COMP 204

Variables

Mathieu Blanchette,
based on material from Yue Li, Carlos Oliver and Christopher Cameron

Quiz 3 password: on the blackboard

Reminder: Data types

In Python, data comes in different native types:

- Strings (called str): sequence of zero or more characters.
- Integers (called int): Any positive or negative integer: 17, 0, -53, 64729237463928
- Decimal numbers (called float): Any decimal number: 3.1416, -2.43, 0.0
- Boolean (called bool): True or False
- and many more we will encounter later

To know the type of an object, use the type function:

```
type("Yue") # returns <class 'str'>
type(29.34) # returns <class 'float'>
```

In Python, data types are automatically handled by the interpreter. However, in other languages such as Java or C, we will need to declare the specific type of variable before we use it.

Operations on whole and fractional numbers

Python supports all basic arithmetic operations, which can be done on either whole numbers (int) or fractional numbers (float).

Operations	Example	Value	Type
Addition	7+12	19	int
Subtraction	3.14 - 2.78	0.3600000000000003	float
Multiplication	2 * 3.1416	6.2832	float
Division	33 / 8	3.3	float
	33 / 11	3.0	float
Modulus (only on int)	27 % 10	7	int
Exponentiation	4**3	$4^3 = 64$	int
Combination	2 + 6*2 - 8**2 / 4	-2.0	float
	(2+6)*(2 - 8**2/4)	-112.0	float

Precedence of arithmetic operators:

Exponentiation > multiplication/division > addition/subtraction

Use parentheses to group terms as desired

Basic operations on strings

String Operations	Example	Value	Type
Concatenation	'Hello'+'World'	'HelloWorld'	str

...

and many more later!

So Python is just a fancy calculator?

- No! Programming is about linking multiple operations together
- For this, it is useful to be able to save to memory the results of an operation
- To this end, we use **variables**

Variables

Variables allow a program to remember values throughout the execution of the program.

This is how a program uses the computer's memory.

A variable has a *name* and a *value*.

A program can

- Create new variables
- Set the value of variables
- Look up the value of variables to include them in expressions
- Change the value of variables (hence the name)

Variables assignment

We can think of a variable as a box:

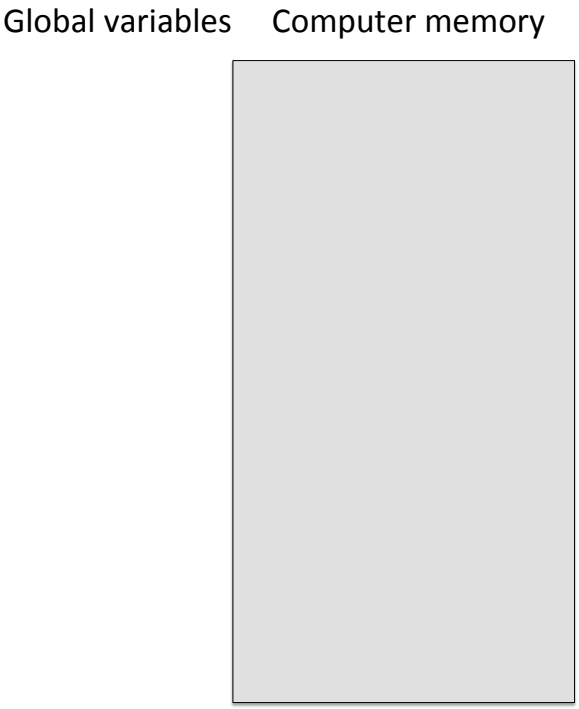
- the *name* of variable is the name of the box
- the *value* of the variable is the content of the box

A variable *assignment* assigns a certain value to the variable:
Syntax: variable_name = some_value
Meaning: the object some_value is stored in the variable named variable_name. Important:

- The value of a variable can be changed by assigning a new value to it. The old value is lost.
- In an assignment, the *right hand side is evaluated first*, and the result is stored in the variable.

