CMPE 102
Programming Logic and Design

Module 7 – Part II
Python's User-defined Functions

Outline

- Local and global variables
- Multiple returned values
- Calling functions with positional and named arguments

Circle Area – Revisited

```
1: def compute_circle_area(radius):
2:     circle_area = math.pi*radius**2
3:     return circle_area
4:
5: r = float(input("Enter a radius: "))
6: area = compute_circle_area(r)
7: print(f"Area of the circle is {area:.2f}")
```

Circle Area – Revisited

```
1: def compute_circle_area(radius):
2:     circle_area = math.pi*radius**2
3:     return circle_area
4:
5: r = float(input("Enter a radius: "))
6: area = compute_circle_area(r)
7: print(f"Area of the circle is {area:.2f}")
```

Let's try adding one more line to the above program

What will happen?

```
8: print(circle_area)
```

```
>>> print(circle_area)
NameError: name 'circle_area' is not defined
```

Why?

Circle Area – Revisited

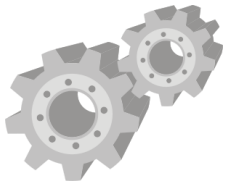
```
1: def compute_circle_area(radius):
2:     circle_area = math.pi*radius**2
3:     return circle_area
4:
5: r = float(input("Enter a radius: "))
6: area = compute_circle_area(r)
7: print(f"Area of the circle is {area:.2f}")
```

circle_area is only ***locally*** known to the function compute_circle_area()

```
8: print(circle_area)
```

```
>>> print(circle_area)
NameError: name 'circle_area' is not defined
```

Local vs. Global Variables



- In Python, a variable defined inside a function can only be used inside that function
  - **x** at ❶ is called a ***local variable*** of `function1()`
  - **x** at ❷ is called a ***global variable***
  - These two **x**'s are different variables

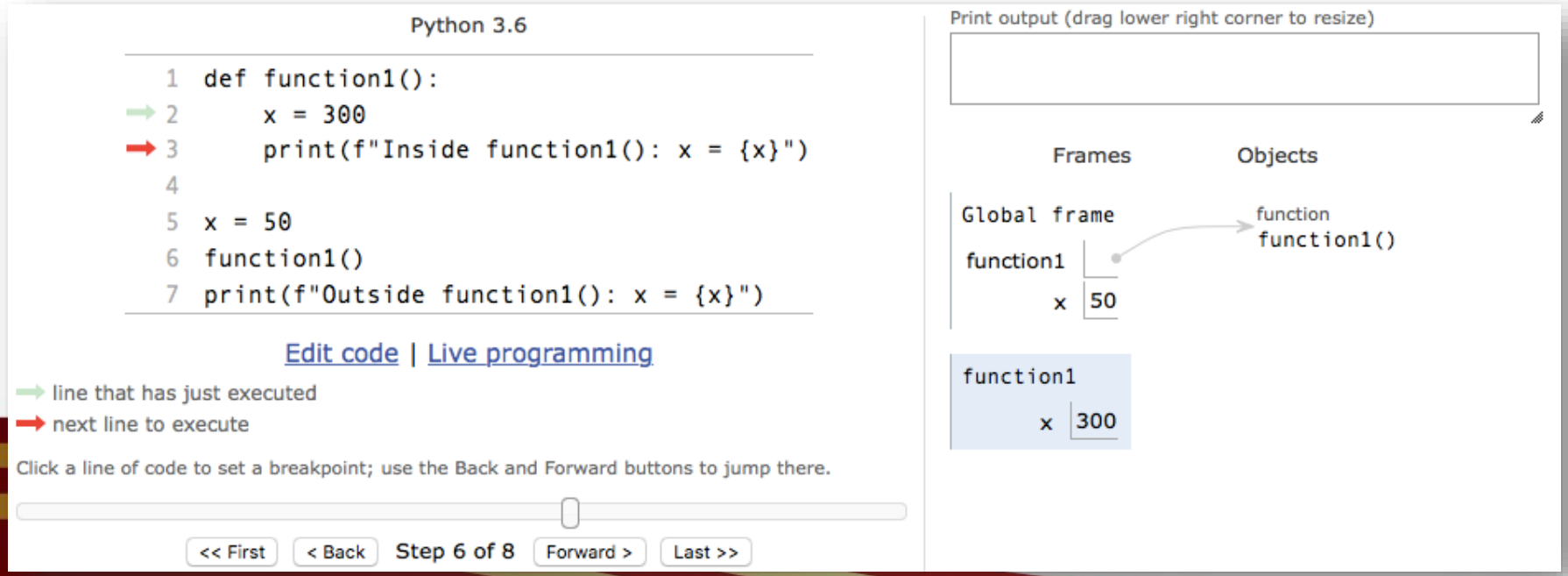
```
def function1():
    x = 300                                  ❶
    print(f"Inside function1(): x = {x}")

x = 50                                       ❷
function1()
print(f"Outside function1(): x = {x}")
```

