

# GET214: **COMPUTING AND SOFTWARE ENGINEERING**

Lesson 2

**PYTHON EXPRESSIONS**

# Learning outcomes

At the end of the lesson, students should be able:

1. Use a Python shell to run statements and expressions interactively
2. Explain how the interpreter uses implicit type conversion
3. Use explicit type conversion with `int()`, `float()`, and `str()`
4. Identify the data types produced by operations with integers, floats, and strings
5. Use operators and type conversions to combine integers, floats, and strings
6. Use the `round()` function to mitigate floating-point errors in output
7. Evaluate expressions that involve floor division and modulo
8. Use Built-in functions and constants defined in the `math` module

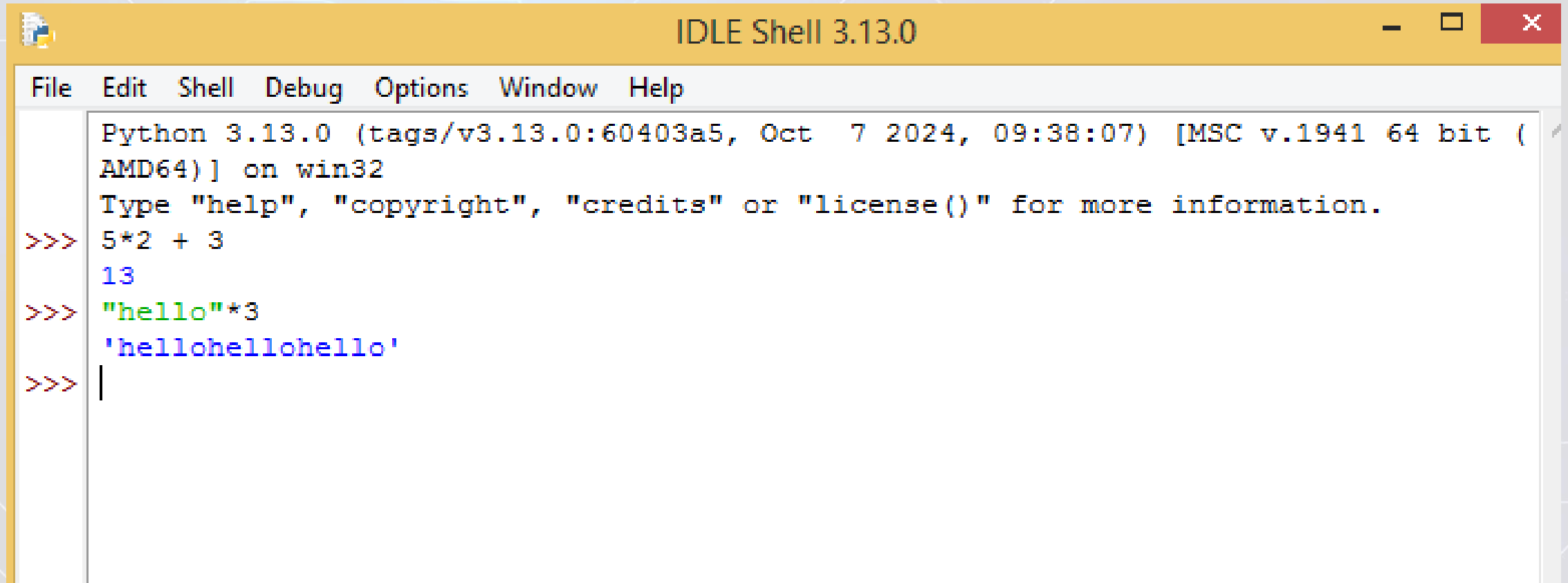
# What is an expression?

- An **expression** represents a single value to be computed.
- For instance, the expression  $3 * x - 5$  evaluates to 7 when  $x$  is 4.
- Expressions are often a combination of literals, variables, and operators.
- In the previous example, 3 and 5 are literals,  $x$  is a variable, and  $*$  and  $-$  are operators.
- Expressions can be made of strings or strings and numbers.
- Example: "Hello" + "World" evaluates to "HelloWorld".
- "hey"\*3 evaluates to "heyheyhey".