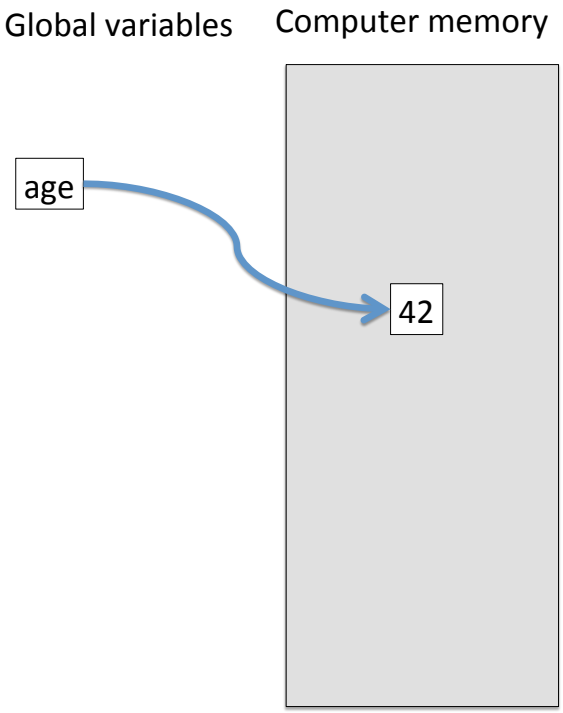
Example

```
age = 42
# puts 42 in the box called age.
# type(age) is int

weight = 76.6
# puts 76.6 in the box called weight.
# type(weight) is float

name = "Mathieu"
# puts "Mathieu" in the box called name.
# type(name) is str

age = 43
# changes value of age to 43.
# Previous value is overwritten

44 = age # Illegal: variable's name must always
# be on the left side of the = sign.
```



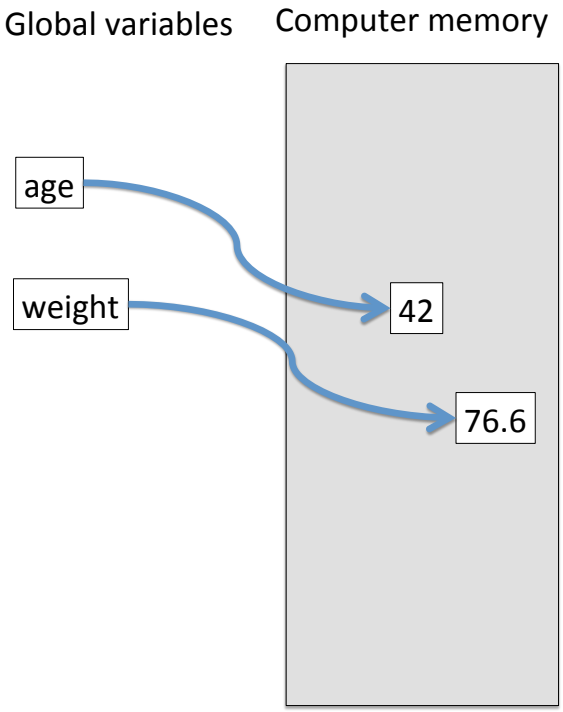
Example

```
age = 42
# puts 42 in the box called age.
# type(age) is int

weight = 76.6
# puts 76.6 in the box called weight.
# type(weight) is float

name = "Mathieu"
# puts "Mathieu" in the box called name.
# type(name) is str

age = 43
# changes value of age to 43.
# Previous value is overwritten

44 = age # Illegal: variable's name must always
# be on the left side of the = sign.
```



