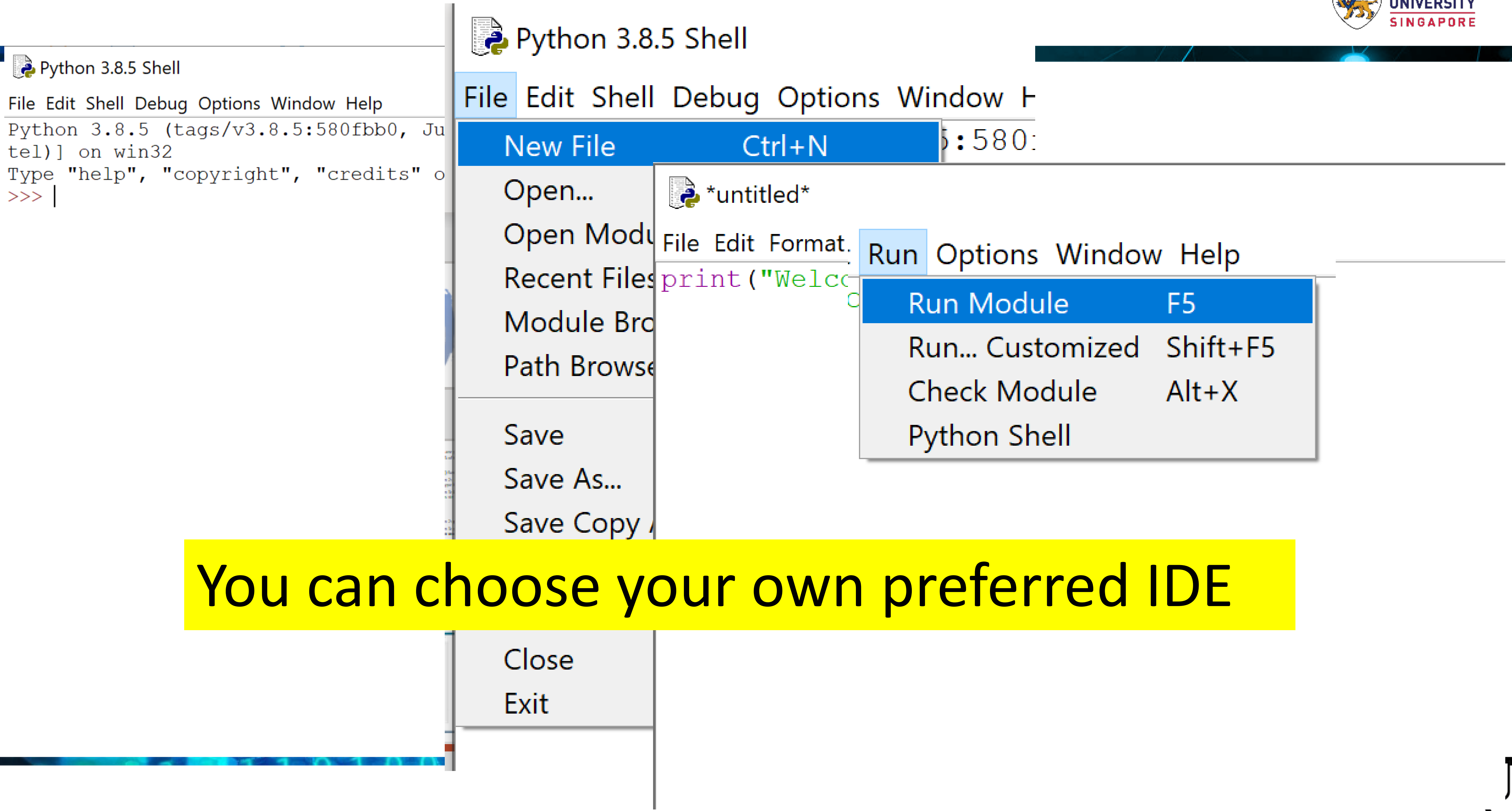
SC1003 Introduction to computational thinking and programming

Python 2 vs Python 3

- There are plenty of differences between version 2 and version 3 of the Python language.

- **input()** function
- Python 2: evaluates the data it receives and returns the data type based on what it ‘thinks’ it should be.
- Python 3: **input()** is a function that always returns the data as str (i.e. string) data type object.

- **Print**
- Python 2: **print “Hello world”**
- Python 3: print() is a function and is written as **print (“Hello world”)**
-


Python 3.8.5 Shell

```
Python 3.8.5 (tags/v3.8.5:580fbb0, Ju
tel)] on win32
Type "help", "copyright", "credits" o
>>> |
```

untitled

```
print("Welc
```

You can choose your own preferred IDE

Variables, Keywords, Data types, Naming Convention, built-in functions Constants,

Variables - Place holders (Naming convention)

Keywords – Special words reserved in Python

Data types – assignment to variables

Built in functions – libraries

Argument input and output to function calls

Course Review
Variables
Identifier

Identifier in Python

Identifier: a name given to an entity in Python

- Helps in differentiating one entity from another
- Name of the entity must be unique to be identified during the execution of the program

640509 040147

Rules for Writing Identifiers

What can be used?

- Uppercase and lowercase letters A through Z (26 * 2 = 52)
- The underscore, '_' (1)
- The digits 0 through 9, except for the first character (10)

52 + 1 + 10 = 63

Syntax Rules in Python

- Must begin with a letter or _
  - **'Ab123'** and **'_b123'** are ok
  - **'123ABC'** is not allowed
- May contain letters, digits, and underscores

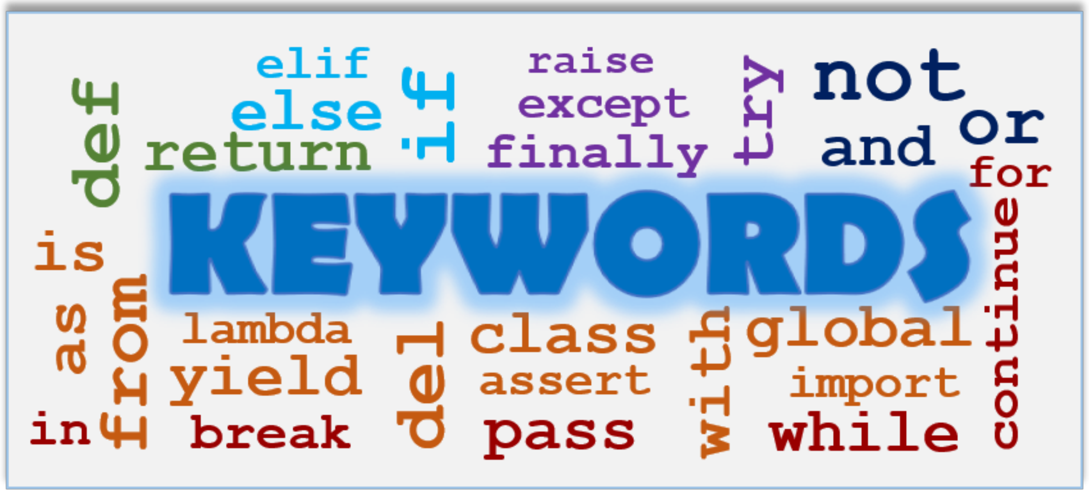
  this_is_an_identifier_123
- Should **not** use keywords
- Upper case and lower case letters are different

  'LengthOfRope' is not **'lengthofrope'**

  ⚠ *Python is **case sensitive***
- Can be of any length
- Names starting with _ have special meaning

Keywords

- Special words reserved in Python
- Programmers should not use keywords to *name* things



Note: Old Python keyword ‘***exec***’ was removed in Python 3

slido

Identify the invalid variable names among the following ones:

int
return
For
Us$
2person
userName
HALF_WINWIDTH
__name__
Phone#

ⓘ Start presenting to display the poll results on this slide.