# The Python Shell

- A **shell**, also called a console or terminal, is a program that allows direct interaction with an interpreter.
- The compiler usually runs an entire program all at once. But the interpreter can run one line of code at a time within a Python shell.
- Python's proprietary development environment is called the IDLE.
- Python shell is the first window that shows up whenever the idle is started
- The shell executes any expression typed in front of the prompt `>>>` at the press of the **Enter** key

# The Python Shell of python 3.13.0



```
Python 3.13.0 (tags/v3.13.0:60403a5, Oct 7 2024, 09:38:07) [MSC v.1941 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 5*2 + 3
13
>>> "hello"*3
'hellohellohello'
>>> |
```

# REPL

- The acronym REPL (pronounced "rep ul") is often used when referring to a shell. **REPL** stands for "read-eval-print loop," which describes the repetitive nature of a shell:
  1. **R**ead/input some code
  2. **E**valuate/run the code
  3. **P**rint any results
  4. **L**oop back to step 1
- Most shells maintain a history of every line of code the user types.
- Pressing **Alt+P** key on the keyboard displays the history.

# Activity 2.1: Exploring the Python Shell

Running code interactively is a great way to learn how Python works. Open the shell in python IDLE and type in the following code one line at a time, to see the results.

- `x = 5`
- `3*x - 5`
- `3 * (x-5)`
- `x`
- `type(1)`
- `type('1')`
- `str(1)`
- `int('1')`
- `abs(-5)`
- `abs(5)`
- `len("Yo")`
- `len("HoHo")`
- `round(9.49)`
- `round(9.50)`

**Note:** These functions (type, str, int, len, and round) will be explored in more detail later in the lesson.



# Activity 2.2: Debugging Code

Open a python shell and type the following two lines of code

```
>>> x = 123
```

```
>>> y = 456
```

- Making mistakes is common while typing in a shell.
- The lines in the next slide include typos and other errors.
- For each line:
  - (1) run the line in a shell to see the result,
  - (2) press the up arrow to repeat the line, and
  - (3) edit the line to get the correct result.



```
>>> print("Easy as", X)
>>> print("y divided by 2 is", y / 0)
>>> name = input("What is your name? ")
>>> print(name, "is", int(name), "letters long.")
>>> print("That's all folks!")
```

The expected output, after correcting typos, should look like:

- Easy as 123
- y divided by 2 is 228.0
- (no error/output)
- Stacie is 6 letters long.
- That's all folks!

# Type Conversion

- Changing from one type of data to another is a common practice in programming.
- The `input()` function, by default reads in string data, even when the user enters numeric data.
- Thus, the programmer must explicitly convert such data to a numeric type using **`int()`** or **`float()`** functions if some mathematical computations are to be carried out with it.
- There may be need to convert also from one numeric type to another.
- This can be done implicitly or explicitly.

# Implicit Conversion

- Common operations update a variable such that the variable's data type needs to be changed.
- For instance, a GPS first assigns distance with 250, an integer.
- After a wrong turn, the GPS assigns distance with 252.5, a float.
- The Python interpreter uses **implicit type conversion** to automatically convert one data type to another.
- Once distance is assigned with 252.5, the interpreter will convert distance from an integer to a float without the programmer needing to specify the conversion.

# Questions to attempt from reference text

The Python logo, consisting of two interlocking snakes, one light blue and one light yellow, is centered in the background.

Page 43

# Explicit Conversion

- A programmer often needs to change data types to perform an operation.
- For example, a program should read in two values using `input()` and sum the values.
- Remember `input()` reads in values as strings.
- A programmer can use **explicit type conversion**, *for compatible data types*, to convert one data type to another.

