2019

Python input output

Natalia Novak

CUNY Bronx Community College

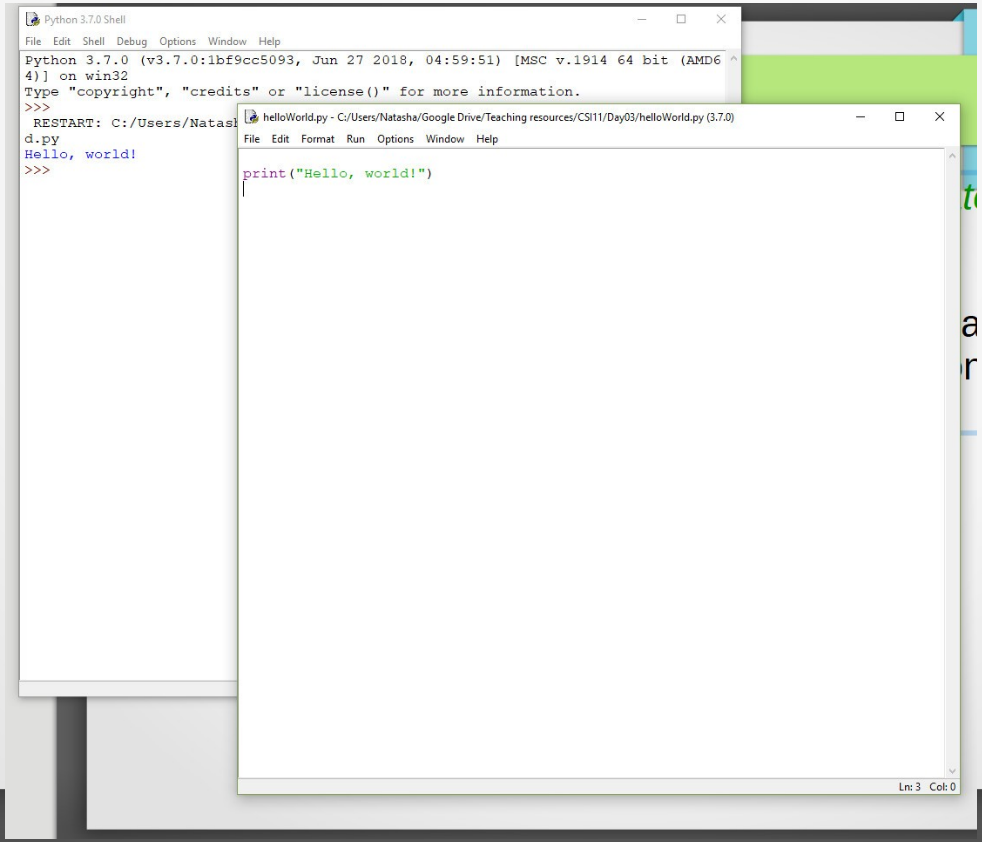
Topics to be discussed

- Development Environment
- Basic input and output
- Variables and assignments
- Python expressions
- Division and modulo
- Math module

Development Environment

Code development is usually done with an *Integrated Development Environment*, or *IDE*.

There are various IDEs, we will be using the official Python IDE that is distributed with the installation of Python, called *IDLE*.



Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
helloWorld.py - C:/Users/Natasha/Google Drive/Teaching resources/CSI11/Day03/helloWorld.py (3.7.0)
File Edit Format Run Options Window Help
Ln: 3 Col: 0

