




LECTURE 3: FUNCTIONS

Hung-Yu Wei

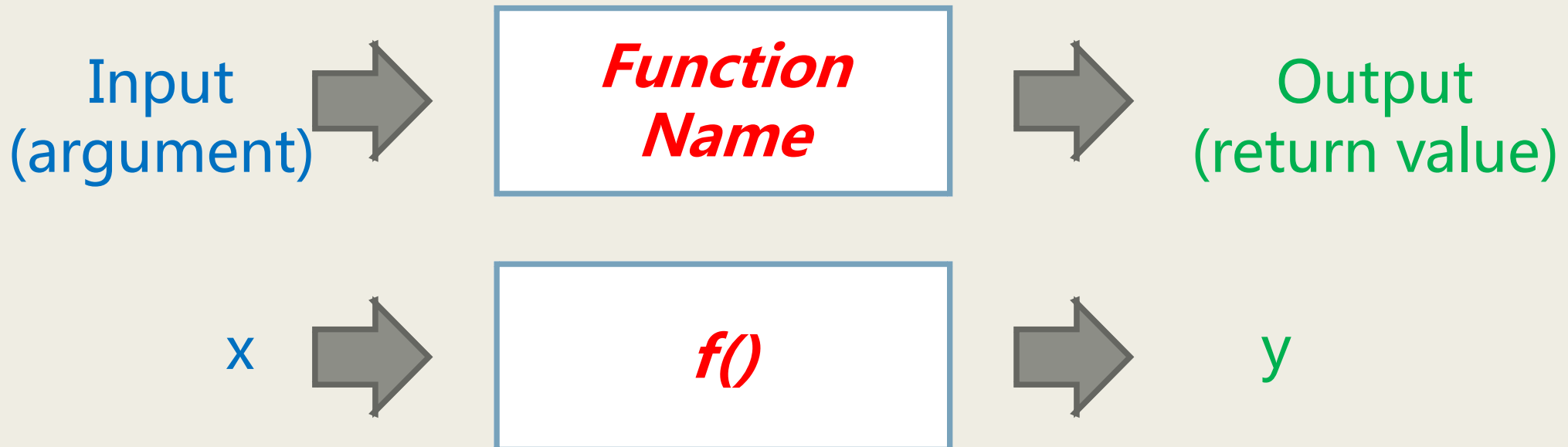


What's function?

- Do you remember functions in your math class?

$\sin(x)$, $\cos(x)$, $\tan(x)$

$$y = f(x)$$



Function calls

- Call (invoke) a function
 - *Execute this function*
- A function might include input and/or output
 - *Input: **argument***
 - *Output: **return value***
- A function could be
 - *Pre-defined in standard library*
 - *Defined by you*

Review: types of variables

- Number
 - *Integer*
 - *Floating point*
 - *Boolean (True, or False)*
- String

```
1 number_Student = 62           # an integer variable
2 width_cm      = 165.2         # a floating-point variable
3 power_OnOff   = True          # a boolean variable
4 name_Leading_Actor = 'John Smith' # a string
5
6
7 print (number_Student)
8 print (width_cm)
9 print (power_OnOff)
10 print (name_Leading_Actor)
```

```
62
165.2
True
John Smith
```

Examples of function calls

■ `type()`

- A function to show the type of input

```
1 type(42)
int
1 int('32')
32
```

■ `int()`

- A function to convert the input variable into integer
 - It will get rid of the fraction parts

```
1 int(3.9999)
3
1 int(-2.3)
-2
```

```
1 int('Hello')

-----
ValueError                                Traceback (most recent call last)
<ipython-input-4-6765ce49acfe> in <module>()
----> 1 int('Hello')

ValueError: invalid literal for int() with base 10: 'Hello'
```

More examples of function calls

■ **float()**

- A function to convert the input into floating-point

■ **str()**

- A function to convert the input into string

```
1 float(32)
32.0
1 float('3.14159')
3.14159
```

```
1 str(32)
'32'
1 str(3.14159)
'3.14159'
```

Math module and functions

- **module**: a collection of functions
 - *a function library*
 - *Many programming languages have built-in standard libraries*
- **import** statement
 - *Import a library before using it*
- **dot** notation
 - *Call a function within a module*
- Example
 - ***math** module*
 - ***import math** before using any function in this module*
 - ***log10** is a function defined in **math** module*
 - *To use the logarithm function, we use **dot** .*
math.log10

```
1 import math
2 math
```

```
<module 'math' (built-in)>
```

```
1 math.log10(100)
```

```
2.0
```