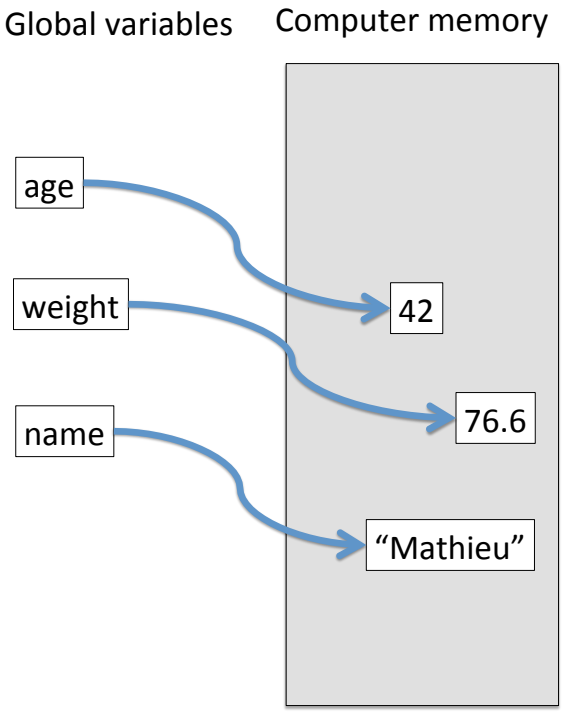
Example

```
age = 42
# puts 42 in the box called age.
# type(age) is int

weight = 76.6
# puts 76.6 in the box called weight.
# type(weight) is float

name = "Mathieu"
# puts "Mathieu" in the box called name.
# type(name) is str

age = 43
# changes value of age to 43.
# Previous value is overwritten

44 = age # Illegal: variable's name must always
# be on the left side of the = sign.
```



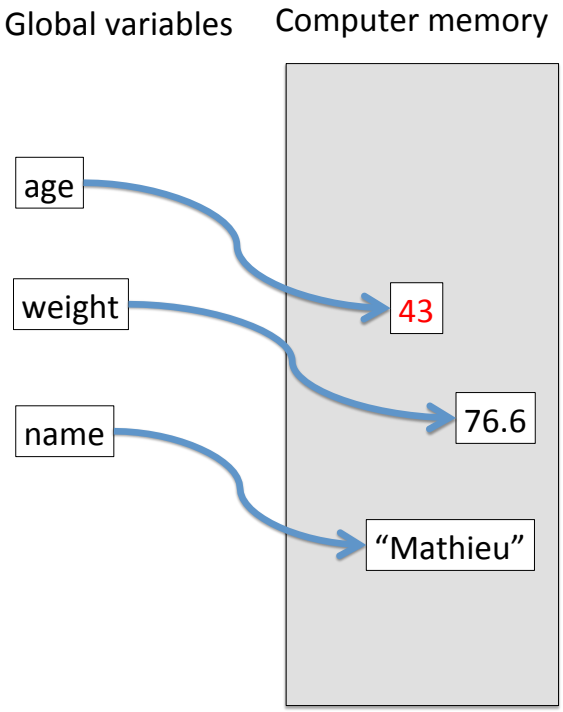
Example

```
age = 42
# puts 42 in the box called age.
# type(age) is int

weight = 76.6
# puts 76.6 in the box called weight.
# type(weight) is float

name = "Mathieu"
# puts "Mathieu" in the box called name.
# type(name) is str

age = 43
# changes value of age to 43.
# Previous value is overwritten

44 = age # Illegal: variable's name must always
# be on the left side of the = sign.
```



