Lecture 3

Topics to be covered:

Chapter 5:
- Python expressions
- Division and modulo
- Math module
- Representing text

5.6 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”?

5.6 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”?

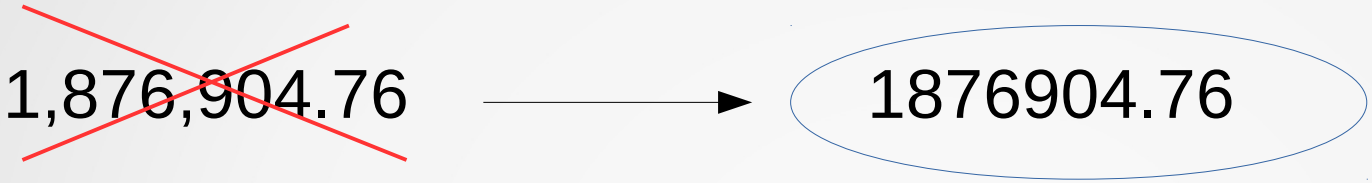
5.6 Python expressions

No commas in numbers!

1,876,904.76

5.6 Python expressions

No commas in numbers!



5.6 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
```

5.7 Division and modulo

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

Examples:

```
>>> 40 / 5
8.0

>>> 8 / 10
0.8
```

5.7 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
```

5.7 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

Refer to the handout
Work on problems 1-4

5.9 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.

5.9 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) =
$1 \times 2 \times 3 \times 4 = 24$

pi is a
constant (π)

5.9 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

5.9 Math module

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

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

>>> factorial(4)
24

>>> pi
3.141592653589793
```

5.9 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

5.10 Representing text

Unicode

String variables represent text.

Examples: the character 'k'
the word 'Pineapple'

5.10 Representing text

Unicode

String variables represent text.

Examples: the character 'k'
the word 'Pineapple'

Python uses Unicode to represent every possible character as a *unique number*, known as a code point.

Examples:
the character 'a' has the code point of 97 (decimal system)

the character ‘A’ has the code point of 65 (decimal system).

5.10 Representing text

Unicode

See our ZyBooks (Section 5.10) for the ASCII table.

And visit this page: https://www.ascii-code.com/

5.10 Representing text

ord and chr

`>>> ord('a')` returns the ASCII code of letter `a`

`>>> chr(97)` converts ASCII code of 97 to the corresponding letter (which is `a`)

5.10 Representing text

Escape sequences

Escape Sequence	Explanation	Example code	Output
\\	backslash (\)	print('\\home\\users\\')	\home\users\
\’	single quote (')	print('Name: Tom O\'Lee')	Tom O'Lee
\”	double quote (")	print("He said, \"Hello !\".")	He said, "Hello !".
\n	newline	print('My name...\nIs Joe')	My name... Is Joe
\t	tab (indent)	print('To-do list:\n\t1. vacuum')	To-do list: 1. vacuum

5.10 Representing text

Raw strings

Escape sequences can be ignored using a raw string.

A raw string is created by adding an 'r' before a string literal, as in r'this is a raw string'.

If we try this in Python Shell:

```
>>> str1 = "my \'favorite\' play"
>>> str2 = r"my \'favorite\' play"
>>> print(str1)
my 'favorite' play
>>> print(str2)
my \'favorite\' play
```

In-class practice

Refer to the handout
Work on problems 5-6