Python Naming Conventions

```python
import math
radiusString = input("Enter the radius of your circle:")
radiusFloat = float (radiusString)
circumference = 2 * math.pi * radiusFloat
area = math.pi * radiusFloat * radiusFloat
```

vs.

```python
import math
a = input("Enter the radius of your circle:")
b = float (a)
c = 2 * math.pi * b
d = math.pi * b * b
```

What is c? It is not immediately clear.

- **Both programs work**
- **They are different when readability counts**

- variable names should be in lowercase, with words separated by underscores as necessary to improve readability

  e.g. radius_float
- mixedCase is allowed

  e.g. radiusFloat

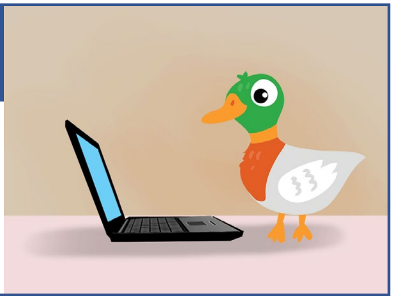
Course Review
Data type

Data Types

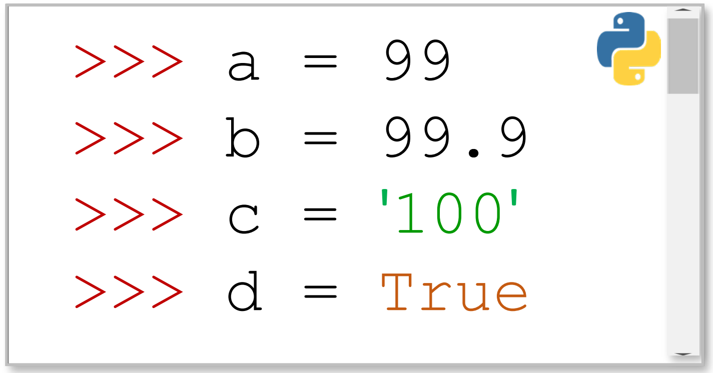
Compared to C and Java, how does Python know the data types?

Python uses *Duck-Typing*

"When I see a bird that walks like a duck and swims like a duck and quacks like a duck, I call that bird a duck." – James Whitcomb Riley



Examples



```
>>> a = 99
>>> b = 99.9
>>> c = '100'
>>> d = True
```

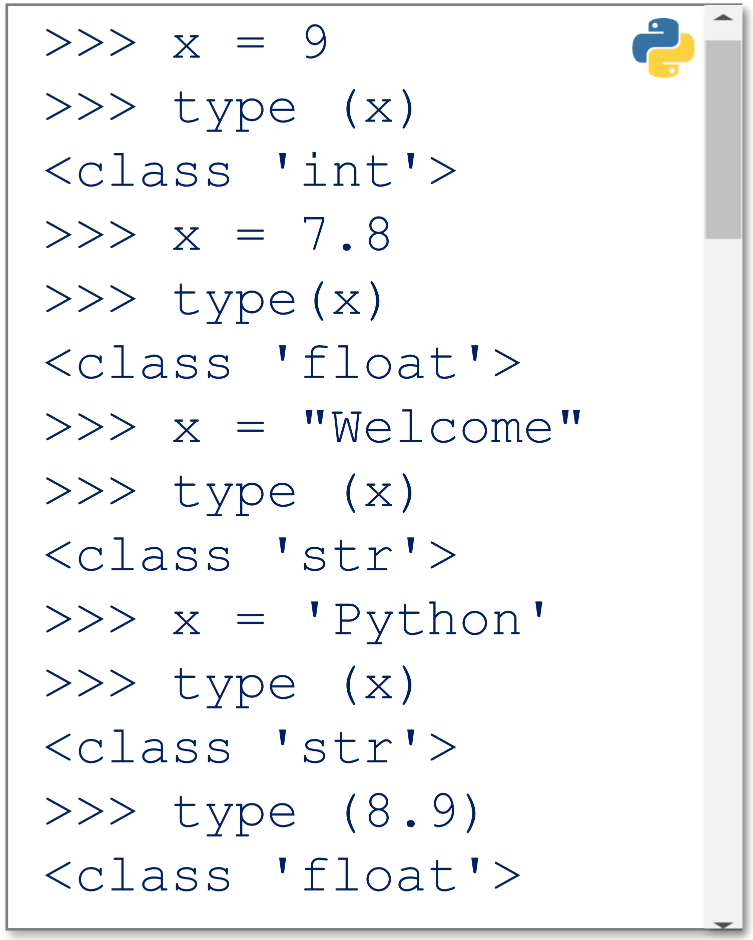
Four variables!

What are their data types?

Data Types (Cont'd)

Type Function

In Python, the `type()` function allows you to know the type of a **variable** or **literal**.



```
>>> x = 9
>>> type (x)
<class 'int'>
>>> x = 7.8
>>> type(x)
<class 'float'>
>>> x = "Welcome"
>>> type (x)
<class 'str'>
>>> x = 'Python'
>>> type (x)
<class 'str'>
>>> type (8.9)
<class 'float'>
```

- Python does not have variable declaration, like Java or C, to announce or create a variable.
- A variable is created by just assigning a value to it and the type of the value defines the type of the variable.
- If another value is re-assigned to the variable, its type can change.

Assignment Operator

```
import math
radiusString = input("Enter the radius of your circle:")
radiusFloat = float (radiusString)
circumference = 2 * math.pi * radiusFloat
area = math.pi * radiusFloat * radiusFloat
```

Basic Syntax

Left Hand Side (LHS) = Right Hand Side (RHS)

$\downarrow$ **a variable** $\qquad \downarrow$ **an expression**

Assignment means:

1. Evaluate the expression on RHS
2. Take the resulting value and assign it to the name (variable) on the LHS.

Data Types String

String - designated as '**str**'

- It is basically a sequence, typically a sequence of characters delimited by single quote ('…') or double quotes ("…") for a single line sentence, even triple quotes ("'…"') to display a paragraph with multiple lines
- First *collection* type that was discussed
- Collection type contains multiple objects organized as a single object

Examples

```
>>> a = 'Length'
>>> b = "8225 welcome"
>>> c = "ewwew sdcd &8 $5##"
>>> d = 'ewwew sdcd &8 $5##'
>>> e = '''
Welcome to Stock Master!
Please select a function to continue:
(1) Search for a stock code by
keyword
(2) View stock trend by code
(sample input 1 or 2)
'''
```

```python
e = '''
Welcome to Stock Master!
Please select a function to continue:
 (1) Search for a stock code by keyword
 (2) View stock trend by code
 (sample input 1 or 2)
'''

print(e)

a = 'Length'
print(a)
b= "8225 welcome"
print(b)
```

```
====== RESTART: D:/LF work/LF current Cc

Welcome to Stock Master!
Please select a function to continue:
 (1) Search for a stock code by keyword
 (2) View stock trend by code
 (sample input 1 or 2)

Length
8225 welcome
```

slido

What is the output of the following program?

```
pay_1 = 12
pay1= 23.2
resulT_1 = pay_1*2 - pay1
result_1 = pay1 - pay_1
print(resulT_1)
```