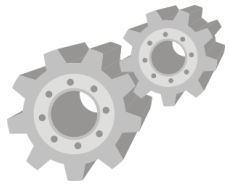
```
Inside function1(): x=300
Outside function1(): x=50
```

Try it on pythontutor.com

- The web http://pythontutor.com provides excellent visualization tool for code execution
- Click "Start visualizing your code now" and paste the code from the example page in the box



Local vs. Global Variables



- A variable referenced, but not defined, inside a function is considered a global variable
  - However, these variables are ***read-only*** by default

```
def function1():
    print(f"Inside function1(): x = {x}")

x = 50
function1()
x = 80
function1()
```

This x is not assigned inside function1() before.

- Again, try it on pythontutor.com!

```
Inside function1(): x=50
Inside function1(): x=80
```

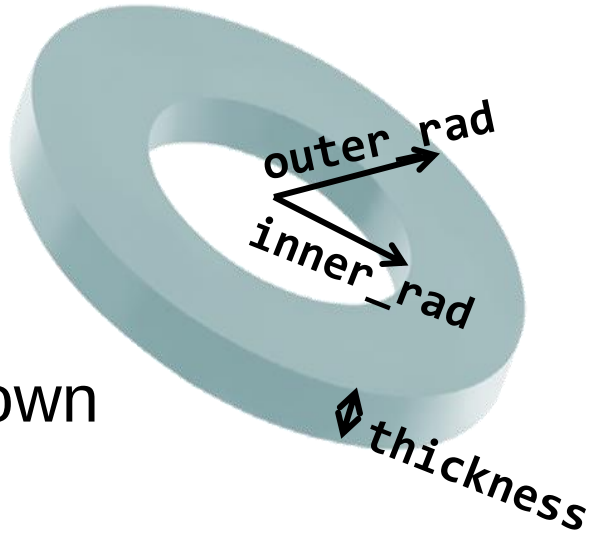
Task: Flat Washers

- You work for a hardware company that manufactures flat washers. To estimate shipping costs, your company needs a program that computes the weight of a specified quality of flat washers.

