Lab 3 Python Concepts

Lab3 Python Concepts

1. New Type of Variable
2. Boolean Expressions
3. Conditionals (IF Statements)
4. User-Defined Functions

Open the Python interpreter to get the prompt:

>>>

We will also use PythonTutor: pythontutor.com

Python Types

Def: An **integer** (int) is a number that displays with no decimal points.

Def: A **float** is a number that displays with decimal points.

Def: A **string** is a sequence of characters surrounded by single quotes.

Def: A **list** is an ordered collection of items.

Def: A **boolean** can be one of two values: `True` or `False`.

Boolean Expressions

Numerical Operators (input numbers, output number)

Operator	Example	Result
Sum	`1 + 2`	`3`
Difference	`1 - 2`	`-1`
Multiply	`1 * 2`	`2`
Divide	`1 / 2` `1.0 / 2.0`	`0` `0.5`
Power	`1 ** 2`	`1`

Numerical Operators (input numbers, output Boolean)

Operator	Example	Result (`True` or `False`)
Equality	`1 == 2`	`False`
Inequality	`1 != 2`	`True`
Less Than	`1 < 2`	`True`
Less Than or Equal To	`1 <= 2`	`True`
Greater Than	`1 > 2`	`False`
Greater Than or Equal To	`1 >= 2`	`False`

Boolean Expressions

String Operators (input strings)

Operator	Example	Result
Sum	`'a' + 'b'`	`'ab'`
Equality	`'a' == 'b'`	`False`
Inequality	`'a' != 'b'`	`True`

output string

output boolean

Boolean Operators (input booleans, output boolean)

Operator	Example	Result
And	`True and False` `(4<5) and (6>=6)`	`False` `True`
Or	`True or False` `(4<5) or 'a'!= 'b'`	`True` `True`
Not	`not True` `not (4<5)`	`False` `False`

The IN Function

Boolean Operators (input booleans, output boolean)

Operator	Example	Result
And	`True and False` `(4<5) and (6>=6)`	`False` `True`
Or	`True or False` `(4<5) or'a'!= 'b'`	`True` `True`
Not	`not True` `not (4<5)`	`False` `False`
In	`'a' in ['a','b','c']` `1 in ['a','b','c']`	`True` `False`

Python IF Statements

"If some value is true, do something."

```
x = 4
y = 10
if x < y:
    print 'x is less than y.'
```

```
if <boolean expression>:
    statement 1
    statement 2
    ...
```

Indentation matters!

"If some value is true, do something, otherwise do something else."

```
x = 4
y = 10
if x < y:
    print 'x is less than y.'
else:
    print 'x >= y.'
```

```
if <boolean expression>:
    statement 1
    ...
else:
    statement 2
    ...
```

Python ELIF Statements

"If some value is true, do something. Otherwise…"

"If some OTHER value is true, do something. Otherwise…"

"If some OTHER value is true, do something. Otherwise…"

"…"

```
x = 4
y = 10
if x < y:
    print 'x is less than y.'
elif x == y:
    print 'x and y are equal.'
else:
    print 'x > y.'
```

```
if <boolean expression>:
    statement 1
    ...
elif <boolean expression>:
    statement 2
    ...
else:
    statement 3
    …
```