Example

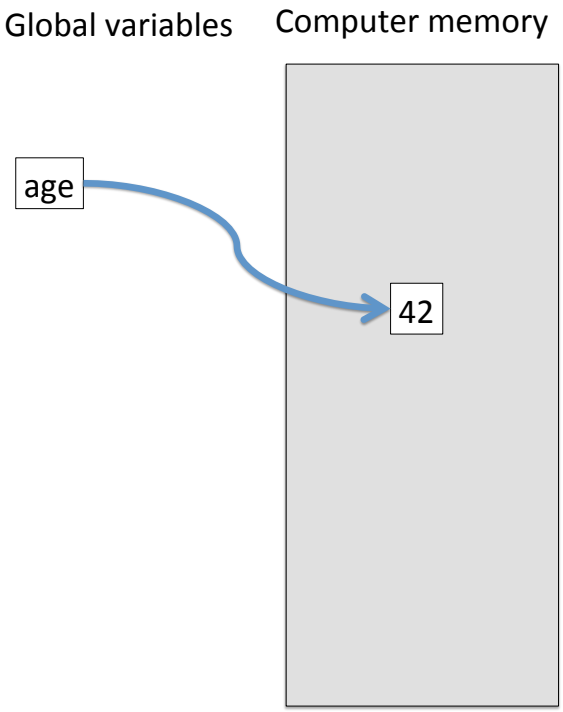
```
age = 42
# puts 42 in the box called age.
# type(age) is int

weight = 76.6
# puts 76.6 in the box called weight.
# type(weight) is float

name = "Mathieu"
# puts "Mathieu" in the box called name.
# type(name) is str

age = 43
# changes value of age to 43.
# Previous value is overwritten

44 = age # Illegal: variable's name must always
# be on the left side of the = sign.
```



Accessing variables

We can access the value stored in a variable by just writing the variable's name.
Example:

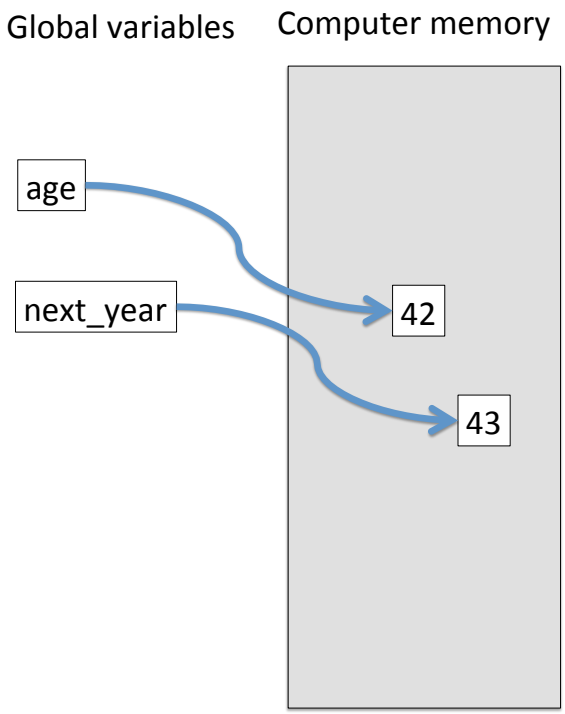
```
age = 42
print(age) # prints 42
next_year = age + 1  # starts by evaluating age+1, which requires looking up the value of the age variable (which is 42). Then calculates 42+1, and stores the result (43) in next_year.
age = 55 # age is now 55, but next_year is still 43
```



Accessing variables

We can access the value stored in a variable by just writing the variable's name.
Example:

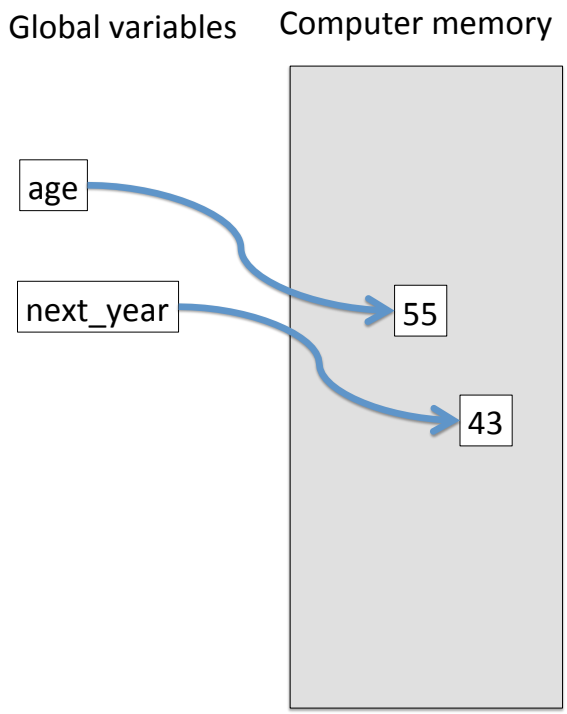
```
age = 42
print(age) # prints 42
next_year = age + 1  # starts by evaluating age+1, which requires looking up the value of the age variable (which is 42). Then calculates 42+1, and stores the result (43) in next_year.
age = 55 # age is now 55, but next_year is still 43
```