# Type Conversion Functions

- **int()** converts a data type to an integer.
  - Any fractional part is truncated (removed without approximation).  
Eg. `int(5.9)` produces 5.
- **float()** converts a data type to a float.  
Eg. `float(2)` produces 2.0.
- **str()** converts a data type to a string.  
Eg. `str(3.14)` produces "3.14".



# Activity 2.3: Grade Average Computation

- The program in the next slide computes the average of three predefined exam grades and prints the average twice.
- Improve the program to read the three grades from input and print the average first as a float, and then as an integer, using explicit type conversion.
- Ignore any differences that occur due to rounding.

```
exam_1 = 93.0
```

```
exam_2 = 84.0
```

```
exam_3 = 85.5
```

```
average = (exam_1 + exam_2 + exam_3) / 3
```

```
print(average)
```

```
print(average)
```

# Activity 2.4: Product as float

- The program in the next slide should read in the ounces of water the user drank today and compute the number of cups drank and the number of cups left to drink based on a daily goal. Assume a cup contains 8 ounces. Fix the code to calculate cups\_drunk and cups\_left and match the following:
  - ounces is an integer representing the ounces the user drank.
  - cups\_drunk is a float representing the number of cups of water drank.
  - cups\_left is an integer representing the number of cups of water left to drink (rounded down) out of the daily goal of 8 cups.

```
ounces = input()
print('Input ounces:', ounces, ', type:', type(ounces))
print(cups_drunk, 'cups drank, type:', type(cups_drunk))
print('About', cups_left, 'cups left to drink, type:',
type(cups_left))
```

*(The last two lines should be typed as a single line)*

# Questions to attempt from reference text

The Python logo, consisting of two interlocking snakes, one light blue and one light yellow, is centered in the background.

Page 44

# Mixed data type conversion

- Programmers often need to combine numbers of different data types. Eg. The program below computes the total for an online shopping order:

```
quantity = int(input())  
price = float(input())  
total = quantity * price  
print(total)
```

- quantity is an integer, and price is a float.
- So what is the data type of total?
- For input 3 and 5.0, total is a float, and the program prints 15.0.



# Questions to attempt from reference text

The Python logo, consisting of two interlocking snakes, one blue and one yellow, is centered in the background.

Page 46-47

# Floating Point round() function

- Python's round() function is used to round a floating-point number to a given number of decimal places.
- The function requires two arguments.
  - The first argument is the number to be rounded.
  - The second argument decides the number of decimal places to which the number is rounded.
- If the second argument is not provided, the number will be rounded to the closest integer.

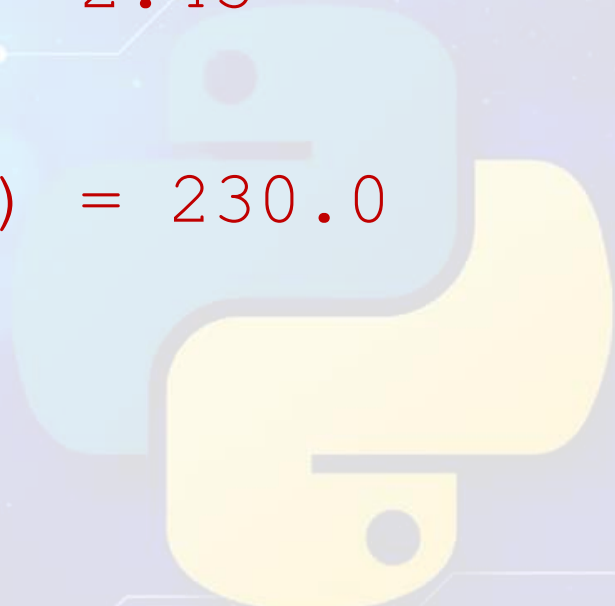
# Use of the round() function

```
round(2.451, 2) = 2.45
```

```
round(2.451) = 2
```

```
round(234.631, -1) = 230.0
```

```
Round(312.63, 0)
```



# Activity 2.6: Area of a Triangle

- Complete the following steps to calculate a triangle's area, and print the result of each step.
  - The area of a triangle is  $\frac{bh}{2}$ , where b is the base and h is the height.
1. Calculate the area of a triangle with base = 7 and height = 3.5.
  2. Round the triangle's area to one decimal place.
  3. Round the triangle's area to the nearest integer value.

