Functions

April 23, 2020

1 Functions

2 Functions

- Functions in programming languages are **self-contained** "mini-programs" that are dedicated for a specific task
- Use in two steps: **Define** and **Call**

3 Function - Define and Call

- Similar to math functions, but in Python sytax rather than math formula.
  - Math function
    - Define: y = f(a,b,c) = a + b + c
    - Call: y = f(3,2,5) = 3 + 2 + 5 = 10
  - Python function

[2]:
```
# define: def + function_name + arguments (a, b, c)
def return_sum_of_numbers(a, b, c):
    """
    Summary Line: A function that adds three numbers and return the result
    Extended description: so simple. Nothing to explain more:)
    Arguments:
        a, b, c(int or float):three numbers to add
    Returns:
        int or float: the result of adding three numbers
    """
    result = a + b + c
    return result

y = return_sum_of_numbers(3,2,5); print(2 * y)# call: function_name(inputs:␣
 ↪3 >a,   >b, 5 >c). Nothing happens w/o calling
print(2 * return_sum_of_numbers(3,2,5))# returned value can be used as a␣
 ↪variable to pass in to another fuction as inputs
help(return_sum_of_numbers)# documentation string
```

```
20
20
Help on function return_sum_of_numbers in module __main__:

return_sum_of_numbers(a, b, c)
    Summary Line: A function that adds three numbers and return the result
    Extended description: so simple. Nothing to explain more:)
    Arguments:
        a, b, c(int or float):three numbers to add
    Returns:
        int or float: the result of adding three numbers
```

4 Student Exercise - Function

- Rewrite the program for Fibonacci series by creating a function that writes the Fibonacci series to an **arbitrary** boundary.

[3]:
```
a = 0; b = 1; # set up base
while b < 20:  # condition
    print(b, end=' '); c = a + b; a = b; b = c
```

[4]:
```
def fib(n): # Define function   write Fibonacci series up to n
    """Print a Fibonacci series up to n."""# Documentation string
    a = 0; b = 1; # set up base
    while b < n:  # condition
        print(b, end=' '); c = a + b; a = b; b = c

fib(20)# Now call the function we just defined
fib(200)# Call the function whenever needed w/o rewriting the code
print(fib(20))
```

```
1 1 2 3 5 8 13 1 1 2 3 5 8 13 21 34 55 89 144 1 1 2 3 5 8 13 None
```

5 Function - `return`

- Explicit `return` is optional
- If no explicit return, funtion return a value `None`

6 Student Exercise - Function with explicit return

Rewrite the function for Fibonacci series by returning a list of numbers of the Fibonacci series

[18]:
```
def fib(n): # Define function   write Fibonacci series up to n
    """Return a list containing a Fibonacci series up to n."""# Documentation␣
 ↪string
    series = []#create an emply list for holding the series
    a = 0; b = 1; # set up base
    while b < n:  # condition
        series.append(b); c = a + b; a = b; b = c
    return series

print(fib(20))# Now call the function we just defined
```

```
[1, 1, 2, 3, 5, 8, 13]
```

7 Built-in Functions

- Built-in Functions are already defined by Python, thus call directly
- A complete list of built-in functions: https://docs.python.org/3/library/functions.html
- Getting help - `help()`
- Revisiting a more powerful `print()`
- Manipulating data types - `type()`, `int()`, `float()` and `str()`
- Getting user input - `input()`

8 Built-in Functions - `help()`

- Get information for any built-in or user defined functions, classes, modules etc
- Information is built on **documentation strings**

[6]:
```
help(return_sum_of_numbers)#get help on user defined function
```

```
Help on function return_sum_of_numbers in module __main__:

return_sum_of_numbers(a, b, c)
    Summary Line: A function that adds three numbers and return the result
    Extended description: so simple. Nothing to explain more:)
    Arguments:
        a, b, c(int or float):three numbers to add
    Returns:
        int or float: the result of adding three numbers
```

9 Built-in Functions - a more powerful `print()`

- Using **format string** to display string in certain format. Commonly used ones are:
  - `d`: Signed Decimal Int
  - `e` or `E`: Exponential notation with 'e' or 'E'

  - `f`: Floating point real number.
  - `s`: String

- Review special characters such as `\n`, `\t`

[7]:
```
a = 11; b = 12.34567; c= "Hello World"
print("%010d\t%2x\t%8.3f\t%.3e\t%s" % (a, a, b, b, c))# using format operator %
print(f"{a:10d}\t{a:2X}\t{b:8.4f}\t{b:.4E}\t{c:s}")#m.n  (the minimum total␣
 ↪width):(the number of digits after the decimal point)
```

```
0000000011        b        12.346        1.235e+01        Hello World
        11        B       12.3457        1.2346E+01       Hello World
```

10 Built-in Functions - Manipulating data types

- `type()`: returns the type
- `int()`, `float()`, `str()` etc: converting or casting to other types

11 Built-in Functions - `type()`

- Getting type or confirm/compare type

[8]:
```
x = "This is a string"
y = 1
z = True
a = 1.17
b = [x, y, z, a]
print (type(x), type(y), type(z), type(a), type(b))
print ("Is this true:", type(y) == int)
```

```
<class 'str'> <class 'int'> <class 'bool'> <class 'float'> <class 'list'>
Is this true: True
```

12 Built-in Functions - `int()`, `float()`, `str()` etc

- Converting/Casting types

[9]:
```
print(int("13"), end =';'); print(str(13))# string < > int(base 10)
print(int("0b1101", 2), end =';'); print(bin(13)) # string < > int(base  )
print(int("0xd", 16), end =';'); print(hex(13)) # string < > int(base 16)

print(chr(97), end =';'); print(ord("a")) # unicode < > char

print(int(3.0), end = ';'); print(float(3)) # float < > int
```

```
13;13
13;0b1101
13;0xd
a;97
3;3.0
```

13 Built-in Functions - `input()`

- `input()` returns **string** from input prompt
- To use returned string for calculation, convert to numbers

[10]:
```
name = input('My name is ')
print('Ah, so your name is ', name)
```

```
My name is Jing
Ah, so your name is  Jing
```

[11]:
```
# add two number input by user
a = input("Enter number a: ")
b = input("Enter number b: ")
print(a + b) # incorrect! a, b are strings. Thus a+b returns a new string␣
 ↪combining a and b
print(int(a) + int(b)) #convert string a, b to numerical values before␣
 ↪calculation
```

```
Enter number a: 5
Enter number b: 4
54
9
```