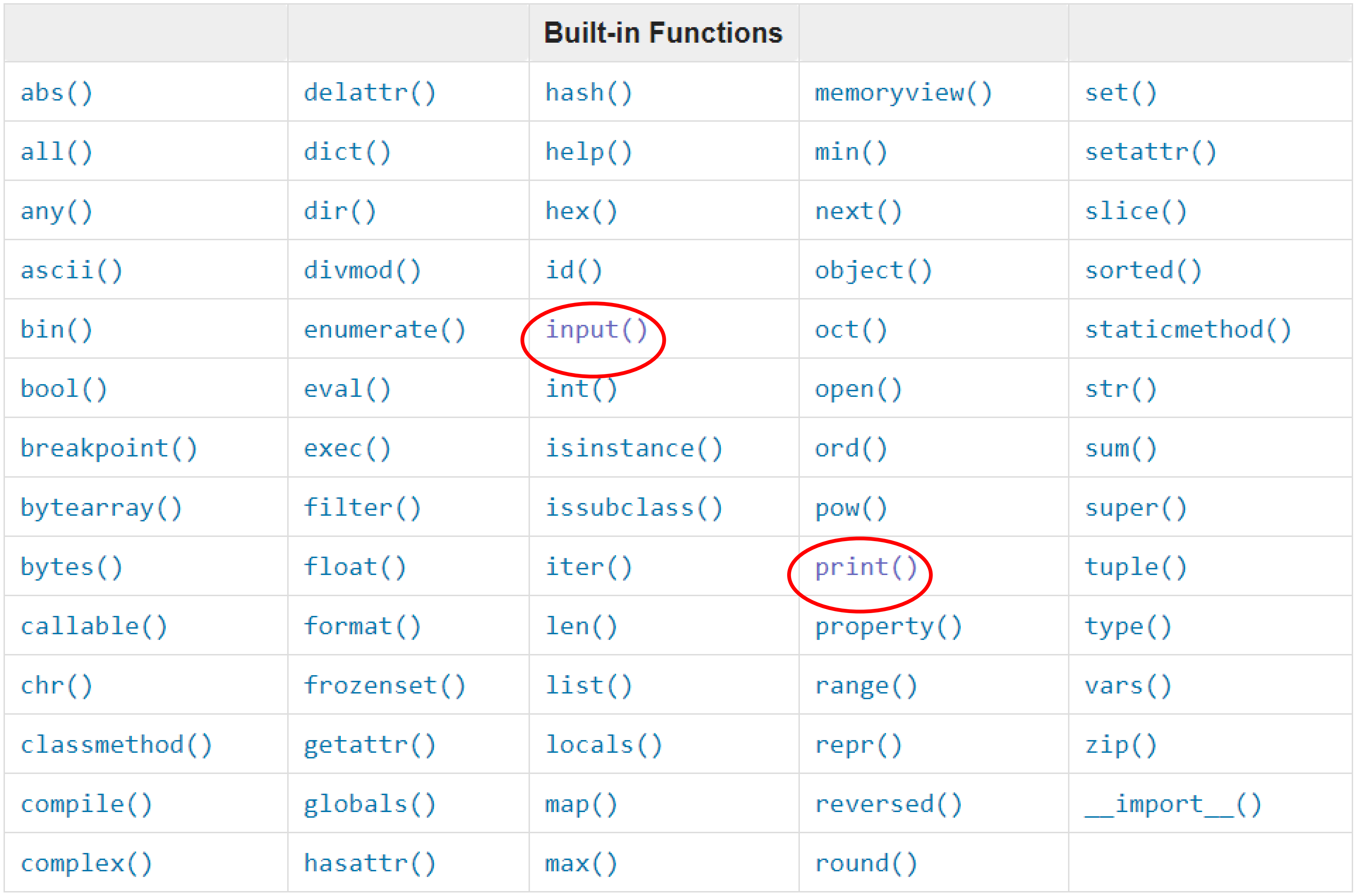
1/2	**1/4**	**1/8**	**1/16**	**1/32**	**1/64**	**1/128**	**1/256**	
0.5	0.25	0.125						
0.5	0.25		0.0625					
0.5	0.25			0.03125	0.015625	0.0078125		
0.5	0.25			0.03125	0.015625		0.00390625	

FUNCTION CALL
INPUT, OUTPUT

3. INPUT, PROCESSING, AND OUTPUT

- Typically, computer performs three-step process
  - Receive input
    - Input: any data that the program receives while it is running
  - Perform some process on the input
    - Example: mathematical calculation
  - Produce output

```
# 1. prompt user for the radius
# 2. apply circumference and area formulae
# 3. print the results

import math
radiusString = input("Enter the radius of your circle:")
radiusFloat = float (radiusString)
circumference = 2 * math.pi * radiusFloat
area = math.pi * radiusFloat * radiusFloat

print()    # print a line break
print ("The circumference of your circle is:",circumference, ", and the area
is:", area)
```



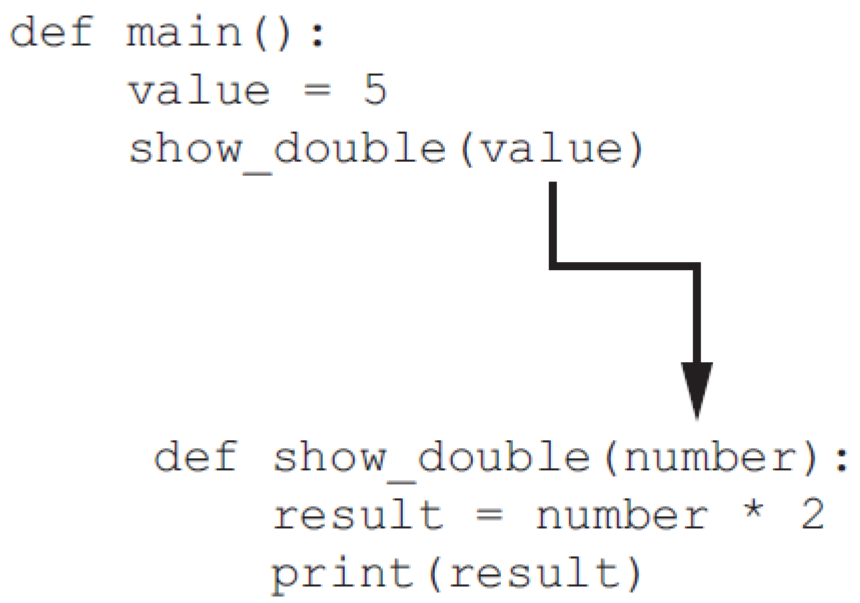
		Built-in Functions		
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

https://docs.python.org/3/library/functions.html

Passing Arguments to Functions

- Argument: piece of data that is sent into a function
  - Function can use argument in calculations and processing
  - When calling the function, the argument is placed in parentheses following the function name

Figure 5-13 The `value` variable is passed as an argument

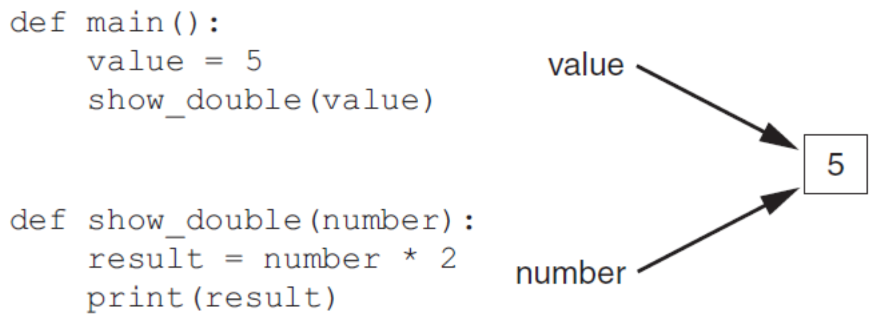


```
def main():
    value = 5
    show_double(value)


def show_double(number):
    result = number * 2
    print(result)
```

Parameter variable in Functions

- Parameter variable: variable that is assigned the value of an argument when the function is called
  - The parameter and the argument reference the same value
  - General format:
  - `def function_name(parameter):`
  - Scope of a parameter: the function in which the parameter is used