https://commons.wikimedia.org/wiki/File%3AWashers.agr.jpg

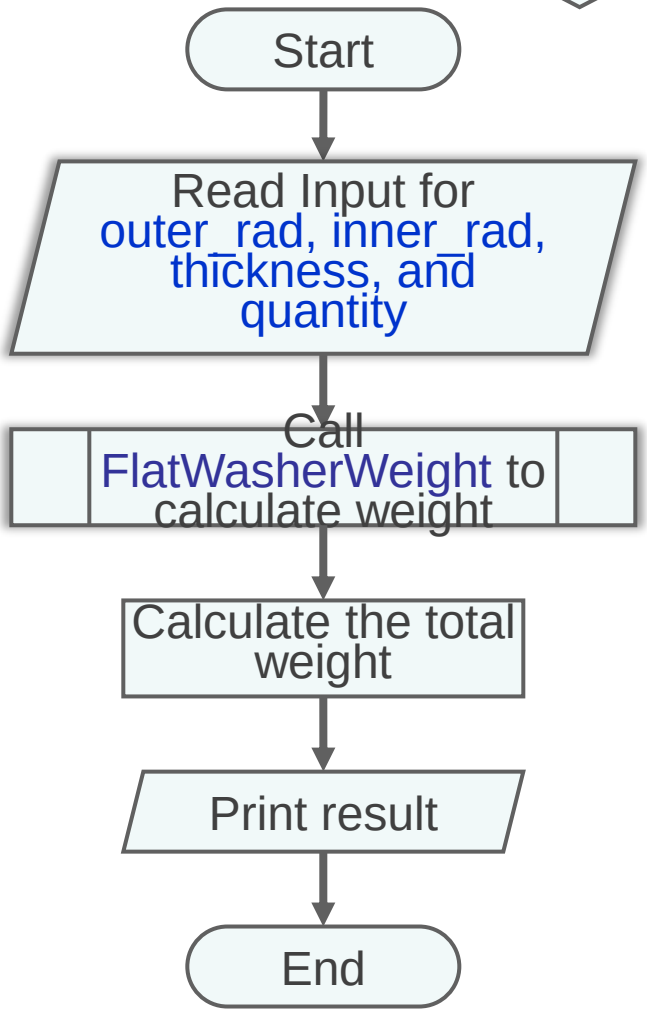
Flat Washers - Ideas

- A flat washer resembles a small donut (see the figure).
- To compute the weight of a single flat washer, you need to know its *rim area*, *thickness*, and *density* of the material
  - Here, we can **reuse** **`compute_circle_area()`** function
- Requirements:
  - Radius of flat washer and hole
  - Thickness
  - Density
  - Quantity
- We will assume that the material used is aluminum, whose density is well-known



Flat Washers – Steps

- Get the washer's outer radius, inner radius, thickness, and quantity
- Compute the weight of one flat washer
  - unit_weight = rim_area × thickness × density
- Compute the weight of batch of washers
  - total_weight = unit_weight × quantity
- Print the resulting weight of batch



Flat Washer

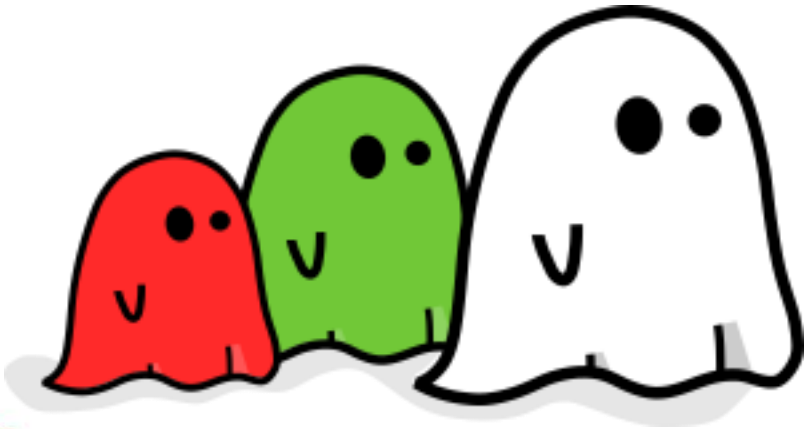
```
Enter the outer radius (cm.): 15
Enter inner radius (cm.): 10
Enter thickness (cm.): 3
Enter the quantity (pieces): 10
Weight of the batch is 31808.63 grams
```

Notice how the variable MATERIAL_DENSITY is defined and used as a global variable

```
1:  import math
2:
3:  MATERIAL_DENSITY = 2.70
4:
5:  def compute_circle_area(radius):
6:      return math.pi*radius**2;
7:
8:  def flat_washer_weight(outer_r,inner_r,thickness):
        rim_area=compute_circle_area(outer_r)-compute_circle_area(inner_r)
9:      return rim_area*thickness*MATERIAL_DENSITY
10:
11: outer_rad = float(input('Enter the outer radius (cm.): '))
12: inner_rad = float(input('Enter inner radius (cm.): '))
13: thickness = float(input('Enter thickness (cm.): '))
14: quantity = int(input('Enter the quantity (pieces): '))
15: unit_weight = flat_washer_weight(outer_rad,inner_rad,thickness)
16: total_weight = unit_weight * quantity
17: print(f'Weight of the batch is {total_weight:.2f} grams')
```

Task: Average of Three

- Program will ask three integer input values from the user, calculate the average of those three values, and then print the result to screen.