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD6
4)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:/Users/Natash
d.py
Hello, world!
>>>
```

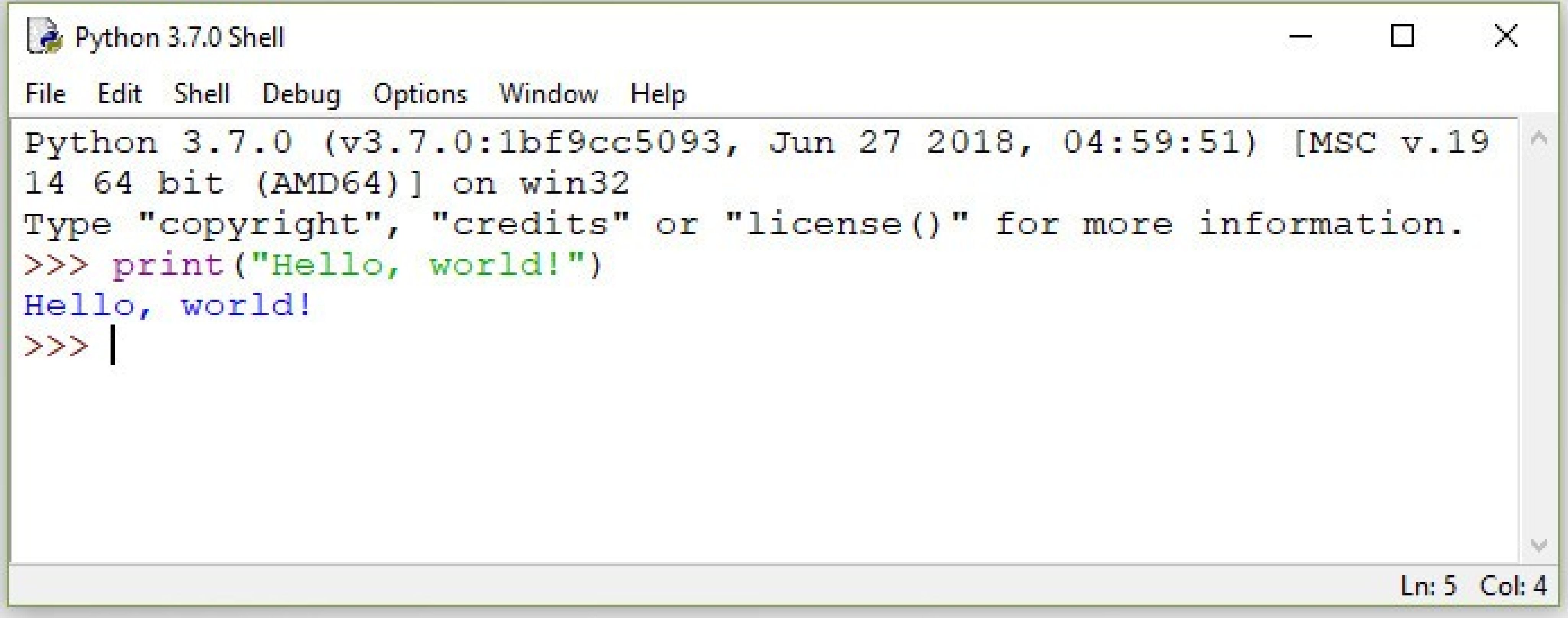
```
print("Hello, world!")
```

Development Environment

- Demonstrate IDLE
- Discuss Python Interpreter
- Discuss Python Shell (line prompt, …)
-
- Discuss File Editor (python files have extension **.py**)

My first program

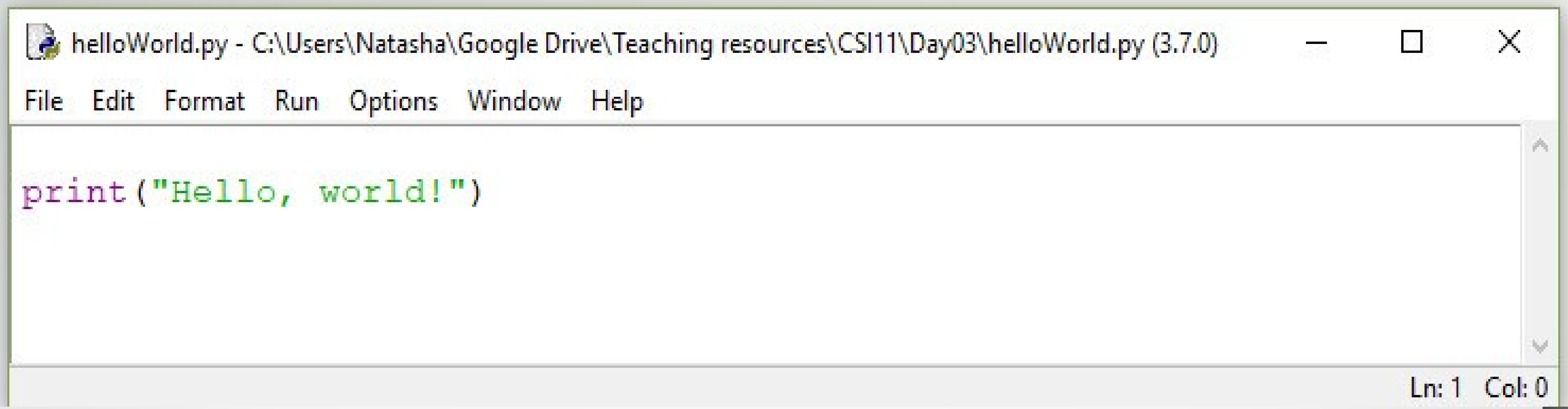
In Python shell:



