precondition: repeat_count >= 0
   ```

   Body

   ```
   return word * count
   ```

   Test

   ```
   >>> repeat_word('Nice ', 5)
   'Nice Nice Nice Nice Nice '
   ```

CSC108H Winter 2024 Worksheet 06 : Function Design Recipe

2. **Function Design Recipe: Type Contract and Description**

   Consider this code:

```
def number_of_cents(change:  float  ) ->  int    :
    """
     Returns the number of cents remaining from a total monetary amount after
     removing the whole dollar component from change.




    >>> number_of_cents(1.25)
    25
    >>> number_of_cents(20.00)
    0
    """

    dollars = change // 1             # alternate: dollars = int(change)
    cents = (change - dollars) * 100
    return round(cents)
```

   (a) Add the missing *Type Contract* (parameter and return types) to the function *Header*.

   (b) The description of this function is poor:

   `The number of cents left when you take away all the dollars from an amount of money.`

   In the space provided in the docstring, write a better function description.

3. **Function Design Recipe**

   Following the Function Design Recipe, write the following function.

Function name: **(parameter types) → return type**	**Description**
`calculate_tax`: `(float, float)` → `float`	The first parameter represents a bill and the second represents a tax rate (a number between 0.0 and 1.0 inclusive). Return the amount of tax to be paid on this bill.

```
def calculate_tax(bill: float, tax_rate: float) -> float
    """
    Returns the bill times the tax_rate to reflect the amount of tax to be paid on this
bill.

    Precondition: 0.0 <= tax_rate <= 1.0

    >>> calculate_tax(50.0, 0.13)
    6.5

    >>> calculate_tax(62.0, 0.10)
    6.2
    """
    return bill * tax_rate
```

CSC108H Winter 2024 Worksheet 07 : Nested Function Calls

Recall that for a function call the arguments are evaluated left to right, and only then is the function call executed.
As an example, we have underlined parts of the expression below and numbered them to indicate the order in which the subexpressions are evaluated.

```
f(g() + 3, h())
  ---   -
   1    2
  -------  ---
     3      4
---------------
       5
```

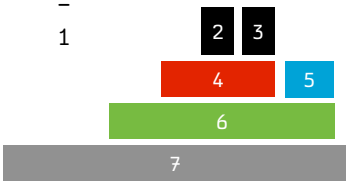
That says that function call `g()` is evaluated first, then the `3`, then the result of the call on `g` is added to `3`, then function call `h()` happens, and finally (now that the arguments to `f` have been evaluated) the call on `f` happens.

1. In function `max_of_min` below, underline and number the subexpressions in the `return` expression (the one starting with `max(min...)`) to indicate the order in which the subexpressions are evaluated. We've done the first two for you. (The function's type contract has been omitted to save space.)

```
def max_of_min(num1, num2, value1, value2):
    return max(min(num1, num2), min(value1, value2))
                   ----  ----
                     1     2
```

2. In the following expression, underline and number the subexpressions to indicate the order in which they are evaluated. We've done the first one for you.

```
pow(2, pow(pow(2, 1), 2))
    -
    1
```



CSC108H Winter 2024 Worksheet 08 : Function Reuse

Often, you will call one function from within another function definition.

1. Following the Function Design Recipe, write the following function.

Function name: (parameter types) → return type	**Description**
`format_name`: `(str, str)` → `str`	The first parameter represents a first name and the second represents a last name. Return a string in the format: `last_name, first_name`, where `last_name` and `first_name` are replaced by the given last and first names.

```
def format_name(first_name: str, last_name: str) -> str:
    """
    Returns a string in the format of last_name, first_name.

    >>> format_name('Devansh', 'Gandhi')
    'Gandhi, Devansh'

    >>> format_name('Jen', 'Campbell')
    'Campbell, Jen'
    """
    return last_name + ", " + first_name
```

2. Following the Function Design Recipe, write the following function to produce a telephone directory listing. In the function body, call `format_name` to have it do some of your work for you.

Function name: (parameter types) → return type	**Description**
`to_listing`: `(str, str, str)` → `str`	The first parameter represents a first name, the second represents a last name, and the third represents a phone number. Return a string in the format: `last_name, first_name: phone_number`, where `last_name`, `first_name`, and `phone_number` are replaced by the given last name, first name, and phone number.

```
def to_listing(first_name: str, last_name: str, phone_number: str) -> str:
    """
    Returns a string in the format of last_name, first_name: phone number.

    >>> to_listing('Devansh', 'Gandhi','555')
    'Gandhi, Devansh: 555'

    >>> to_listing('Jen', 'Campbell', '111')
    'Campbell, Jen: 111'
    """
    return last_name + ", " + first_name, + ": " +phone_number
```