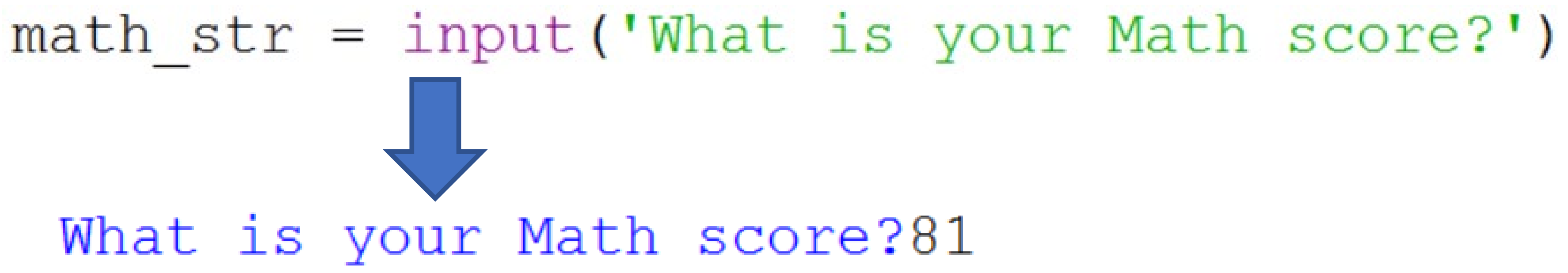
Figure 5-14 The `value` variable and the `number` parameter reference the same value



```
def main():
    value = 5
    show_double(value)

def show_double(number):
    result = number * 2
    print(result)
```

Reading Input from the Keyboard

- Most programs need to read input from the user
- Built-in `input` function reads input from keyboard
  - Returns the data as a string, even if the user enters numeric data.
  - Format: `*variable* = input(*prompt*)`
    - `prompt` is typically a string instructing user to enter a value
  - Does not automatically display a space after the prompt

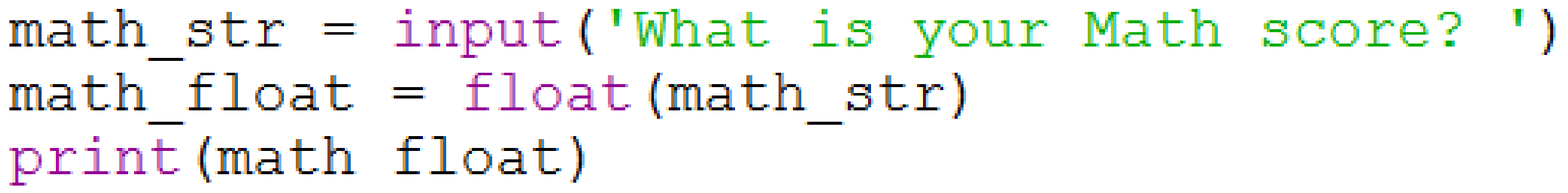


```
math_str = input('What is your Math score?')
```

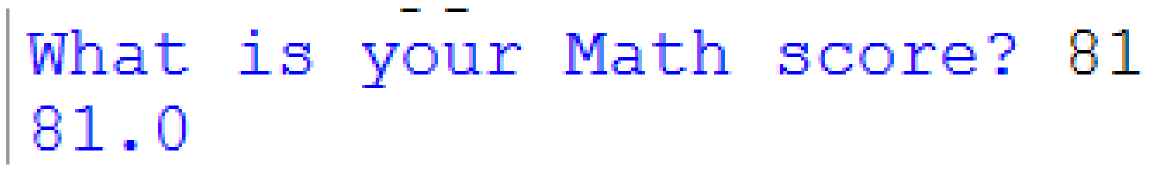
```
What is your Math score?81
```

Reading Numbers with the `input` Function

- `input` function always returns a string
- Built-in functions convert between data types
  - `int(`*`item`*`)` converts *`item`* to an `int`
  - `float(`*`item`*`)` converts *`item`* to a `float`
  - Type conversion only works if item is valid numeric value, otherwise, throws exception



```
math_str = input('What is your Math score? ')
math_float = float(math_str)
print(math float)
```



```
What is your Math score? 81
81.0
```

slido

How the following code works? Which step is the first step? math_score = float(input('What is your Math score? '))

ⓘ Start presenting to display the poll results on this slide.

Passing Multiple Arguments

Python allows writing a function that accepts multiple arguments

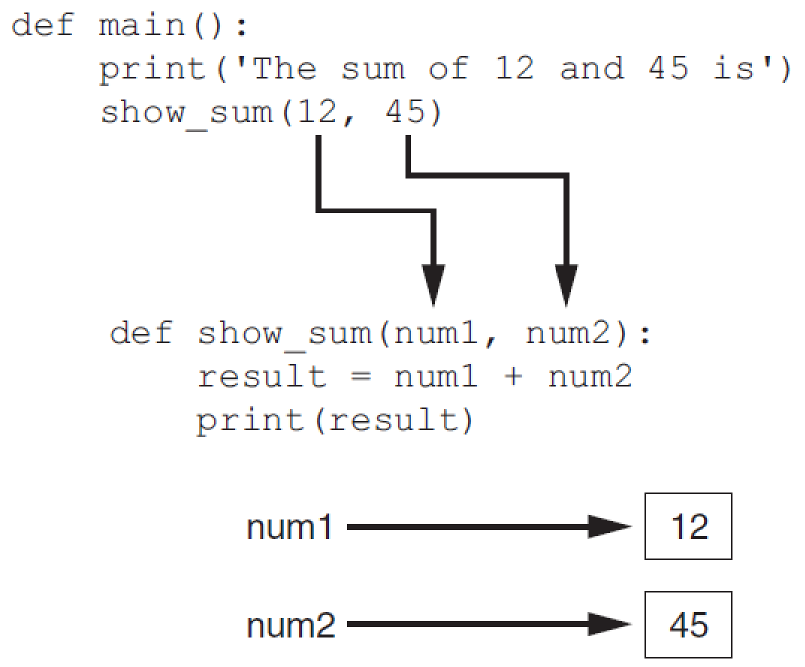
- Parameter list replaces single parameter
  - Parameter list items separated by comma

Arguments are passed *by position* to corresponding parameters

- First parameter receives value of first argument, second parameter receives value of second argument, etc.

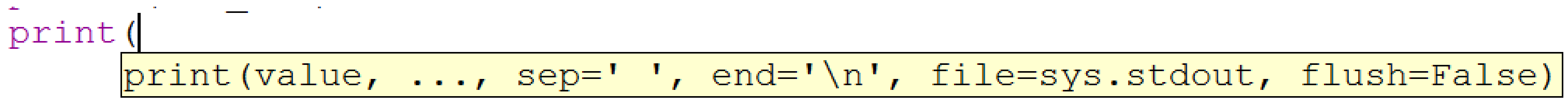
Passing Multiple Arguments (cont'd.)

Figure 5-16 Two arguments passed to two parameters



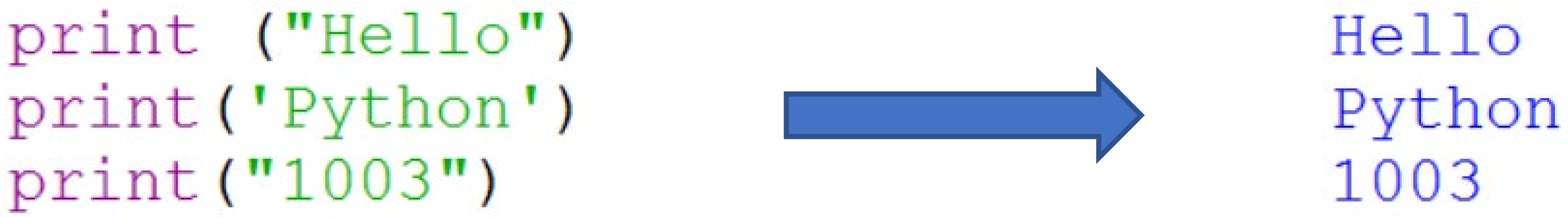
```
def main():
    print('The sum of 12 and 45 is')
    show_sum(12, 45)

def show_sum(num1, num2):
    result = num1 + num2
    print(result)
```

Displaying Output with the `print` Function



```
print(
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
```

- `print` function: displays output on the screen
- Argument: data given to a function
  - Example: data that is printed to screen
- Statements in a program execute in the order that they appear
  - From top to bottom



```
print ("Hello")
print('Python')
print("1003")
```

```
Hello
Python
1003
```

slido

What is the output of the following code?