```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.19
14 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("Hello, world!")
Hello, world!
>>> 
```

My first program

In IDLE’s file editor:



helloWorld.py - C:\Users\Natasha\Google Drive\Teaching resources\CSI11\Day03\helloWorld.py (3.7.0)
File Edit Format Run Options Window Help
Ln: 1 Col: 0

```
print("Hello, world!")
```

Save the program (File → Save) as a file named helloWorld.py

Then press F5 or go to Run → Run Module

Then check what you see in Python shell...

My second program

Create a new file (File → New File) and type in the following:

```
# this is my second program!

print("   *")
print(" * | *")
print("  \\*/")
print("*-***-*")
print("  /*\\")
print(" * | *")
print("   *")
print("Do you like my snowflake?")
```

Comment:
🟦 denotes one space (*whitespace*)

Save the program (File → Save) as mySecondProgram.py
Then press F5 or go to Run → Run Module
Then check what you see in Python shell...

My second program

```
# this is my second program!

print("   *")
print(" * | *")
print("  \\*/")
print("*-***-*")
print("  /*\\")
print(" * | *")
print("   *")
print("Do you like my snowflake?")
```

comment
(starts with #)

`print()` method displays variables or expression values

text enclosed in quotes is known as a *string literal*

Each print statement will output on a *new line*,
unless directed otherwise by a previous print statement

In-class practice:

In the Python shell type in the following commands/ instructions (after each instruction, hit **Enter** key) and observe the result:

```
>>> print("4")

>>> print(4)

>>> print("Alexa")

>>> print(Alexa)

>>> print("3"*5)

>>> print(3*5)

>>> print("2*8=",2*8)
```

In-class practice (continues):

In the Python shell type in the following commands/ instructions (after each instruction, hit **Enter** key) and observe the result:

```
>>> name ="Peter"
>>> print("Hello",name)

>>> print("Hello",name,", how are you?")

>>> print("Hello",name,", how are you?)


>>> x = 8
>>> y = 20
>>> print(x*y)
```

In-class practice (continues):

In the Python shell type in the following commands/ instructions (after each instruction, hit **Enter** key) and observe the result:

```
>>> print("Hello, \t how are you?")

>>> print("Hello! \n It is hot today, isn’t it?")

>>> print("\")

>>> print("\\")

>>> print("\\"*10)

>>> print("\\   "*10)
```

My third program

Create a new file (File → New File) and type in the following:

```
# this is my third program!

name = input("Enter your name, please:")

print("*"*40)
print("Nice to meet you,",name,"!")
print("The weather is wonderful today,
isn't it?")
print("*"*40)
```

Save the program (File → Save) as myThirdProgram.py
Then press F5 or go to Run → Run Module
Then check what you see in Python shell...

My third program

Create a new file (File → New F
type in the following:

```
# this is my thi

name = input("Enter your name, please:")

print(
print("N                                me,"!")
print("T                                ful today,
isn't it'
print("*"*40)
```

input() *function/method* will read text entered by the user, and assign the entered text to the name variable

variable
(named reference where the information is stored)

Save the program (File → Save) as myThirdProgram.py
Then press F5 or go to Run → Run Module
Then check what you see in Python shell...

Programs and terminology

A *computer program* mostly consists of a series of commands/instructions, called *statements*.

Each statement usually appears on its own line.

In a program we can see:

- *expressions* (code that return a value when evaluated)
  - `x * 5`
- *assignment statements* (using the = symbol)
  - `y = x * 5`
- *print() statements* (displays variables, or expression values, or string literals)
  - `print(“My name is”,name)`
- and many other things we will learn later

My fourth program

Visit our web-site:
go to ... page
scroll down to the date ...
right click on the *myFourthProgram.py*

choose *Save link as ...*
navigate to *Documents* folder, click on Save button

My fourth program

Go over the program

Variables and Assignments

Consider the following code fragment:

```
x = 10
y = x+2
z = x-y

x = 5
```

assignment statements

variables (*identifiers, names*)

An *identifier* (*name*), is a sequence of letters (a-z, A-Z), underscores (_), and digits (0-9), and must start with a letter or an underscore.

Python is case sensitive, meaning upper and lower case letters differ.

`myFriendsName`   `_counter`   `n4`

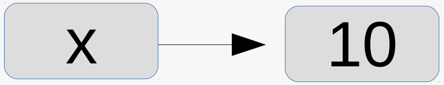
~~`4toGo`~~   ~~`it’s`~~   ~~`$hk`~~   ~~`pl-7h`~~

Variables, Assignments and Objects

Consider the following code fragment:

```
x = 10
y = x+2
z = x-y

x = 5
```

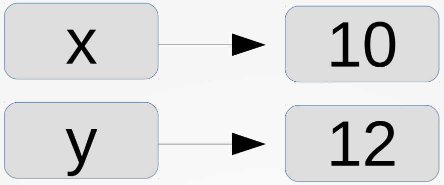


Variables, Assignments and Objects

Consider the following code fragment:

```
x = 10
y = x+2
z = x-y

x = 5
```

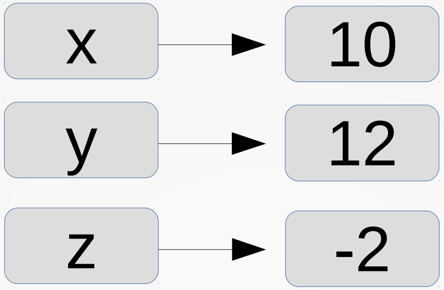


Variables, Assignments and Objects

Consider the following code fragment:

```
x = 10
y = x+2
z = x-y

x = 5
```

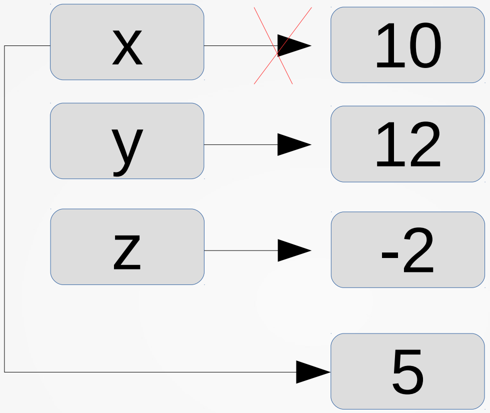


Variables, Assignments and Objects

Consider the following code fragment:

```
x = 10
y = x+2
z = x-y

x = 5
```



Variables, Assignments and Objects

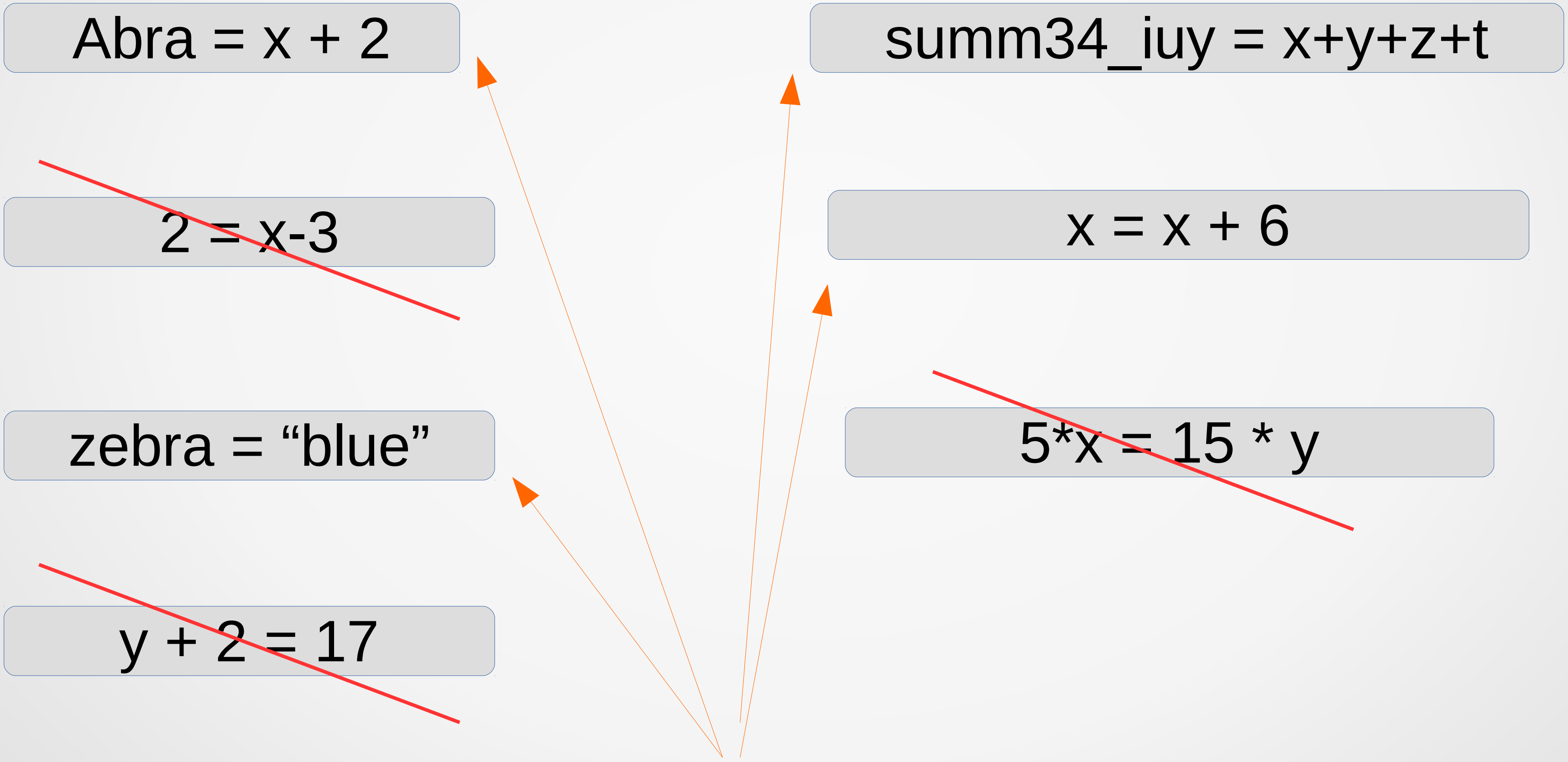
Consider the following assignment statements:

```
Abra = x + 2
2 = x-3
zebra = “blue”
y + 2 = 17
summ34_iuy = x+y+z+t
x = x + 6
5*x = 15 * y
```

Which ones of them are valid assignment statements?

Variables, Assignments and Objects

Consider the following assignment statements:



Abra = x + 2

~~2 = x-3~~

zebra = “blue”

~~y + 2 = 17~~

summ34_iuy = x+y+z+t

x = x + 6

~~5*x = 15 * y~~

Which ones of them are valid assignment statements?

Data types

By now we saw three types of data:

integers	1, 4, -16
real numbers (floating-point numbers)	1.2, -1.8, 0.54
strings	“Peter”, “Hello, how are you?”

Python has built-in function that allows us to get the type of an object: `type()`

Data types

By now we saw three types of data:

integers	1, 4, -16	`int`
real numbers (floating-point numbers)	1.2, -1.8, 0.54	`float`
strings	“Peter”, “Hello, how are you?”	`string`

Python has built-in function that allows us to get the type of an object: `type()`

In-class practice:

In the Python shell type in the following assignment statements and instructions and observe the result:

```
>>> name ="Peter"
>>> type(name)

>>> x = 287
>>> type(x)
>>> x = chr(x)
>>> type(x)

>>> y = 2.87
>>> type(y)
```

the built-in function `chr()` converts from integer to string

We will see later in the course the examples when this built-in function is useful.

Arithmetic Expressions

We would like to be able to work with algebraic expressions such as

$2x+5$ or $3x^2\text{-}6y^3+1$ or $\dfrac{x-y}{x+2}$ ….

Arithmetic operator	Description	Python operator
+	addition x+5	+
×	multiplication 2×a	*
÷	multiplication a÷7	/
-	subtraction x-10	-
x^2	exponent x^2	**

Arithmetic Expressions

let’s see some conversions from math to Python:

Algebraic expression in math	Algebraic expression in Python
$2x+5$	`2*x+5`
$3x^2$-$6y^3+1$	`3*x*x-6*y**3+1`
$(a+b+c) \div 3$	`(a+b+c)/3`
y-$2(x+9)$	`y-2*(x+9)`
x^8	`x**8`
$\frac{x-y}{x+2}$	`(x-y)/(x+2)`

In-class practice:

In the Python File Editor finish the program (follow the comments):