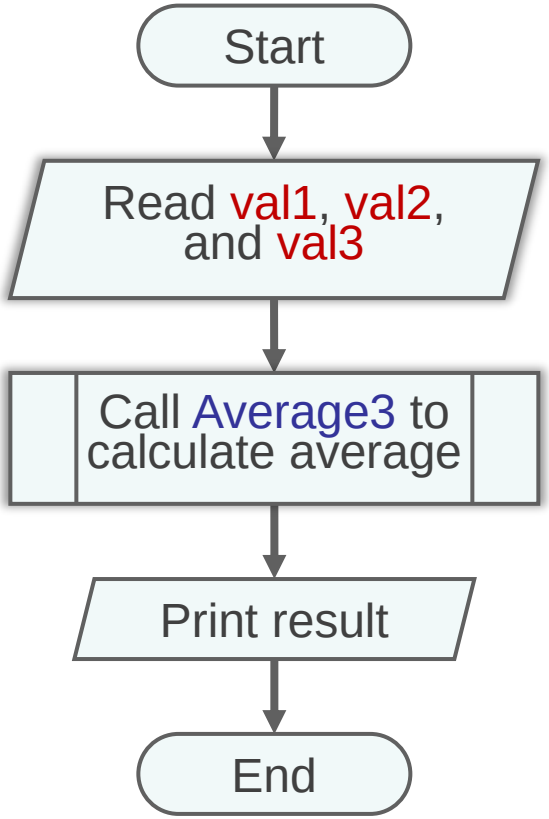


Average of Three - Ideas

- Need to know the three integer values, i.e., val1, val2, val3
- Compute the average
  - average = (val1 + val2 + val3)/3
- Show the result to screen

Average of Three - Steps

- Get input three input integer values from the user
- Calculate the average
  - average = (val1 + val2 + val3)/3
- Print the resulting average

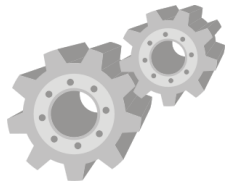


Average of Three – Program#1

```
 1: def average3(x, y, z):
 2:       return (x+y+z)/3;
 3:
 4: # read three integers
 5: val1 = int(input('1st value: '))
 6: val2 = int(input('2nd value: '))
 7: val3 = int(input('3rd value: '))
 8:
 9: # compute and output their average
10: average = average3(val1, val2, val3)
11: print(f'average is {average:.4f}')
```

```
1st value: 15
2nd value: 20
3rd value: 25
average is 20.0000
```

Returning Multiple Values



- A function can return multiple values by separating them by comma sign
  - Values must be assigned the same number as the return values

```
def Read3Integers():
    ...
    return x, y, z

val1, val2, val3 = Read3Integers()
```

send to

send to

send to

Average of Three – Program#2

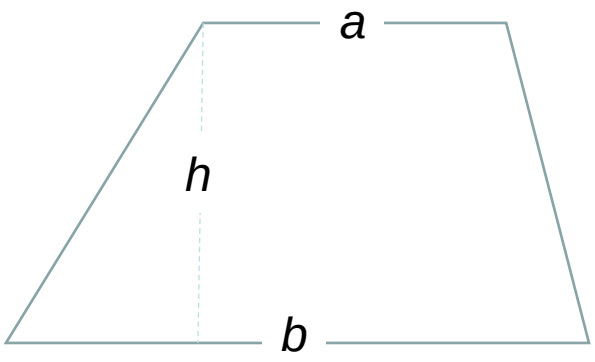
```
1st value: 15
2nd value: 20
3rd value: 25
average is 20.0000
```

```
1:  def read_3integers():
2:      # read three integers
3:      a1 = int(input("1st value: "))
4:      a2 = int(input("2nd value: "))
5:      a3 = int(input("3rd value: "))
6:      return a1, a2, a3
7:
8:  def average3(x, y, z):
9:      return (x+y+z)/3
10:
11: val1, val2, val3 = read_3integers()
12: # compute and output their average
13: print(f"average is {average3(val1, val2, val3):.4f}")
```

Task: Trapezoid

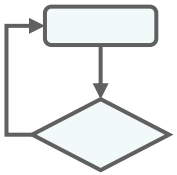
- In Euclidean geometry, a convex quadrilateral with at least one pair of parallel sides is referred to as a **trapezoid.**

(ref: https://en.wikipedia.org/wiki/Trapezoid)