ⓘ Start presenting to display the poll results on this slide.

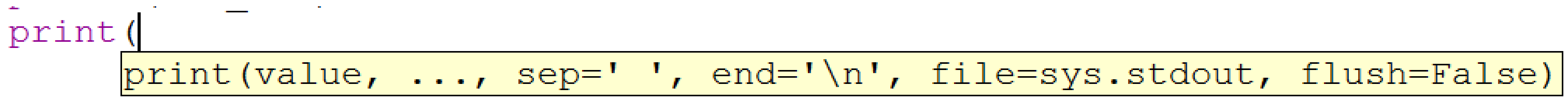
Function with Default Arguments

- Default arguments in Python functions are those arguments that take default values if no explicit values are passed to these arguments from the function call.

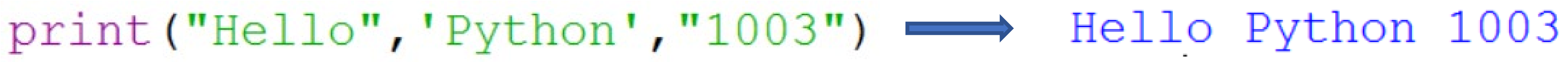
https://stackabuse.com/default-arguments-in-python-functions/

Displaying Multiple Items with the `print` Function

- Python allows one to display multiple items with a single call to `print`
  - Items are separated by commas when passed as arguments
  - Arguments displayed in the order they are passed to the function
  - Items are automatically separated by a space when displayed on screen while using default sep



```
print(
      print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
```

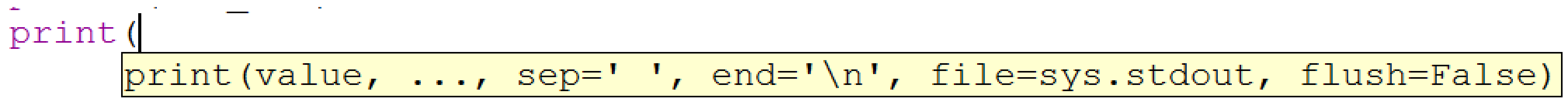


```
print("Hello",'Python',"1003")
```

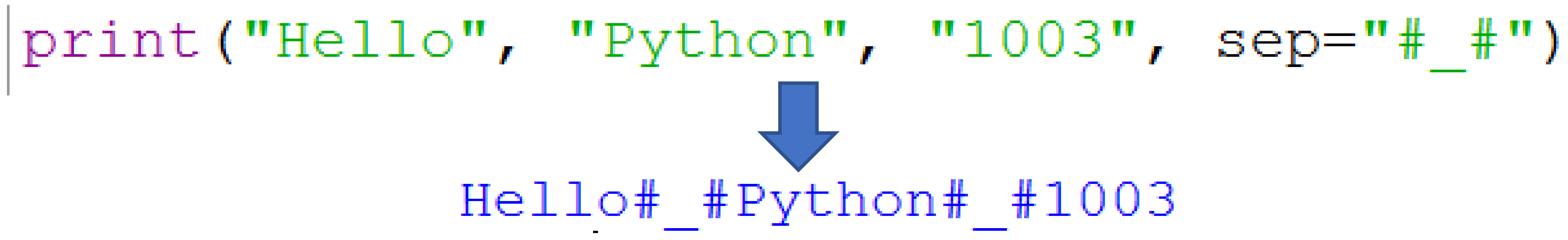
```
Hello Python 1003
```

Displaying Multiple Items with the `print` Function cont’

- `print` function uses space as item separator
  - Special argument `sep='`*`delimiter`*`'` causes `print` to use *`delimiter`* as item separator

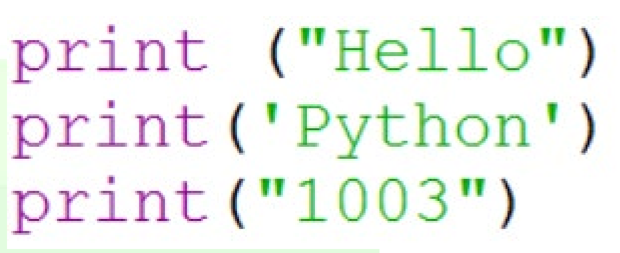


```
print(
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
```



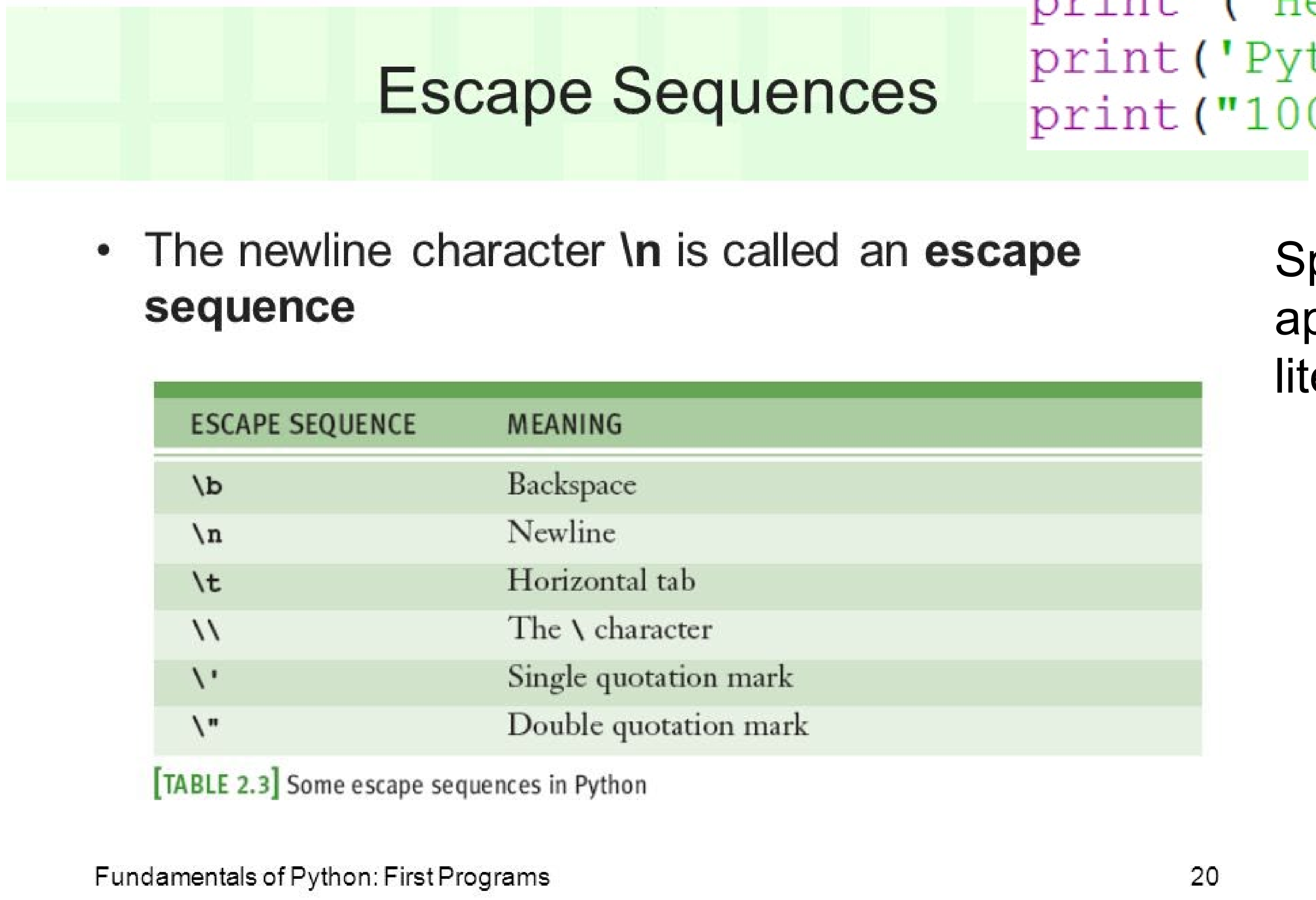
```
print("Hello", "Python", "1003", sep="#_#")
```

```
Hello#_#Python#_#1003
```



```
print ("Hello")
print('Python')
print("1003")
```

```
Hello
Python
1003
```