# Dividing Integers

- Python provides three ways to divide numbers:
- True division (/) converts numbers to floats before dividing.  
Eg.  $7 / 4$  becomes  $7.0 / 4.0$ , resulting in  $1.75$
- Floor division (//) computes the quotient, or the number of times divided.

Eg.  $7 // 4$  is 1 because 4 goes into 7 one time, remainder 3

- The modulo operator (%) computes the remainder.

Eg.  $7 \% 4$  is 3.

- Note: *The % operator is traditionally pronounced "mod" (short for "modulo"). For instance, when reading  $7 \% 4$  out loud, a programmer would say "seven mod four".*

# Activity 2.7: Arrival Time

- Having a mobile device can be a lifesaver on long road trips. Programs like Google Maps find the shortest route and estimate the time of arrival. The time of arrival is based on the current time plus how long the trip will take.
- Write a program that:
  - (1) inputs the current time and estimated length of a trip,
  - (2) calculates the time of arrival, and
  - (3) outputs the results in hours and minutes.



- Your program should use the following prompts (user input in bold):

Current hour (0-23)? **13**

Current minute (0-59)? **25**

Trip time (in minutes)? **340**

- In this example, the current time is 13:25 (1:25pm). The trip time is 340 minutes (5 hours and 40 minutes). 340 minutes after 13:25 is 19:05 (7:05pm).
- Your program should output the result in the format on the next slide:

Arrival hour is 19  
Arrival minute is 5

- The arrival hour must be between 0 and 23.
- Example: Adding 120 minutes to 23:00 should be 1:00, not 25:00.
- The arrival minute must be between 0 and 59. Ex: Adding 20 minutes to 8:55 should be 9:15, not 8:75.
- Hint: Multiply the current hour by 60 to convert hours to minutes. Then, calculate the arrival time, in total minutes, as an integer.
- *Your code must not use Python keywords that you are yet to learn, such as if or while. The solution requires only addition, multiplication, division, and modulo.*

# Questions to attempt from reference text

The Python logo, consisting of two interlocking snakes, one blue and one yellow, is centered in the background.

Page 50, 52

# The math module

- Python comes with an extensive standard library of modules.
- A **module** is previously written code that can be imported in a program.
- The **import statement** defines a variable for accessing code in a module.
- Import statements often appear at the beginning of a program.
- A commonly used module in the standard library is the math module.
- This module defines functions such as `sqrt()` (square root).

## *Example 2.1:* Computing the Distance between two points.

- Given two points in a Cartesian plane with coordinates  $P(x_1, y_1)$  and  $Q(x_2, y_2)$ , the distance between  $P$  and  $Q$  is given by the relation
- $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$
- The following code obtains the coordinates of  $P$  and  $Q$  and computes the distance between them.

```
import math
x1 = float(input("Enter x1: "))
y1 = float(input("Enter y1: "))
x2 = float(input("Enter x2: "))
y2 = float(input("Enter y2: "))
distance = math.sqrt((x2-x1)**2 + (y2-y1)**2)
print("The distance is", distance)
```

Enter x1: 1

Enter y1: 3

Enter x2: 5

Enter y2: 6

The distance is 5.0



# Constants and Methods in the math module

- Popular mathematical constants are packaged into the python math module.
- Table 2.1 shows a few of them and their descriptions.
- Table 2.2 shows some methods in the python math module and their usages.