```
a = int(input(“Enter an integer value:”))
b = int(input(“Enter another integer value:”)
c = int(input(“Enter the last integer value:”)

# find the average of a,b, and c
# and display it

# find the product of a,b, and c and store
# the result in variable z
```

Python expressions

```
my_pay = base_pay + overtimeRate * numberOfHours

my_pay = base_pay + overtimeRate * numberOfHours


my_pay=base_pay+overtimeRate*numberOfHours
```

- may be it is a little bit less “readable”?

Python expressions

my_pay = base_pay + overtimeRate * numberOfHours

my_pay = base_pay + overtimeRate * numberOfHours

~~my_pay=base_pay+overtimeRate*numberOfHours~~

- may be it is a little bit less “readable”?

Python expressions

No commas in numbers!

1,876,904.76

Python expressions

No commas in numbers!

~~1,876,904.76~~ → 1876904.76

Python expressions

No commas in numbers!

~~1,876,904.76~~ → 1876904.76

We have compound operators!

x = x + 1 ↔ x += 1

n = n *100 ↔ n *= 100

a = a - 7 ↔ a -= 7

k = k / 5 ↔ k /= 5

Division and modulo

The division operator / performs division and returns a *floating-point number*.

Examples:

```
>>> 40 / 5
8.0

>>> 8 / 10
0.8
```

Division and modulo

The quotient of the division can be found using the *floored division operator //*

The resulting value is an integer type if both operands are integers; if either operand is a float, then a float is returned.

Examples:

```
>>> 4 // 5
0

>>> 4.0 // 5.0
0.0

>>> 8.0 // 5.0
1.0
```

Division and modulo

The *modulo* operator (%) evaluates the remainder of the division of two integer operands.

Examples:

56 % 10 is 6.   Reason: 5 tens fit into 56, 6 is left (reminder)
9 % 9 is 0.   Reason: 1 nine fits into 9, nothing is left
5 % 2 is 1.   Reason: 2 twos fit into 5, 1 is left (remainder)

```
>>> 56 % 10
6

>>> 9%9
0

>>> 5 % 2
1
```

In-class practice

In the Python shell type in the following commands/ instructions (after each instruction, hit **Enter** key) and observe the result, then do the assignment:

```
>>> my_age = 192
>>> his_age = 827
>>> my_age += his_age
>>> my_age
>>> his_age
```

Stop! Think: What happened there?

Next: type in a print statement that will display:
“I’m … years old and he is … years old”
In the … space should be displayed the values of `my_age` and `his_age` variables.

In-class practice

Create a new file (File → New File), save it as qr.py and write a program that does the following:

1) gets two numbers from the user (x and d) – *use input()*
2) Finds the quotient and remainder of the division x 🕒 d
3) Displays the division, the quotient and the remainder of the division – *use print()*

Then press F5 or go to Run → Run Module
Then check what you see in Python shell...

Here is how it might look in the Python shell when the program is run:

```
Enter the dividend: 18
Enter the divisor: 7
The quotient of 18 / 7 is 2 and the remainder is 4.
```

Math module

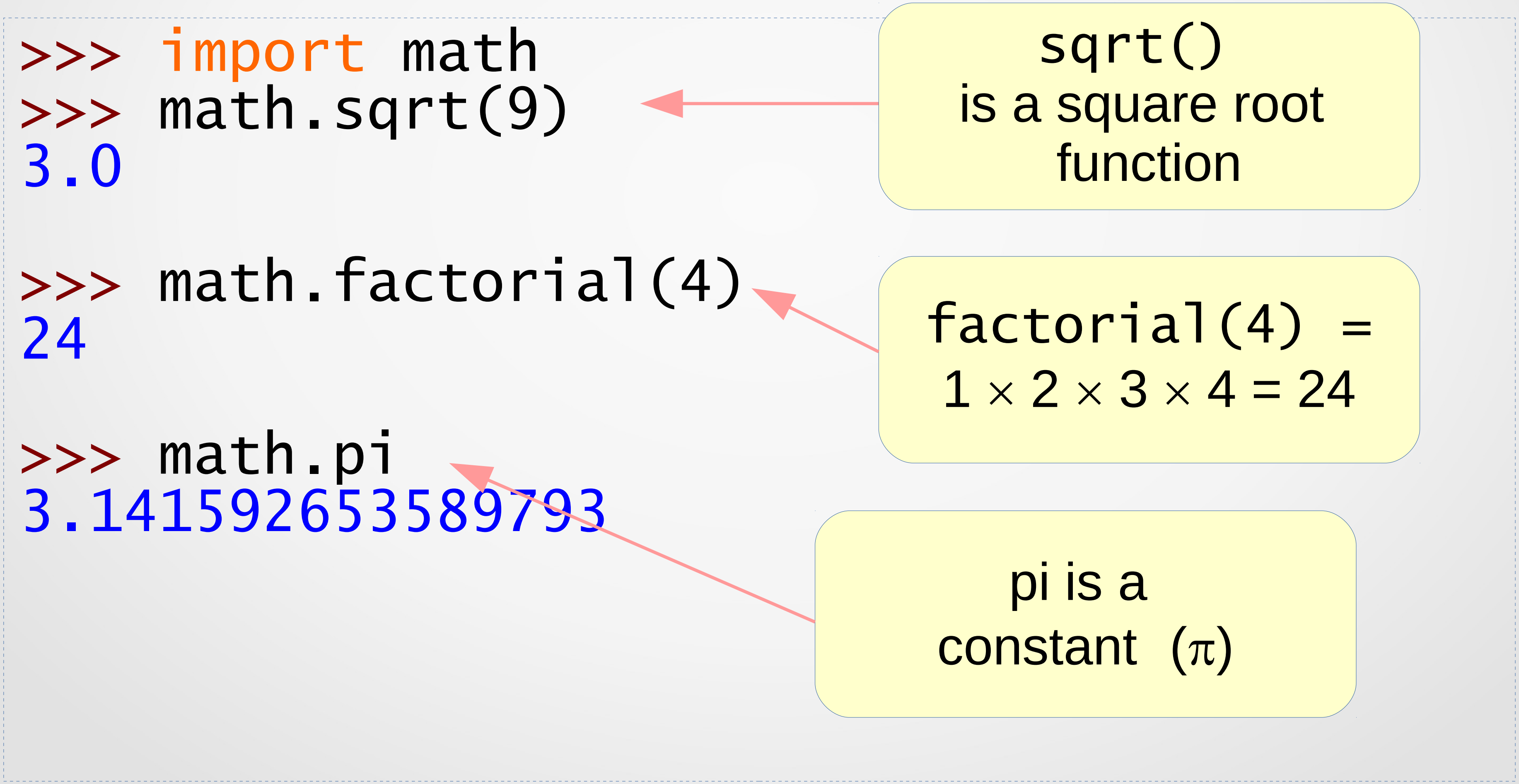
While basic math operations like + or * are sufficient for some computations, programmers sometimes wish to perform more advanced math operations such as computing a square root.

Python comes with a standard math module to support such advanced math operations.

A *module* is Python code located in another file. The programmer can import the module for use in their own file, or in an interactive interpreter.

Math module

The programmer can *import* the module for use inan interactive interpreter (Python shell)