Escape Sequences

- The newline character **\n** is called an **escape sequence**

ESCAPE SEQUENCE	MEANING
`\b`	Backspace
`\n`	Newline
`\t`	Horizontal tab
`\\`	The \ character
`\'`	Single quotation mark
`\"`	Double quotation mark

[TABLE 2.3] Some escape sequences in Python

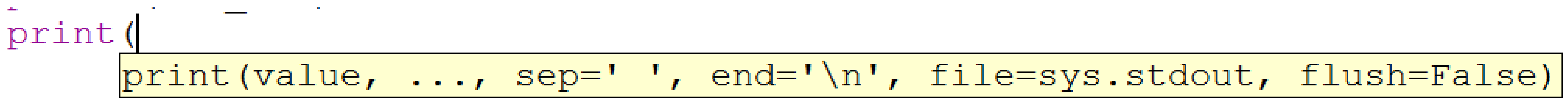
Fundamentals of Python: First Programs 20

Special characters appearing in string literal
- Preceded by backslash (\)
  - Examples: newline (`\n`), horizontal tab (`\t`)
- Treated as commands embedded in string

https://slideplayer.com/slide/7354455/

argument `end in print function`

- `print` function displays line of output
  - Newline character at end of printed data
  - Special argument `end='`*`delimiter`*`'` causes `print` to place *`delimiter`* at end of data instead of newline character



```
print(
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
```

```
print ("Hello",end = '*')
print('Python',end = '@')
print("1003")
```

```
Hello*Python@1003
```

slido

What is the output of the following code?

```
print("Python",'1003', sep='~', end='#')
print('Have',"fun")
```

ⓘ Start presenting to display the poll results on this slide.

slido

What is the output of the following code?

```
print('Hi'+"Python"+'1003')
```

ⓘ Start presenting to display the poll results on this slide.

+ operator

+ operator for two numbers: Add two numbers

+ operator for two strings: String concatenation.

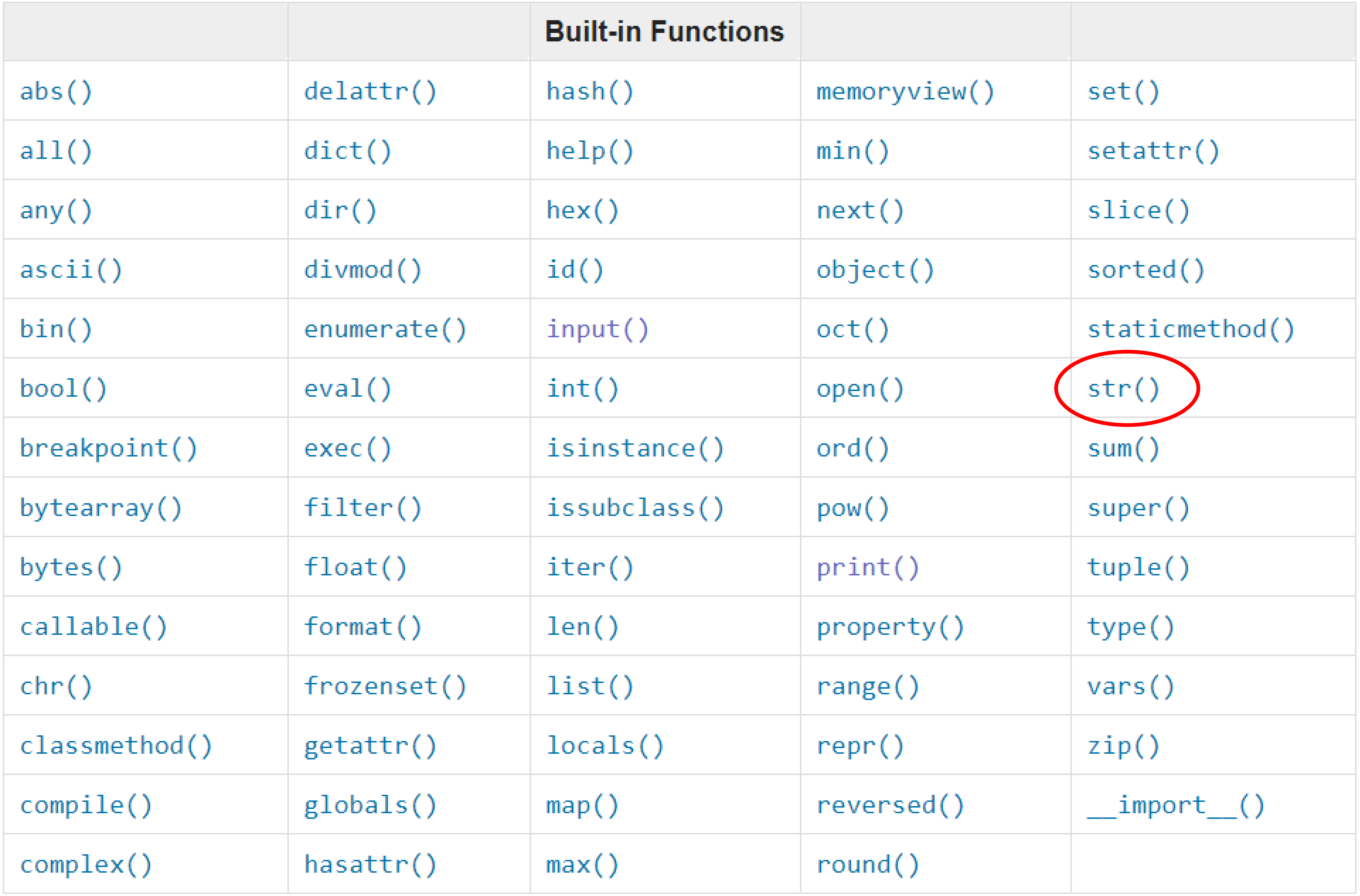
It appends one string to another.

slido

What is the output of the following code?

