Variables, Expressions, and Statements

CSC 1200 - Principles of Computing

Overview

- Values and Types
- Variables
- Operators and Operands
- Expressions and Statements
  - Order of Operations
- Comments

Values and Types

- Data can have different **types**.
  - Integers are type int
  - Numbers with a decimal point are called floating-point numbers, and are type float
  - A character or sequence of characters is called a string, and has type str
  - Many more
- Each type can represent different **values**.
  - An integer can have values like 2, 0, -5, …
  - A floating-point number can have values like 2.0, 0.0, -5.25, …
  - A string can have values like “Hi”, “2”, “program”, “let’s get started”, …
- The values for each type have a specific **syntax**.
  - Integers may have a negative sign followed by a sequence of digits. They CANNOT have commas.
  - Floating-point numbers CANNOT have commas. You must use a decimal point if you want the type to be float.
  - Strings can be enclosed in single OR double quotes.

Variables

- A **variable** is a name that can refer to a value. The variable will have the same type as the value to which it refers.
- An **assignment statement** is used to associate a value with a variable.
  - FORMAT: variable name = value
  - n = 5 assigns the integer value 5 to the variable n
  - e = 2.81828 assigns the floating-point value 2.81828 to the variable e
  - message = 'What is your quest?' assigns the string 'What is your quest?' to the variable message.
  - NOTE: the variable name MUST come first. 5 = n is a syntax error.

Variable Names and Keywords

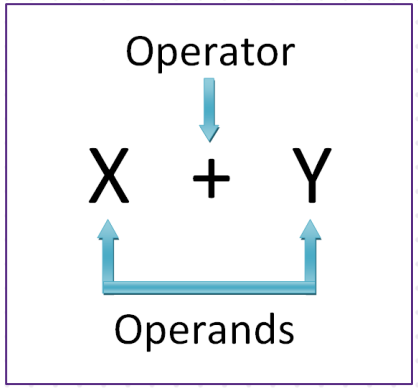
False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

- Variable **names** also have a specific syntax.
  - Must begin with a letter
  - Can contain both letters and numbers
  - Can use the '_' character (which is very common)
  - Can be arbitrarily long
  - CANNOT be a Python keyword
  - Variable names are case sensitive (Name is NOT the same as name or NamE or NAME)

- Keywords are strings that are part of the Python language and have special significance to the Python interpreter.
  - https://realpython.com/python-keywords/

Operators and Operands

- **Operators** are special symbols that represent computation.
- **Operands** are the values on which the operator is applied.



Addition: +

Subtraction: -

Multiplication: *

Floating-Point Division: /

Exponentiation: **

- Note: ^ is a bitwise XOR (exclusive OR)

Integer (or Floor) Division: //

Modulo: %

Expressions and Statements

- An **expression** is a combination of values, variables, and operators
  - An expression has a value
  - 17, 17+2, x, x + 2 -- provided x has been defined
- A **statement** is a unit of code that the Python interpreter can execute.
  - Examples that we've seen: print statements, assignment statements
  - A statement DOES NOT have a value
-
- Interactive mode and script mode
  - You can type expressions or statements in interactive mode
  - Each line of a python program used in script mode will contain a statement.
  - **Example**: expression example

Order of Operations

- Python follows the mathematical convention for rules of precedence of operations: PEMDAS
  - **P**arentheses have the highest precedence
  - **E**xponentiation has the next higher precedence
  - **M**ultiplication and **D**ivision have the same precedence and are performed left to right
  - **A**ddition and **S**ubtraction have the same precedence and are performed left to right

Notes:

- Python does NOT have the same syntax as your calculator
- You CANNOT use (2)(3) for multiplication
- You CANNOT use 2x for multiplication

String Operations

- In general, you can't perform mathematical calculations on strings, even if the numbers look like strings.
  - Can't do '2' - '1'
  - Can't do 'eggs' / 'easy'
  - Can't do 'third' * 'a charm'

There are some special string operators.

- String concatenation: +

- String repetition: *

Comments

- Comments are notes that are added to the program to explain what the program is doing.
  - A comment starts with # and ends at the end of the line
  - Comments should be used to document non-obvious features of the code
  - Comments can (and should) be used to provide visual separation between logical units of the program

```
1  #This comment will be ignored by the interpreter
2  print ("Comment Example")
3
```