Python Coding Schools

9th Lesson: Function

Seed Academy

Agenda

- wk1. Installing Python, HelloWorld
- wk2. Arithmetic Operators
- wk3. Data Types : Integer, Floating point, Boolean, String
- wk4. Data Structures: List
- wk5. Data Structures: Set, Tuples
- wk6. Data Structures: Dictionary

Agenda

- wk7. Control flows : IF statement
- wk8. Loops: While, For
- wk9. Function
- wk10. Class
- wk11. Data Visualization

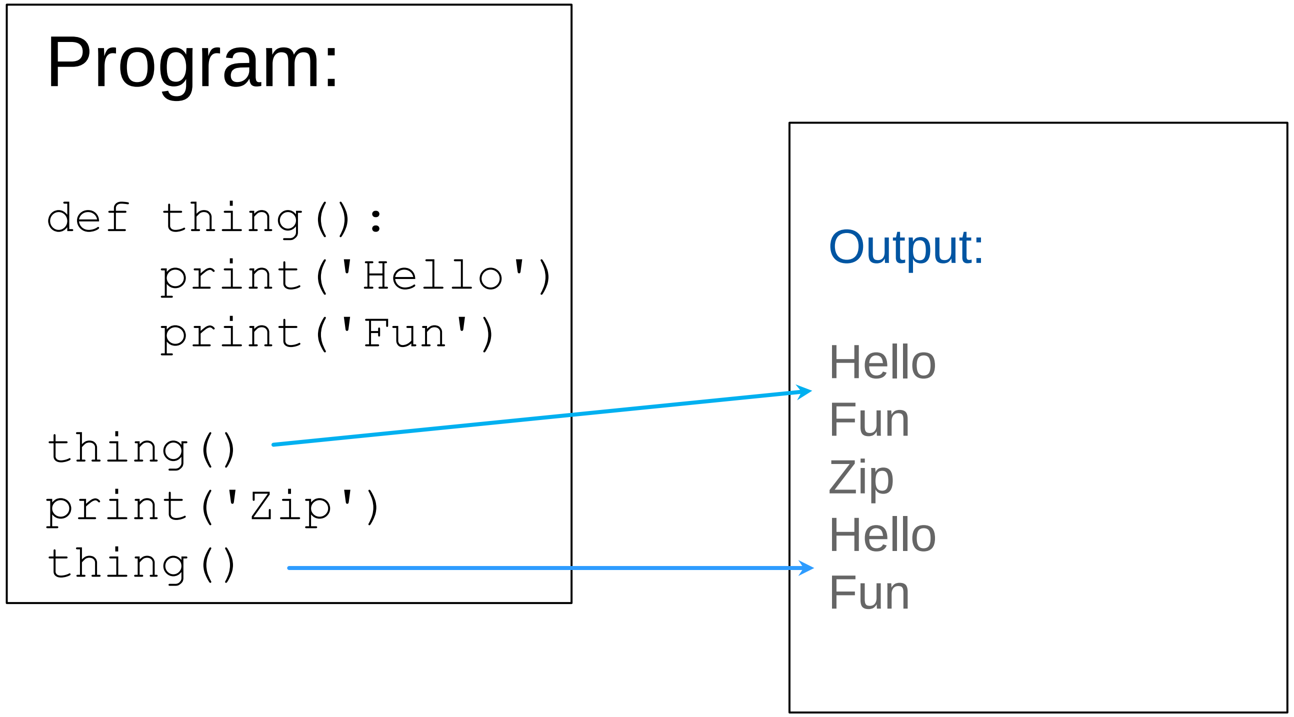
Class materials

https://github.com/TaeheeJeong/seedacademy

https://github.com/TaeheeJeong/SummerCoding2023

Stored (and reused) Steps

We call these reusable pieces of code “functions”



Program:

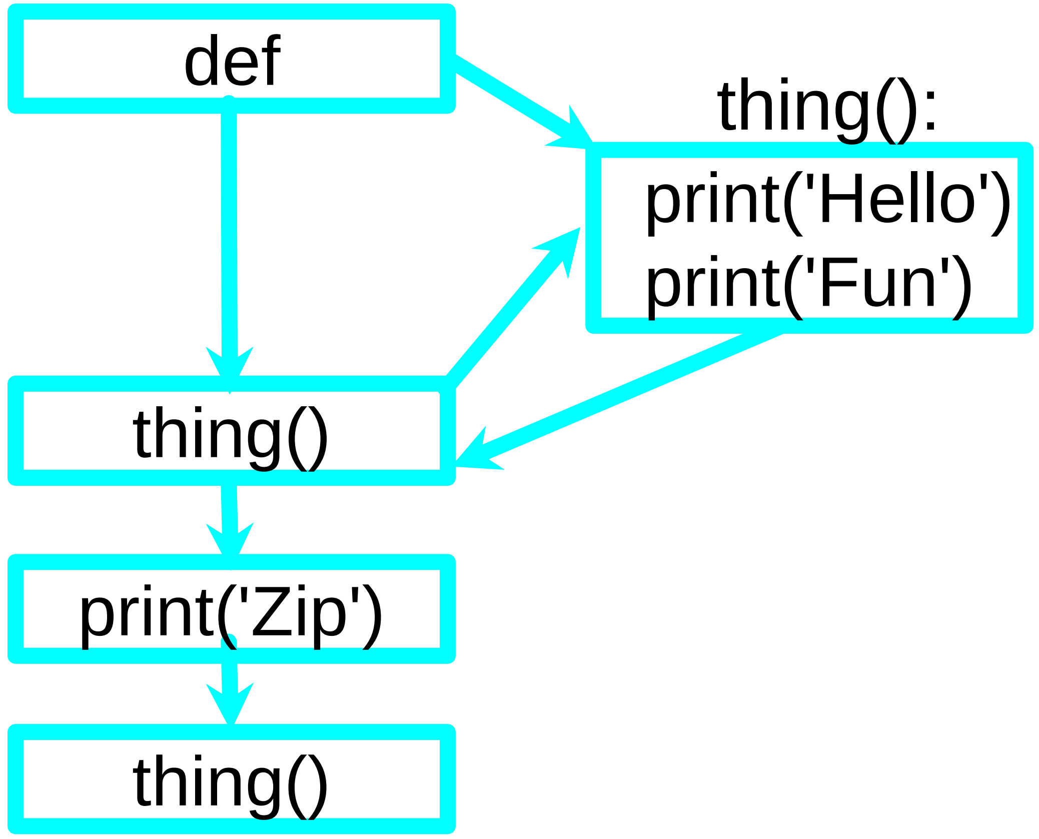
```
def thing():
    print('Hello')
    print('Fun')

thing()
print('Zip')
thing()
```

Output:

```
Hello
Fun
Zip
Hello
Fun
```

Source: Python for Everybody, Charles Severance, University of Michigan School of Information

Stored (and reused) Steps



Program:

```
def thing():
    print('Hello')
    print('Fun')

thing()
print('Zip')
thing()
```

Python Functions

There are two kinds of functions in Python.

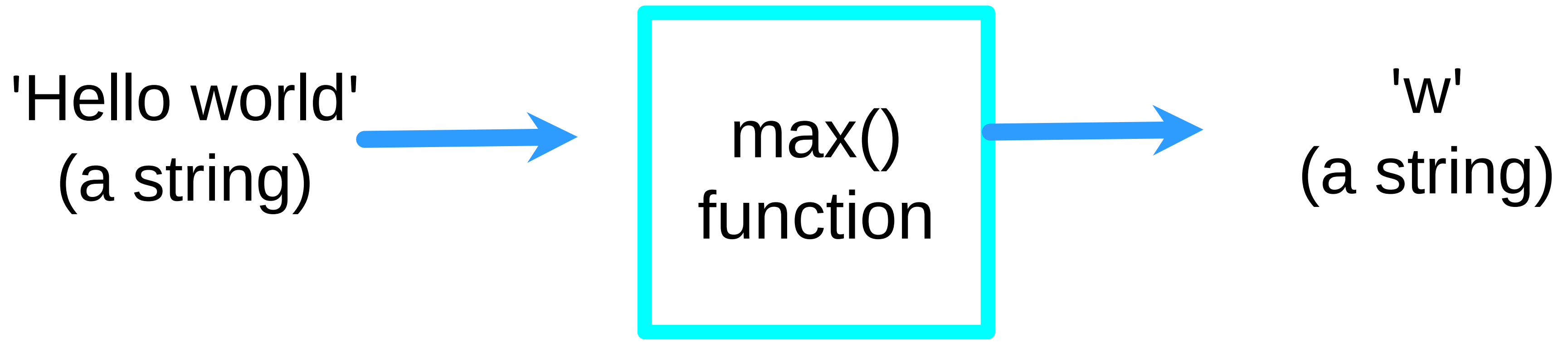
- Built-in functions that are provided as part of Python - print(), input(), type(), float(), int() ...
- Functions that we define ourselves and then use

We treat the built-in function names as “new” reserved words
(i.e., we avoid them as variable names)

Built-in function: max()

```
>>> big = max('Hello world')
>>> print(big)
w
```

A function is some stored code that we use. A function takes some input and produces an output.