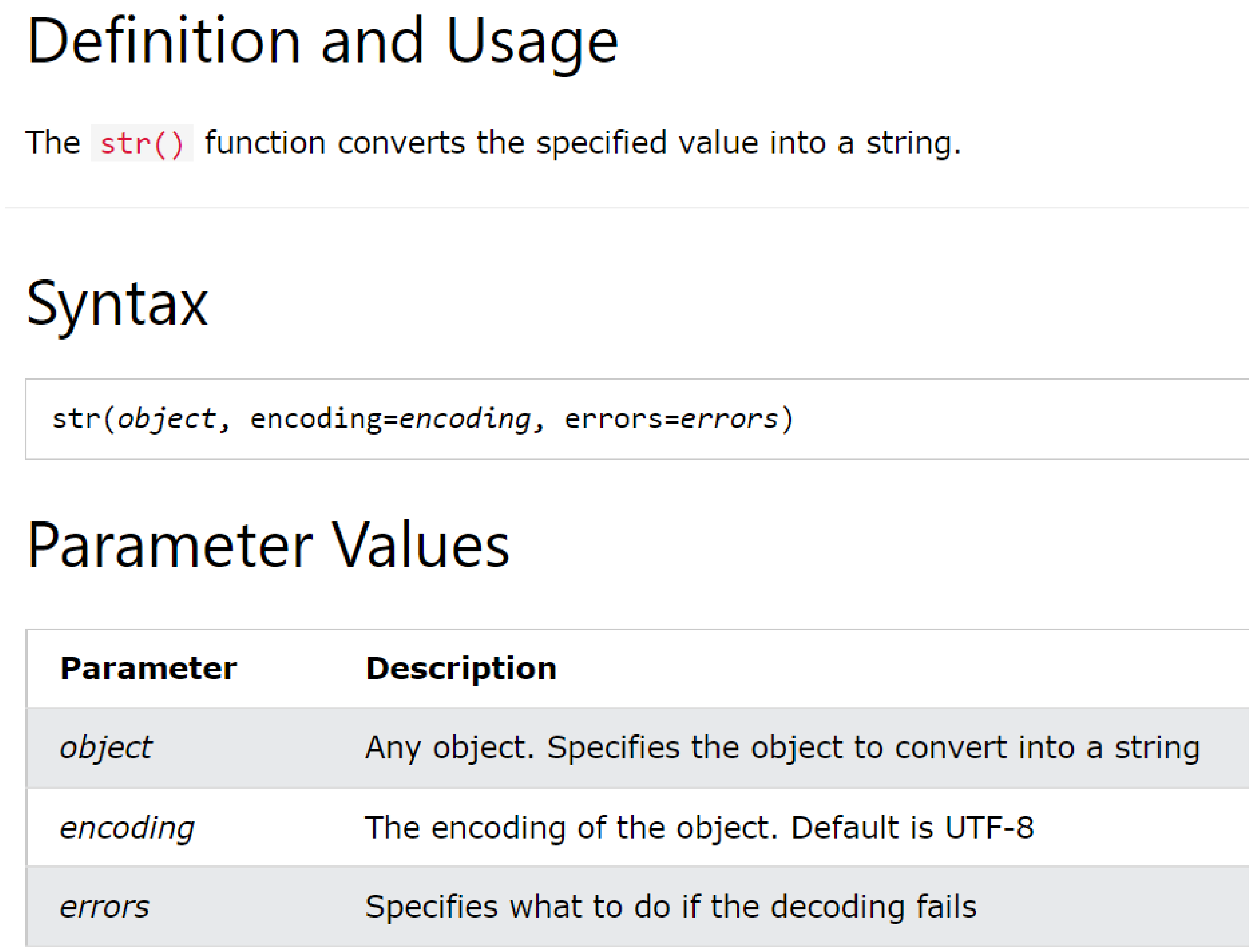
```
print('Hi'+"Python"+1003, sep="$")
```

ⓘ Start presenting to display the poll results on this slide.



		Built-in Functions		
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

https://docs.python.org/3/library/functions.html



Definition and Usage

The `str()` function converts the specified value into a string.

Syntax

```
str(object, encoding=encoding, errors=errors)
```

Parameter Values

Parameter	**Description**
object	Any object. Specifies the object to convert into a string
encoding	The encoding of the object. Default is UTF-8
errors	Specifies what to do if the decoding fails

https://www.w3schools.com/python/ref_func_str.asp

slido

What is the output of the following code? print('Hi'+"Python"+str(1003), sep="$")

ⓘ Start presenting to display the poll results on this slide.

Standard Library Functions

- Standard library: library of pre-written functions that comes with Python
    - *Library functions* perform tasks that programmers commonly need
        - Example: `print`, `input`, `math.sin`
        - Viewed by programmers as a "black box"
- Some library functions built into Python interpreter
    - To use, just call the function
    - Example: `print, input, str`

IMPORT

Standard Library Functions and the `import` Statement

```
import math
radiusString = input("Enter the radius of your circle:")
radiusFloat = float (radiusString)
circumference = 2 * math.pi * radiusFloat
area = math.pi * radiusFloat * radiusFloat
```

Standard Library Functions and the `import` Statement (cont'd.)

- Modules: files that stores functions of the standard library
  - Help organize library functions not built into the interpreter
  - Copied to computer when you install Python
- To call a function stored in a module, need to write an `import` statement
  - Written at the top of the program
  - Format: `import` *`module_name`*
- Dot notation: notation for calling a function belonging to a module
  - Format: `module_name.function_name()`

Standard Library Functions and the `import` Statement (cont'd.)

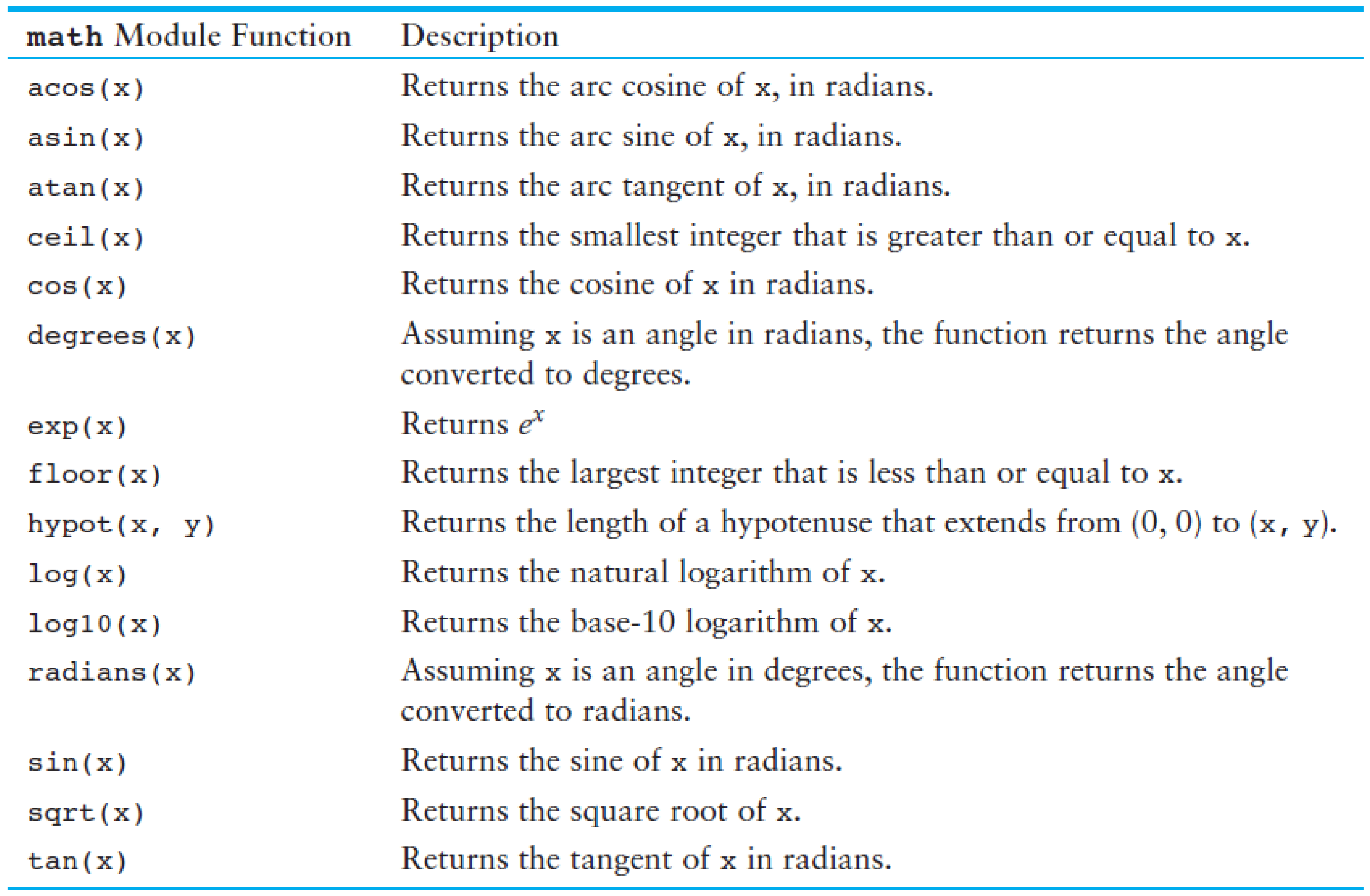
Modules: files that stores functions of the standard library	Help organize library functions not built into the interpreter Copied to computer when you install Python
To call a function stored in a module, need to write an import statement	Written at the top of the program Format: import *module_name*
Dot notation: notation for calling a function belonging to a module	Format: module_name.function_name()

The `math` Module

- `math` module: part of standard library that contains functions that are useful for performing mathematical calculations
  - Typically accept one or more values as arguments, perform mathematical operation, and return the result
  - Use of module requires an `import math` statement

The `math` Module (cont'd.)

