CS A131: Lecture 3

Nadia Ahmed

Orange Coast College, Computer Science

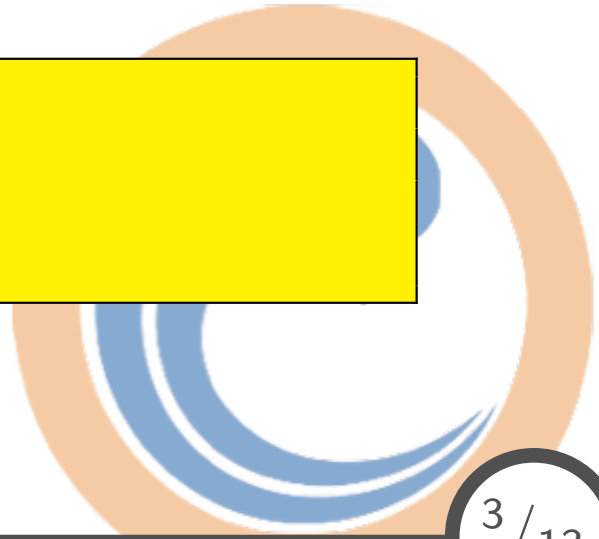
CS A131

Overview

Variables in Python

- A variable is a name that represents a value stored in the computer's memory
- Use an *assignment operator* to assign specific data to a variable name
- The variable name is always to the left of the =
- You cannot use a variable until you have assigned a value to it!
- Python is case-sensitive

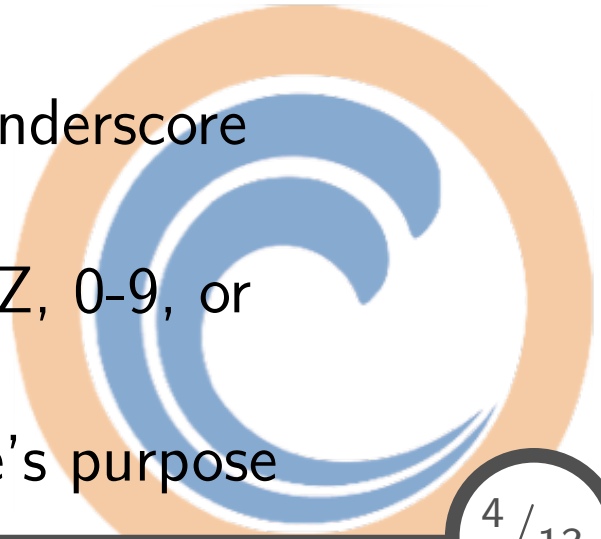
```
# syntax
# variable = expression

age = 25
```



Valid Variable Names

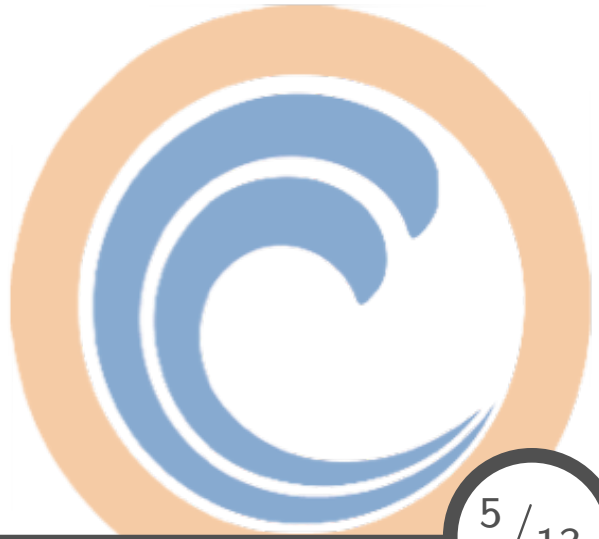
and	del	from	None	True
as	elif	global	nonlocal	try
assert	else	if	not	while
break	except	import	or	with
class	False	in	pass	yield
continue	finally	is	raise	
def	for	lambda	return	

- Python keywords may not be used as a variable name
- Variable names cannot include spaces
- The first character must be a-z, A-Z or an underscore character(_)
- After the first character you may use a-z, A-Z, 0-9, or underscores
- Choose variable names that reflect a variable's purpose



Variable Style

- The variable name begins with lowercase letters
- The first character of the second and subsequent words is written in uppercase (camelCase convention)
- variables containing multiple words can be separated with underscores (_) to represent spaces.



