Lab 04 will be up later today
  – Practice designing functions and writing doc strings

Next week starting control structures
  – Slides, reading and pre-reading quiz will be up later today
Docstrings

• Let's now provide the right documentation (docstring) for the income_tax function:

```python
def income_tax(income, rate):
    """(float, float) -> float
    income >= 9000.0; 0.0 <= rate < 1.0
    Returns the income tax for the given income and tax rate.
    income >= 9000.0. rate is in decimal form between 0 and 1
    i.e. 0.3 means 30%
    >>> income_tax(9000.0, 0.2)
    0.0
    >>> income_tax(9000.0, 0.0)
    0.0
    >>> income_tax(10000.0, 0.3)
    300.0
    ""

    exemption = 9000.00
    net_income = income - exemption
    return net_income * rate
```

• The docstring is mainly for human users. We normally write this BEFORE we write the code for the function

• The command `help(income_tax)` displays the docstring

• The test code inside the docstring is **not** part of the function and will not be executed when the function is called
Function Design Recipe

1. Determine the type contract
   – What are the types of the input and output data?
   – Are there any preconditions on the input values?
2. Write the header
   – Pick meaningful names for the function and parameters
3. Write the description
   – Make sure to include all your parameters in your description
4. Write some examples
   – Pick examples which are as distinct as possible
   – Try to test the limits of what your function is supposed to handle
5. Write the body
   – Pick meaningful names for the local variables
6. Test your function with your examples
   – Better to check now whether your code has a bug
   – Add more examples if you think of some
Example: Room Painting Cost, Version 3

# Functions

```python
def wall_area(length, width, height):
    '''
    (float, float, float) -> float
    length, width, height are >= 0

    Calculates the wall area of a room with the given length, width and
    height.
    
    >>> wall_area(0, 0, 0)
    0.0
    >>> wall_area(10, 10, 10)
    400.0
    '''
    wall_1 = length * height
    wall_2 = width * height
    return 2 * wall_1 + 2 * wall_2

def paint_cost(area, paint_price):
    '''
    (float, float) -> float
    area and paint_price are >= 0

    Calculates the cost for the paint needed to paint the given area.
    paint_price is the price of one gallon of paint.
    
    >>> paint_cost(0, 0)
    0.0
    >>> paint_cost(50, 10)
    10.0
    >>> paint_cost(200, 10)
    40.0
    '''
    return area / 50 * paint_price

def labour_cost(area, labour_price):
    '''
    (float, float) -> float
    area and labour_price are >= 0

    Calculates the labour cost for painting the given area
    labour_price is the labour cost per hour.
    
    >>> labour_cost(0, 0)
    0.0
    >>> labour_cost(50, 10)
    10.0
    >>> labour_cost(200, 10)
    40.0
    '''
    return area / 50 * labour_price
```

Example: Room Painting Cost, Version 3 (cont')

# Main program

cost_for_paint = paint_cost(total_area, paint_price)
print( "Paint cost is:", cost_for_paint)

# Calculate the cost of paint needed

cost_for_labour = labour_cost(total_area, labour_price)
print( "Labour cost is:", cost_for_labour)

# Calculate labour cost

total_cost = cost_for_paint + cost_for_labour
print( "Total cost is:", total_cost)
print()

print( "Bye now. Hope to hear from you soon")

# Get room dimentions and estimate the cost.

room_length = float(input("Enter room's length : "))
room_width = float(input("Enter room's width: "))
room_height = float(input("Enter room's height: "))

# Total area for painting

total_area = wall_area(room_length, room_width, room_height)
print( "Total area is :", total_area)
Default Parameters & Keyword Arguments

• You may provide default values for function parameters in the header line:

```python
def paint_cost(length, width, height = 8,
               paint_price = 45, labour_price = 25):
    ...
```

– Allows you to design flexible functions with simple interfaces
– Default is evaluated only once (which will matter for mutable data)
– Parameters with defaults must appear at the end of the parameter list
– Having default parameter values created optional arguments

• You may provide function arguments out of order using keyword arguments:

```python
print('total:', paint_cost(10, 12, paint_price = 49.99))
```

– Keyword arguments must appear at the end of the argument list
– You may provide keyword arguments for any parameter (whether or not it has a default value)
– Each parameter must be given one value
Recap of Key Terminology

- **Function call**: tells python to execute a function
- **Parameters**: the variable(s) that appear in the parentheses in of the function header
- **Arguments**: the expression (values, variable, function) that you pass to the function when you make a function call
  - These map to the parameters in the function definition
- **Local variable**: used in the function definition for temporary storage. Created when a function is called, and erased when the function returns.
Conclusion

• A large task can be split into subtasks using functions
• A new function is created by the command `def`
  – The function body is the subsequent indented code
• To document a new function, use docstrings and the function design recipe
• Each function can have parameters and local variables that are only accessible inside the function's code
• To execute a function we write a function call that is made of
  – the function name
  – a value for each parameter
• When a function is called, the function code is executed in its own environment (frame)
• When the function return its environment is destroyed
• Python comes with a large number of built-in functions (some of which we'll see later)
Change Log

• Added some additional examples of built in functions (min and max)
• Added slide on evaluation order
• In Function Call slide:
  – added the initial placing of income_tax function name on the main program frame when the code first runs to remind you that the functions are loaded on to the frame first, before you can call them.
• In the doc string slide:
  – fixed an inconsistency in the preconditions listed.
• Removed the ‘return area / 50 * paint_price’ line at the very end of the slide showing paint cost v3