Table 5-2 Many of the functions in the `math` module



`math` Module Function	Description
`acos(x)`	Returns the arc cosine of `x`, in radians.
`asin(x)`	Returns the arc sine of `x`, in radians.
`atan(x)`	Returns the arc tangent of `x`, in radians.
`ceil(x)`	Returns the smallest integer that is greater than or equal to `x`.
`cos(x)`	Returns the cosine of `x` in radians.
`degrees(x)`	Assuming `x` is an angle in radians, the function returns the angle converted to degrees.
`exp(x)`	Returns e^x
`floor(x)`	Returns the largest integer that is less than or equal to `x`.
`hypot(x, y)`	Returns the length of a hypotenuse that extends from (0, 0) to (`x, y`).
`log(x)`	Returns the natural logarithm of `x`.
`log10(x)`	Returns the base-10 logarithm of `x`.
`radians(x)`	Assuming `x` is an angle in degrees, the function returns the angle converted to radians.
`sin(x)`	Returns the sine of `x` in radians.
`sqrt(x)`	Returns the square root of `x`.
`tan(x)`	Returns the tangent of `x` in radians.

The import Statement

- import <module_name>
  - module_name.functionName()

```
import math
y = math.log(4)
```

- from <module_name> import <functionName(s)>
  - functionName()

```
>>>from sense_hat import SenseHat
>>>sense = SenseHat()
>>>sense.show_message("Hello NTU")
```

The `math` Module (cont'd.)

The math module defines variables pi and e, which are assigned the mathematical values for *pi* and *e*

- Can be used in equations that require these values, to get more accurate results

Variables must also be called using the dot notation

- Example:
  - circle_area = math.pi * radius**2

COMMON MISTAKE

slido

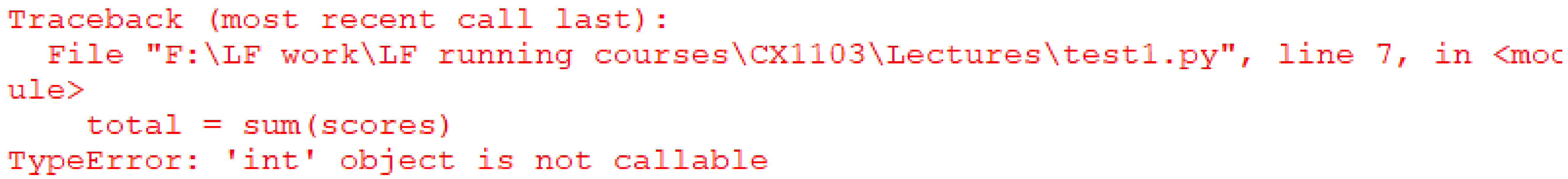
What is the output of the following code?

```
alice = 50
peter = 70
sum = alice + peter
print (sum, end = "#")
scores = [1,2,3,4]
total = sum(scores)
print (total)
```

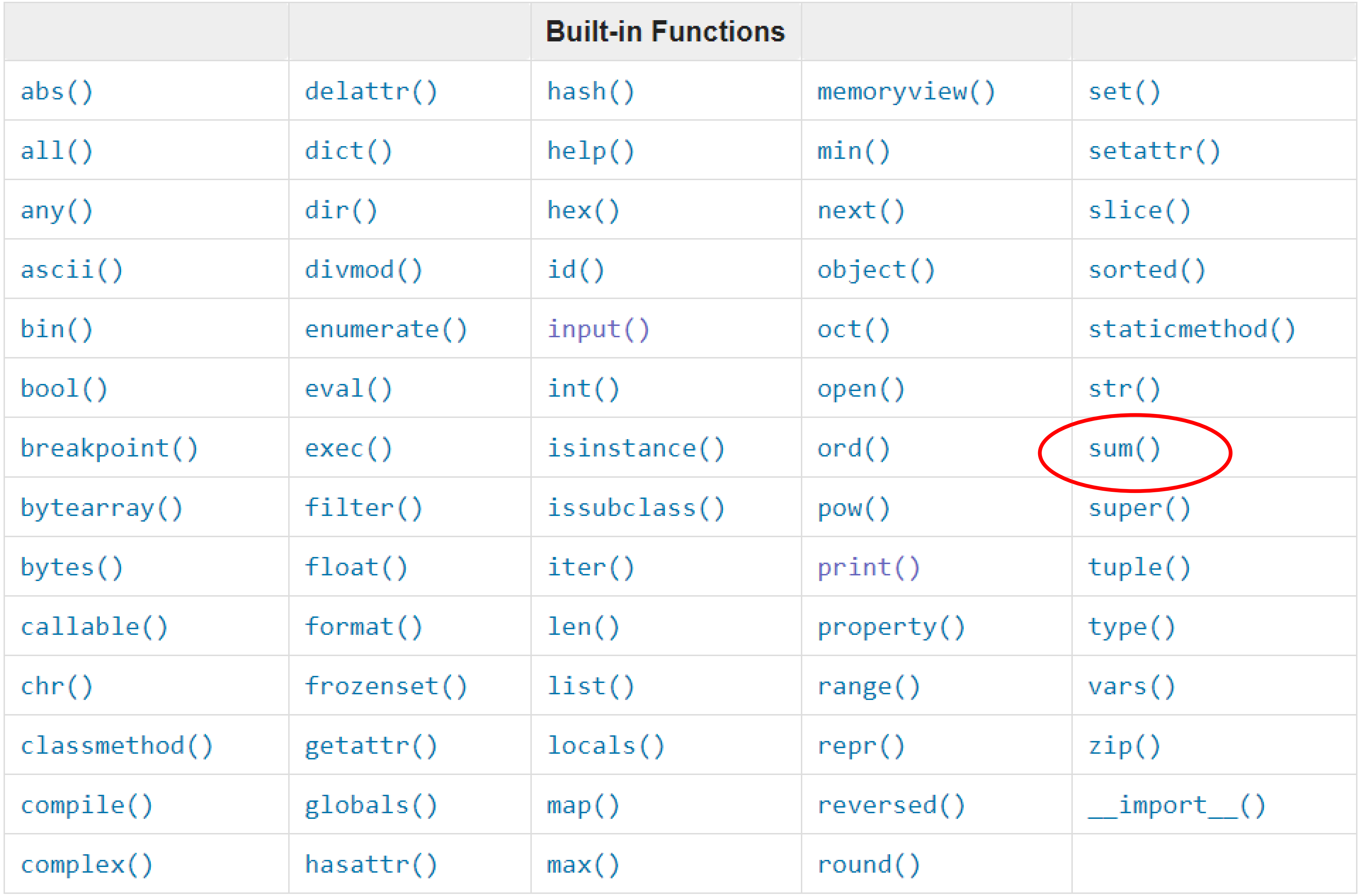
ⓘ Start presenting to display the poll results on this slide.

```
alice = 50
peter = 70
sum = alice + peter
print (sum, end = "#")
scores = [1,2,3,4]
total = sum(scores)
print (total)
```

```
120#
```



```
Traceback (most recent call last):
  File "F:\LF work\LF running courses\CX1103\Lectures\test1.py", line 7, in <moc
ule>
    total = sum(scores)
TypeError: 'int' object is not callable
```



		Built-in Functions		
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

https://docs.python.org/3/library/functions.html

slido

If you get syntax error for the following code which line of code we need to correct?

ⓘ Start presenting to display the poll results on this slide.