Variable Assignment

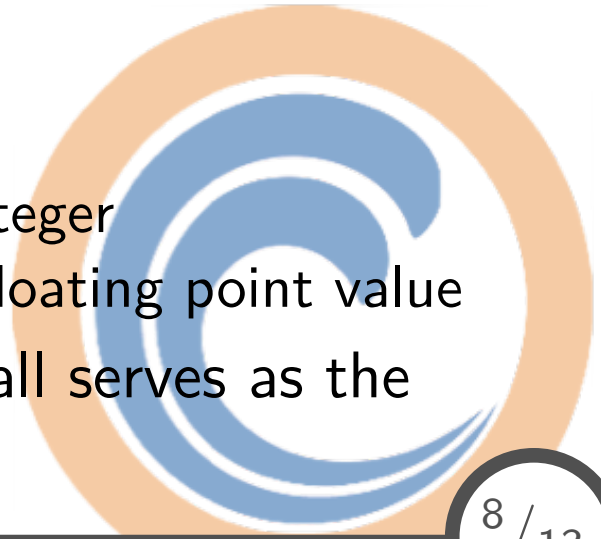
- A variable can be assigned a value using the assignment operator
- A variable can be reassigned a value using the assignment operator and the previously stored value is removed from memory through a process called garbage collection

Basic `data types` in Python

- The interpreter reads a numeric literal in the program code and determines it's data type as
  - `int`: a numeric literal with no decimal point (i.e. `7, -9, 125`)
  - `float`: a numeric literal written with a decimal point (i.e `1.5, 3.141519, 5.0`)
- `type` function allows you to determine the data type of a value and is useful in interactive mode.
- Currency symbols, special characters, or commas cannot be written as a numeric literal!
- `str` is the datatype used to store strings
- You may reassign a variable to a different type

Input

- `variable = input('prompt')` reads input from the keyboard after the user pressed enter.
- When the enter key is pressed the data that was typed it returned as a string and assigned to the variable.
- The input argument of the function is a string that is used to instruct the user to enter a value that will be assigned to variable.
- Since the `input` function always returns the user's input as a string, numeric data entered by the user is assigned a string type.
- To convert the input datatype you may use:
  - `int(item)` where item is converted to an integer
  - `float(item)` where item is converted to a floating point value
- Nested function calls is when one function call serves as the argument of another.



Input Continued...

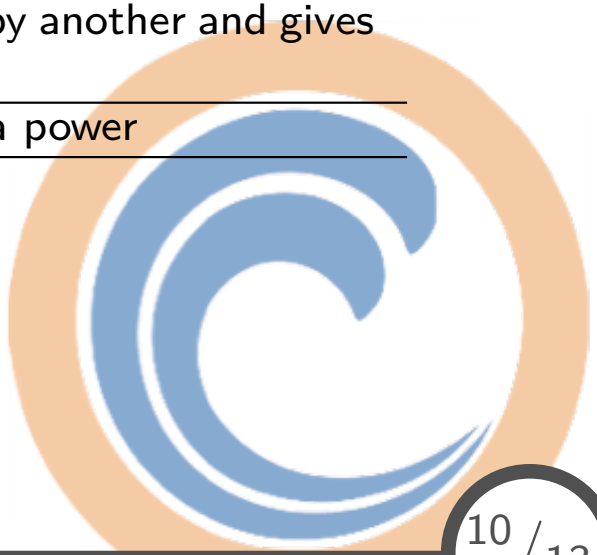
`int()` and `float()` functions give an exception or ValueError for unexpected datatypes.

```
age = int(input('How old are you?'))
```

Math Operators

Follows the mathematical PEMDAS order of operations!

Symbol	Operation	Description
+	Addition	Adds two numbers
-	Subtraction	Subtracts one number from another
*	Multiplication	Multiplies one number by another
/	Division	Divides one number by another and gives the result as a floating-point number
//	Integer Division	Divides one number by another and gives the result as an integer number
%	Remainder	Divides one number by another and gives the remainder
**	Exponent	Raises a number to a power



/ vs //

- 5/2 gives 2.5
- 5//2 gives 2
- -5//2 gives -3
  - When the result is positive, it is truncated, which means that its fractional part is thrown away.
  - When the result is negative, it is rounded away from zero to the nearest integer.

Breaking long statements into multiple lines

```
result = var1 * 2 + var2 * 3 + \
                var3 * 4 + var4 * 5
print('We sold', units_sold, \
          'for a total of', sales_amount)
```

Specifying Item Separator in print()

```
#separate with new line
print('One')
print('Two')
print('Three')

#separate with NO space
print('One', end=' ')
print('Two', end=' ')
print('Three')

#separate with ONE space
print('One', 'Two', 'Three')

#separate with *
print('One', 'Two', 'Three', sep='*')
```