## Table 2.1: Constants in the math module

Constant	Value	Description
<code>math.e</code>	$e = 2.71828 \dots$	Euler's number: the base of the natural logarithm.
<code>math.pi</code>	$\pi = 3.14159 \dots$	The ratio of the circumference to the diameter of a circle.
<code>math.tau</code>	$\tau = 6.28318 \dots$	The ratio of the circumference to the radius of a circle. Tau is equal to $2\pi$ .

# Table 2.2: Methods in the math module

Function	Description	Examples
<b>Number-theoretic</b>		
<code>math.ceil(x)</code>	The ceiling of x: the smallest integer greater than or equal to x.	<code>math.ceil(7.4) → 8</code> <code>math.ceil(-7.4) → -7</code>
<code>math.floor(x)</code>	The floor of x: the largest integer less than or equal to x.	<code>math.floor(7.4) → 7</code> <code>math.floor(-7.4) → -8</code>
<b>Power and logarithmic</b>		
<code>math.log(x)</code>	The natural logarithm of x (to base e).	<code>math.log(math.e) → 1.0</code> <code>math.log(0) → ValueError: math domain error</code>

<code>math.log(x, base)</code>	The logarithm of $x$ to the given base.	<code>math.log(8, 2) → 3.0</code> <code>math.log(10000, 10) → 4.0</code>
<code>math.pow(x, y)</code>	$x$ raised to the power $y$ . Unlike the <code>**</code> operator, <code>math.pow()</code> converts $x$ and $y$ to type float.	<code>math.pow(3, 0) → 1.0</code> <code>math.pow(3, 3) → 27.0</code>
<code>math.sqrt(x)</code>	The square root of $x$ .	<code>math.sqrt(9) → 3.0</code> <code>math.sqrt(-9) →</code> ValueError: math domain error
<b>Trigonometric</b>		
<code>math.cos(x)</code>	The cosine of $x$ radians.	<code>math.cos(0) → 1.0</code> <code>math.cos(math.pi) → -1.0</code>
<code>math.sin(x)</code>	The sine of $x$ radians.	<code>math.sin(0) → 0.0</code> <code>math.sin(math.pi/2) → 1.0</code>
<code>math.tan(x)</code>	The tangent of $x$ radians.	<code>math.tan(0) → 0.0</code> <code>math.tan(math.pi/4) →</code> 0.999 (Round-off error; the result should be 1.0.)

# Activity 2.8: Quadratic Formula

- In algebra, a quadratic equation is written as  $ax^2 + bx + c = 0$ . The coefficients  $a$ ,  $b$ , and  $c$  are known values. The variable  $x$  represents an unknown value. For instance,  $2x^2 + 3x - 5$  has the coefficients  $a = 2$ ,  $b = 3$  and  $c = -5$ . The quadratic formula provides a quick and easy way to solve a quadratic equation for  $x$ :

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- The plus-minus symbol indicates the equation has two solutions. However, Python does not have a plus-minus operator.

- To use this formula in Python, the formula must be separated:

$$x1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$
$$x2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

- Write the code for the quadratic formula in the program below.
- Test your program using the following values for a, b, and c:

Provided input			Expected output	
a	b	c	x1	x2
1	0	-4	2.0	-2.0
1	2	-3	1.0	-3.0
2	1	-1	0.5	-1.0



# Activity 2.9: Cylinder Formulas

- In geometry, the surface area and volume of a right circular cylinder can be computed as follows:

$$A = 2\pi rh + 2\pi r^2$$
$$V = \pi r^2 h$$

- Write the code for these two formulas in the program below.
- Hint:* Your solution should use both `math.pi` and `math.tau`. Test your program using the following values for `r` and `h`:

Provided Input		Expected output	
r	h	area	volume
0	0	0.0	0.0
1	1	12.57	3.14
1	2	18.85	6.28
2.5	4.8	114.67	94.25
3.1	7.0	196.73	211.33

# Questions to attempt from reference text

The Python logo, consisting of two interlocking snakes, one blue and one yellow, is centered in the background.

Page 57, 58