Function Definition

- In Python a function is some reusable code that takes arguments(s) as input, does some computation, and then returns a result or results
- We define a function using the def reserved word
- We call/invoke the function by using the function name, parentheses, and arguments in an expression

Building our Own Functions

- We create a new function using the def keyword followed by optional parameters in parentheses
- We indent the body of the function
- This defines the function but does not execute the body of the function

```
def print_lyrics():
    print("I'm a lumberjack, and I'm okay.")
    print('I sleep all night and I work all day.')
```

Definitions and Uses

- Once we have defined a function, we can call (or invoke) it as many times as we like
- This is the store and reuse pattern

Arguments

- An argument is a value we pass into the function as its input when we call the function
- We use arguments so we can direct the function to do different kinds of work when we call it at different times
- We put the arguments in parentheses after the name of the function

```
big = max('Hello world')
```

Argument

Parameters

- A parameter is a variable which we use in the function definition.
- It is a “handle” that allows the code in the function to access the arguments for a particular function invocation.

```
>>> def greet(lang):
...     if lang == 'es':
...        print('Hola')
...     elif lang == 'fr':
...        print('Bonjour')
...     else:
...        print('Hello')
...
>>> greet('en')
Hello
>>> greet('es')
Hola
>>> greet('fr')
Bonjour
```

Return Values

Often a function will take its arguments, do some computation, and return a value to be used as the value of the function call in the calling expression.

```
def greet():
    return "Hello"

print(greet(), "Glenn")
print(greet(), "Sally")
```

```
Output
Hello Glenn
Hello Sally
```

Return Value

- A “fruitful” function is one that produces a result (or return value)
- The return statement ends the function execution and “sends back” the result of the function

```
>>> def greet(lang):
...     if lang == 'es':
...         return 'Hola'
...     elif lang == 'fr':
...         return 'Bonjour'
...     else:
...         return 'Hello'
...

>>> print(greet('en'),'Glenn')
Hello Glenn
>>> print(greet('es'),'Sally')
Hola Sally
>>> print(greet('fr'),'Michael')
Bonjour Michael
```

Source: Python for Everybody, Charles Severance, University of Michigan School of Information

Multiple Parameters / Arguments

- We can define more than one parameter in the function definition
- We simply add more arguments when we call the function
- We match the number and order of arguments and parameters

```
>>> def addtwo(a, b):
>>>     added = a + b
>>>     return added

>>> x = addtwo(3, 5)
>>> print(x)
>>> 8
```

Void (non-fruitful) Functions

- When a function does not return a value, we call it a “void” function
- Functions that return values are “fruitful” functions
- Void functions are “not fruitful”

To function or not to function...

- Organize your code into “paragraphs” - capture a complete thought and “name it”
- Don’t repeat yourself - make it work once and then reuse it
- If something gets too long or complex, break it up into logical chunks and put those chunks in functions
- Make a library of common stuff that you do over and over - perhaps share this with your friends...

Recap: Functions with ‘def’

- Function can be defined with ‘def’
- Function name
- Function argument or parameter
- indent

```
>>> def greet(lang):
...     if lang == 'es':
...        print('Hola')
...     elif lang == 'fr':
...        print('Bonjour')
...     else:
...        print('Hello')
...
>>> greet('en')
Hello
>>> greet('es')
Hola
>>> greet('fr')
Bonjour
```

Recap: Return Value

A “fruitful” function is one that produces a result (or return value)

The return statement ends the function execution and “sends back” the result of the function

```
>>> def greet(lang):
...     if lang == 'es':
...         return 'Hola'
...     elif lang == 'fr':
...         return 'Bonjour'
...     else:
...         return 'Hello'
...
>>> print(greet('en'),'Glenn')
Hello Glenn
>>> print(greet('es'),'Sally')
Hola Sally
>>> print(greet('fr'),'Michael')
Bonjour Michael
```

Recap: Multiple Parameters / Arguments

- We can define more than one parameter in the function definition
- We simply add more arguments when we call the function
- We match the number and order of arguments and parameters

```
>>> def addtwo(a, b):
>>>     added = a + b
>>>     return added

>>> x = addtwo(3, 5)
>>> print(x)
>>> 8
```

Acknowledgements / Contributions

Initial Development: Charles Severance, University of Michigan School of Information

Modification: Taehee Jeong, San Jose State University