More about “**math**” functions

- Logarithm
 - ***math.log10()***
 - Example:
 - calculating decibel
- Sine
 - ***math.sin()***
- π
 - ***math.pi***

```
1 signal_power = 0.2      # signal is 0.2 Watt
2 noise_power  = 2e-5     # noise is 2e-5 or 2*10**-5
3 SNR_ratio=signal_power/noise_power
4 print(SNR_ratio)
5 SNR_dB = 10*math.log10(SNR_ratio)
6 print(SNR_dB)
```

```
10000.0
40.0
```

- Square root \sqrt{x}
 - ***math.sqrt()***
- Exponential e^x
 - ***math.exp()***

```
1 math.sqrt(2)
```

```
1.4142135623730951
```

```
1 math.exp(2)
```

```
7.38905609893065
```

- Remember to
import math
- More math functions
 - <https://docs.python.org/3/library/math.html>

```
1 radians = 0.7
2 height = math.sin(radians)
3 print(height)
4 degrees = 30
5 radians = degrees/180.0*math.pi
6 print(math.sin(radians))
```

```
0.644217687237691
0.49999999999999994
```


Composition

- Use (compose) multiple building blocks

– $e^{\log(y+1)} = y + 1$

```
1 degrees=90
2 x=math.sin(degrees/360.0*2*math.pi)
3 print(x)
4 y=3
5 y=math.exp(math.log(y+1))
6 print(y)
```

```
1.0
4.0
```

```
1 hours=3
2 minutes=hours*60
3 print(minutes)
```

```
180
```

```
1 hours*60=minutes
```

```
File "<ipython-input-24-d6e468c3fc3d>", line 1
    hours*60=minutes
            ^
```

```
SyntaxError: can't assign to operator
```

Create new functions

- I can use built-in functions. However, I also want to create my own functions.
 - *Flexibility in defining new components*
- Syntax: **define** a new function

def function_name() :

... some operations in your function ...

| | | |
|---------------------------------|--|--------------------------------------|
| 1 | def print_lyrics() : | Header of function definition |
| 2 | print("I like to learn programming !!!") | |
| 3 | print("I love Python ^__^ ") | |
| 4 | | Body of function definition |
| <hr/> | | |
| 1 | print_lyrics() | Call your function |
| I like to learn programming !!! | | |
| I love Python ^__^ | | |

- Tips: remember ":" and "()"

A function that calls another function

```
1 def print_lyrics():
2     print("I like to learn programming !!!")
3     print("I love Python ^__^ ")
4
```

define the 1st function

```
1 def repeat_lyrics():
2     print_lyrics()
3     print_lyrics()
4     print("I just define a repeating function ...")
5
6 repeat_lyrics()
```

**define the 2nd function that
uses the 1st function**

```
I like to learn programming !!!
I love Python ^__^
I like to learn programming !!!
I love Python ^__^
I just define a repeating function ...
```

Argument and parameter

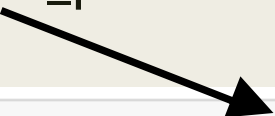
- **Parameter**: local variable to handle the input (e.g. my_cool_parameter)

*def myFunc(**parameter_in_myFunc**)*

.....

- **Argument**: input of a function

*myFunc(**I_am_argument**)*



```
1 def print_twice(my_cool_parameter):  
2     print(my_cool_parameter)  
3     print(my_cool_parameter)  
4  
5 print_twice('This is a test. ')  
6 print_twice('Spam')  
7 print_twice(42)
```

```
This is a test.  
This is a test.  
Spam  
Spam  
42  
42  
3.141592653589793  
3.141592653589793  
-1.0  
-1.0
```

```
1 import math
2 print_twice(math.pi)
3 print_twice(math.cos(math.pi))
```

```
3.141592653589793
3.141592653589793
-1.0
-1.0
```

```
1 some_text="Today is Tuesday."
2 print_twice(some_text)
```

```
Today is Tuesday.
Today is Tuesday.
```

Clarification on Argument and Parameter

- Parameter --- **k**
 - *This is **local***
 - *Only valid within function `print_Three`*
 - Local parameter will be destroyed at the end of function
- Function Argument -- **m**

```
1 def print_Three(k):  
2     print(k)  
3     print(k)  
4     print(k)  
5  
6 m=10  
7 print_Three(m)  
  
10  
10  
10
```