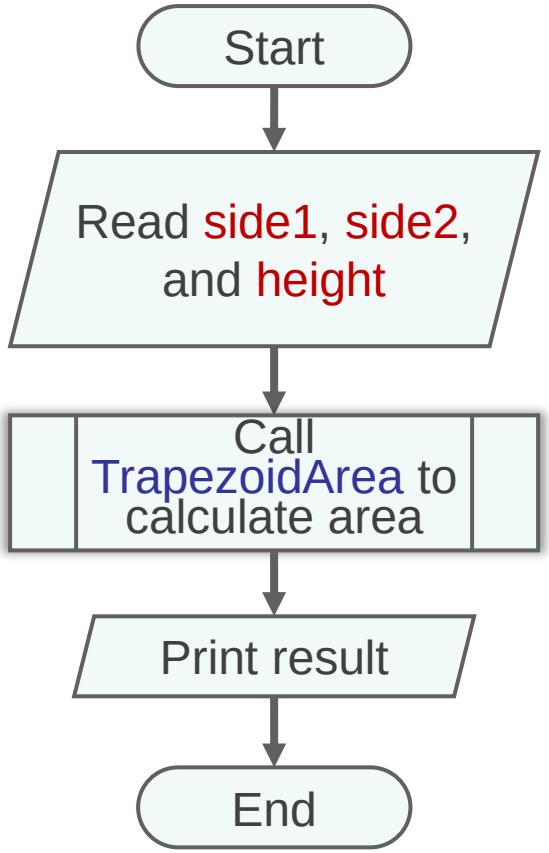


$$area = \frac{a + b}{2} \; h$$

Trapezoid – Steps



- Get three double values from the user:
  - (parallel) side1
  - (parallel) side2
  - height
- Calculate the trapezoid area
  - area = ((side1 + side2)/2) × height
- Print the resulting area



Trapezo

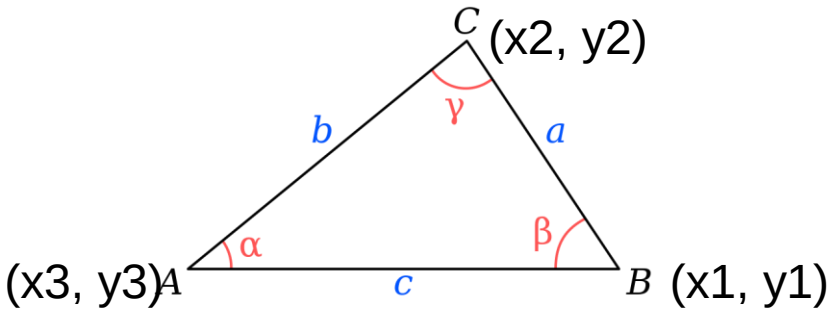
```
Enter the properties of your trapezoid.
Length of parallel side 1: 10
Length of parallel side 2: 15
Height: 13
Trapezoid's area is 162.50
```

```
1:  def read_trapezoid():
2:      print("Enter the properties of your trapezoid.")
3:      a = float(input("Length of parallel side 1: "))
4:      b = float(input("Length of parallel side 2: "))
5:      h = float(input("Height: "))
6:      return a,b,h
7:
8:  def trapezoid_area(a,b,h):
9:      return 0.5*(a+b)*h
10:
11: # main program
12: a,b,h = read_trapezoid()
13: area = trapezoid_area(a,b,h)
14: print(f"Trapezoid's area is {area:.2f}")
```

Task: Triangle Area (Heron)

- In geometry, **Heron's formula** (sometimes called Hero's formula), named after Hero of Alexandria, gives the area of a triangle by requiring no arbitrary choice of side as base or vertex as origin, contrary to other formulas for the area of a triangle, such as half the base times the height or half the norm of a cross product of two sides.