```
>>> import math
>>> math.sqrt(9)
3.0

>>> math.factorial(4)
24

>>> math.pi
3.141592653589793
```

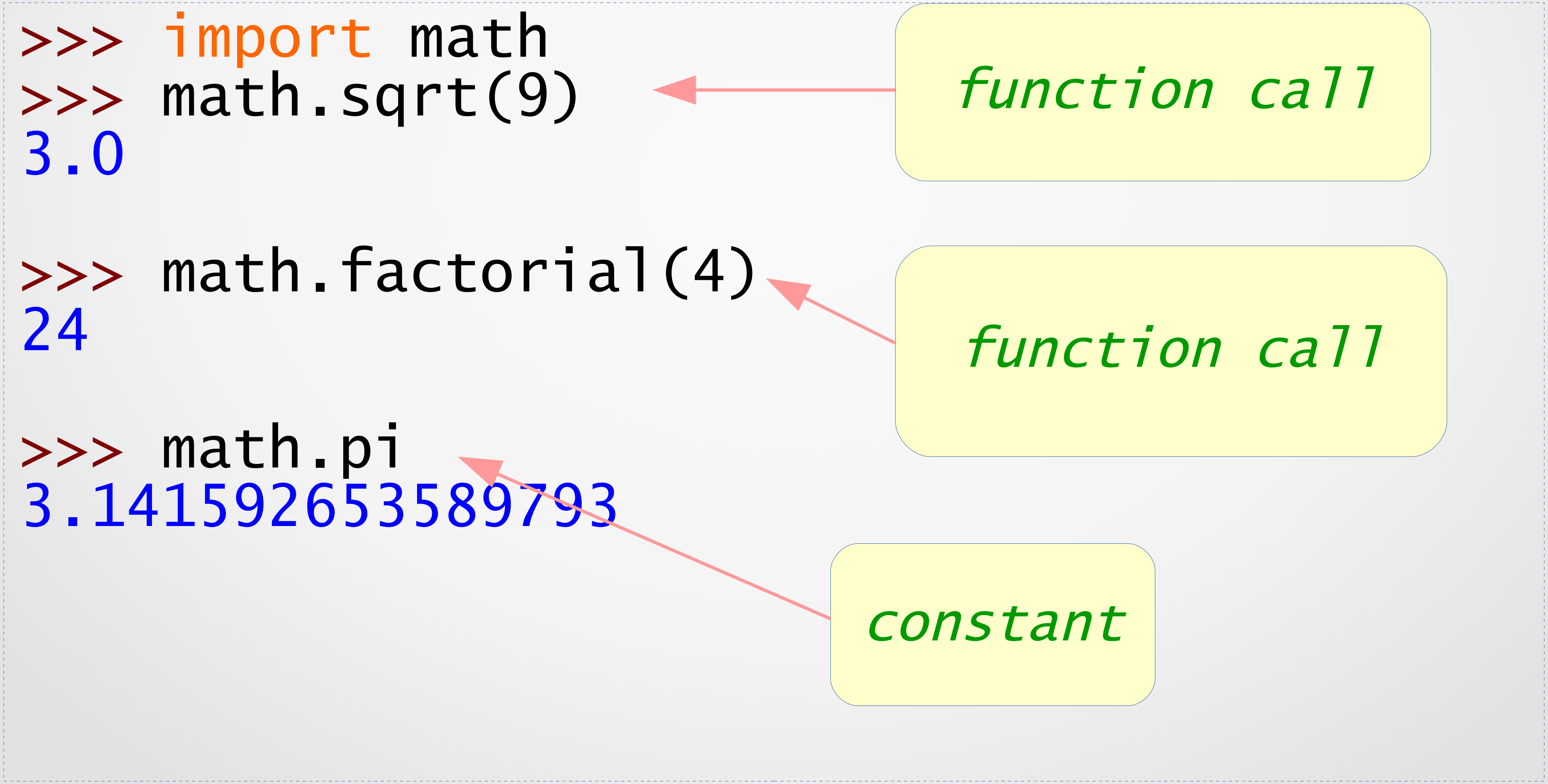
sqrt()
is a square root
function

factorial(4) =
1 × 2 × 3 × 4 = 24

pi is a
constant (π)

Math module

The programmer can *import* the module for use inan interactive interpreter (Python shell)



```
>>> import math
>>> math.sqrt(9)
3.0

>>> math.factorial(4)
24

>>> math.pi
3.141592653589793
```

function call

function call

constant

Math module

The programmer can *import* the module for use inan interactive interpreter (Python shell)

```
>>> import math *
>>> sqrt(9)
3.0

>>> factorial(4)
24

>>> pi
3.141592653589793
```

Math module

I can also use Python File Editor: type in the program, save it and run it!

```
import math

radius = float(input("please enter the
radius of a circle:"))

C = 2 * math.pi * radius  # circumference
A = math.pi * radius ** 2 # area

print("The circumference of the circle of
radius",radius,"is",C)
print("The area of the circle of
radius",radius,"is",A)
```

Go to our web-site (Notices page) – download file circleMath.py

This OER material was produced as a result of the CS04ALL CUNY OER project.