Local variable and parameter

- Local variable
 - *cat*
- Local parameter
 - *part1, part2*

```
1 def cat_Three(part1,part2):  
2     cat = part1+part2  
3     print_Three(cat)  
4  
5     line1="This is "  
6     line2="a fun computer programming class!"  
7  
8     cat_Three(line1,line2)
```

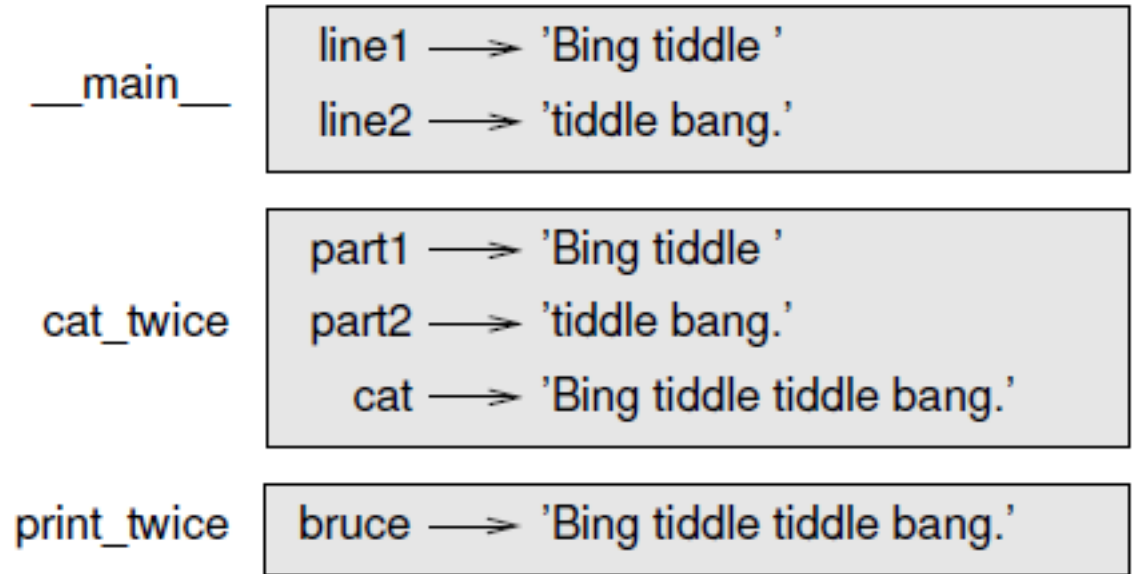
```
This is a fun computer programming class!  
This is a fun computer programming class!  
This is a fun computer programming class!
```

Stack Diagram

- Scope of variables

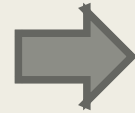
```
1 def print_twice(bruce):  
2     print(bruce)  
3     print(bruce)  
4  
5 def cat_twice(part1, part2):  
6     cat = part1 + part2  
7     print_twice(cat)  
8  
9 line1 = 'Bing tiddle '  
10 line2 = 'tiddle bang.'  
11 cat_twice(line1, line2)
```

Bing tiddle tiddle bang.
Bing tiddle tiddle bang.



Functions With/Without Return

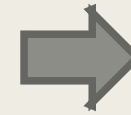
- Function with return
 - *Fruitful function*



***Function
Name***



***Function
Name***



- Function without return
 - *Void function*



***Function
Name***

***Function
Name***

Void function and None

■ None

- *A special value for void function return*
- *Different from string 'None'*

```
1 def print_twice(bruce):  
2     print(bruce)  
3     print(bruce)  
4  
5 result=print_twice('Bing')  
6 print(result)
```

```
Bing  
Bing  
None
```

```
1 type(None)
```

```
NoneType
```

```
1 type('None')
```

```
str
```

Benefits of using function

- Clarity and readability
 - *Creating a new function gives you an opportunity to name a group of statements, which makes your program **easier to read and debug**.*
- Eliminate duplication
 - *Functions can make a program smaller by **eliminating repetitive code**. Later, if you make a change, you only have to make it in one place.*
- Divide into smaller building blocks (make sure each small block is OK)
 - *Dividing a long program into functions allows you to **debug the parts one at a time** and then assemble them into a working whole.*
- Reusability
 - *Well-designed functions are often **useful for many programs**. Once you write and debug one, you can reuse it.*

Summary

- Function
 - *Use functions in standard library*
 - Import
 - *Define your function*
 - `def`

Reading

- Chapter 3 in textbook “Think Python”