(ref: https://en.wikipedia.org/wiki/Heron's_formula)



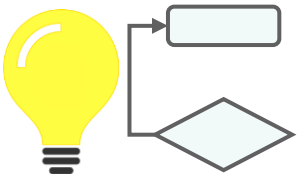
- Heron's formula states that the area of a triangle whose sides have lengths *a*, *b*, and *c* is

$$area = \sqrt{s(s-a)(s-b)(s-c)}\ ,$$

where *s* is the semiperimeter of the triangle; that is,

$$s = \frac{a+b+c}{2}$$

Triangle Area (Heron) - Ideas + Step



- Get the x-y coordinate of the triangle's 3 vertices
- Calculate the length of the lines *a*, *b*, and *c* which are connected to those 3 vertices
- Calculate the semiperimeter
- Calculate the triangle's area using the Heron's formula
- Print the resulting area

Triangle Area (Heron) - Program

```
 1: import math
 2:
 3: def line_length(x1, y1, x2, y2):
 4:     """
 5:     Given X-Y coordiates of 2 points, compute the line length that
 6:     joins them
 7:     """
 8:     return math.sqrt((x1-x2)**2+(y1-y2)**2);
 9:
10: def triangle_area(x1, y1, x2, y2, x3, y3):
11:     """
12:     Given the 3 vertices, compute triangle area using Heron's Formula
13:     """
14:     a = line_length(x1, y1, x2, y2)
15:     b = line_length(x2, y2, x3, y3)
16:     c = line_length(x3, y3, x1, y1)
17:     s = (a+b+c)/2
18:     return math.sqrt(s*(s-a)*(s-b)*(s-c))
```

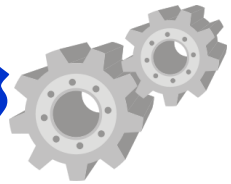
(The code continues on the next page)

Triangle Area (

```
Enter X-Y coordinates of the three vertices of triangle:
1st vertex:
x? 1
y? 1
2nd vertex:
x? 3
y? 2
3rd vertex:
x? 2
y? 3
area of the triangle is 1.50
```

```
19: def read_coordinates():
20:     x = float(input("x? "))
21:     y = float(input("y? "))
22:     return x,y
23:
24: def read_triangle():
25:     """
26:     Read X-Y co-ordinates of 3 vertices of a triangle
27:     """
28:     print("Enter X-Y coordinates of the three vertices of triangle:")
29:     print("1st vertex:")
30:     x1,y1 = read_coordinates()
31:     print("2nd vertex:")
32:     x2,y2 = read_coordinates()
33:     print("3rd vertex:")
34:     x3,y3 = read_coordinates()
35:     return x1,y1,x2,y2,x3,y3
36:
37: x1,y1,x2,y2,x3,y3 = read_triangle()
38: area = triangle_area(x1,y1,x2,y2,x3,y3)
39: print(f"area of the triangle is {area:.2f}")
```

Positional & Named Arguments



- When you call a function, you need to know the parameters that the function take, i.e. the number of arguments as well as the order
  - In addition, you may need to know the unit, i.e. `sin()/cos()` use radians, not degrees
- Don't remember? No problem—use help
  - Still remember about Docstring?
- So far, when we call a function, arguments are arranged in the order according to the parameters—***positional arguments***

Trapezoid - Recall

```
def trapezoid_area(a, b, h):
        return 0.5*(a+b)*h;
```

- The above function is currently called as
  - Notice that the positions of arguments match the positions of parameters —***positional arguments***

```
area = trapezoid_area(side1,side2,height)
```

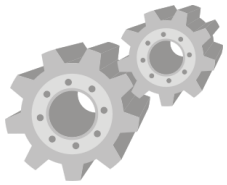
- Named arguments can be used so that positions do not need to match

```
area = trapezoid_area(h=height,a=side1,b=side2)
```

Conclusion

- Local variables are known only within the function definition
- Global variables are known throughout the program, but are read only unless keyword ***`global`*** is used
- Functions can return multiple values and therefore should be assigned accordingly
- Arguments of a function can either be positional or named

Syntax Summary

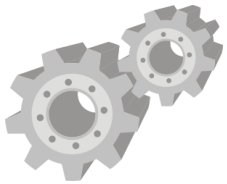


- Returning multiple values from functions

```
def function_name()
      ...
      ...
      return val1, val2, ..., valn

v1, v2, ..., vn = function_name()
```

Syntax Summary



- Positional arguments, i.e. *val1* corresponds to *arg1*, …

```
function_name(val1, val2, ..., valn)
```

- Named arguments

```
function_name(argn=valn, arg1=val1, ...)
```

References

- Python standard library https://docs.python.org/3/library/index.html
- Keyword (named) arguments in Python https://docs.python.org/3/tutorial/controlflow.html#keyword-arguments