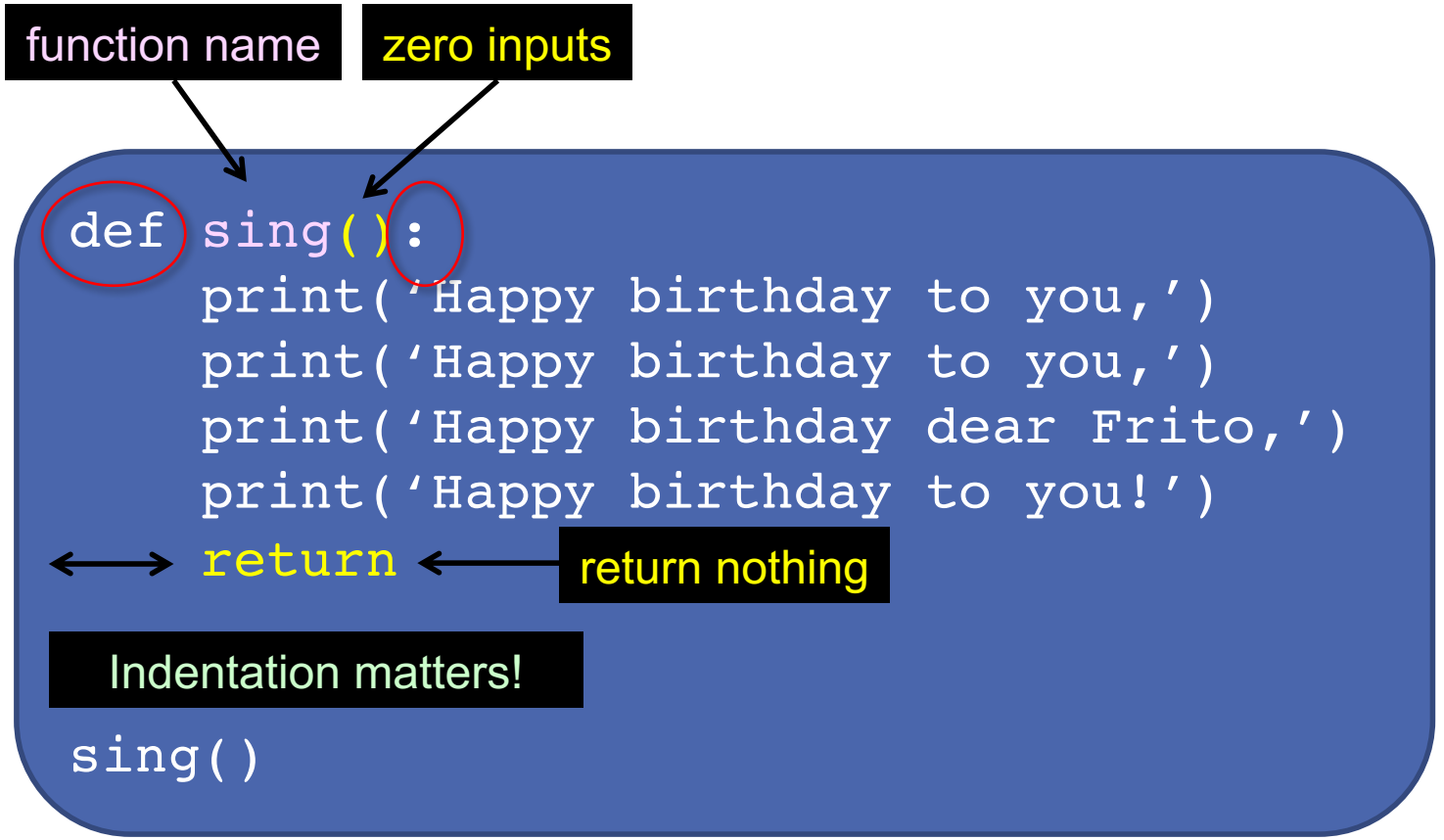
Built-in Functions

Def: A **function** is a reusable block of code

Function Name	Arguments (Inputs)	Returns (Outputs)	Examples
`quit()` or `exit()`	nothing	-	`quit()`
`type()`	expression	type	`type('string')` `type(1)`
`len()`	string or list	integer	`len([1,2,3])` `len('string')`

A function is specified by:
- a function name
- The arguments (inputs), comma-separated if multiple
- The returned variables (outputs), comma-separated if multiple

User-Defined Functions



Functions are like *small subprograms*

Function Name	Arguments (Inputs)	Returns (Outputs)	Examples
`sing()`	nothing	nothing	`sing()`

User-Defined Functions

one input

```
def sing(person):
    print('Happy birthday to you,')
    print('Happy birthday to you,')
    print('Happy birthday dear ' + person + ',')
    print('Happy birthday to you!')
    return

sing('Anna')
```

return nothing

Function Name	Arguments (Inputs)	Returns (Outputs)	Examples
sing()	string	nothing	sing('Anna')

User-Defined Functions: Scope

```
def sing(person):
    print('Happy birthday
    print('Happy birthday to you,')
    print('Happy birthday dear ' + person + ',')
    print('Happy birthday to you!')
    return

name = 'Anna'
sing(name)
```

Memory in sing() function	
Variable Name	Value
person	'Anna'
+ Global Memory…	

GLOBAL Memory	
Variable Name	Value
sing()	<function>
name	'Anna'

User-Defined Functions: Scope

```
def sing(person):
    song = ''
    song = song + 'Happy b
    song = song + 'Happy b
    song = song+'Happy bir                    ,\n'
    song = song + 'Happy b                    you.\n'
    return song

name = 'Anna'
annasong = sing(name)
print(annasong)
```

one input

return a string

Memory in sing() function	
Variable Name	Value
person	'Anna'
song	'Happy Birth…'
+ Global Memory…	

GLOBAL Memory	
Variable Name	Value
sing()	<function>
name	'Anna'

User-Defined Functions: Scope

```
def sing(person):
    song = ''
    song = song + 'Happy birthday to you,\n'
    song = song + 'Happy birthday to you,\n'
    song = song+'Happy birthday dear '+person+',\n'
    song = song + 'Happy birthday to you!\n'
    return song

name = 'Anna'
annasong = sing(name)
print(annasong)
```

GLOBAL Memory	
Variable Name	Value
`sing()`	<function>
`name`	'Anna'
`annasong`	'Happy Birth..'

User-Defined Functions: Scope

```
def sing(person):
    song = ''
    song = song + happy()
    song = song + happy()
    song = song+'Happy bir                    \n'
    song = song + happy()
    return song

def happy():
    return 'Happy bir

name = 'Anna'
annasong = sing(name)
print(annasong)
```

Memory in sing() function

Variable Name	Value
person	'Anna'
song	''
+ Global Memory…	

Memory in happy() function

Variable Name	Value
+ Global Memory…	

GLOBAL Memory

Variable Name	Value
sing()	<function>
happy()	<function>
name	'Anna'

User-Defined Functions: Scope

```
def sing(person):
    song = ''
    song = song + happy()
    song = song + happy()
    song = song+'Happy bir                    \n'
    song = song + happy()
    return song

def happy():
    return 'Happy bir

name = 'Anna'
annasong = sing(name)
print(annasong)
```

Memory in SING() function

Variable Name	Value
person	'Anna'
song	'Happy Birth..'
+ Global Memory…	

Memory in happy() function

Variable Name	Value
+ Global Memory…	

GLOBAL Memory

Variable Name	Value
sing()	<function>
happy()	<function>
name	'Anna'

Good Practice: Global Memory contains functions

```
def sing(person):
    song = ''
    song = song + happy()
    song = song + happy()
    song = song+'Happy birthday
    song = song + happy()
    return song

def happy():
    return 'Happy birt

def main():
    name = 'Anna'
    annasong = sing(n
    print(annasong)
    return

if __name__ == '__main__':
    main()
```

Memory in sing() function

Variable Name	Value
person	'Anna'
song	'Happy Birth…'

Memory in happy() function

Variable Name

GLOBAL Memory

Variable Name	Value
sing()	<function>
happy()	<function>
main()	<functions>

Memory in main() fu

Variable Name	V
name	'Anna
annasong	'Happy Birth..'

…he bottom of the file.
"If running from a Terminal, call the main() function."