Accessing variables

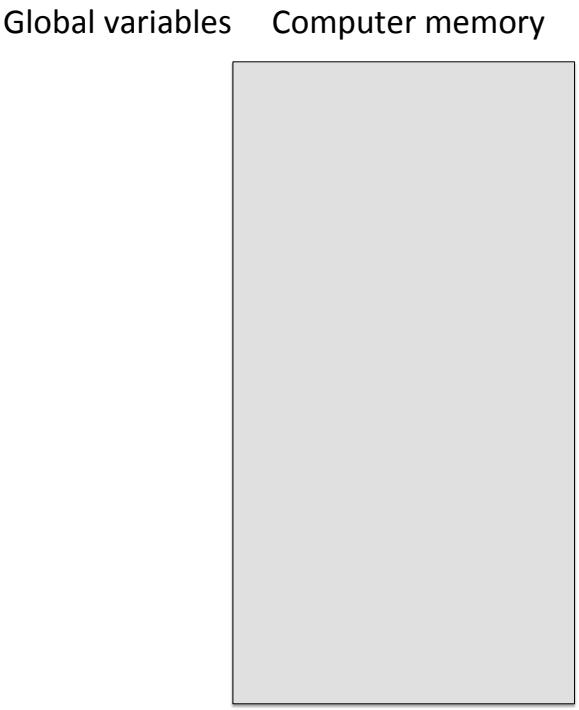
We can access the value stored in a variable by just writing the variable's name.
Example:

```
age = 42
print(age) # prints 42
next_year = age + 1  # starts by evaluating
age+1, which requires looking up the value of the
age variable (which is 42). Then calculates 42+1,
and stores the result (43) in next_year.
age = 55 # age is now 55, but next_year is still 43
```



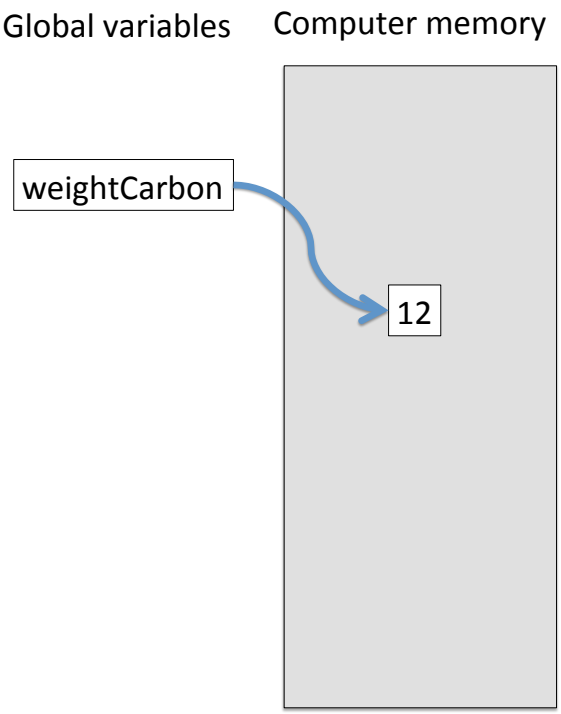
Example of Variable: calculate the molecular mass of CO_2

```
weightCarbon = 12
# This creates a variable weightCarbon,
# assigns it value 12

weightOxygen = 16
# This creates a variable weightOxygen,
# assigns it value 16

print('The weight of carbon is:', weightCarbon)
# This looks up the value of variable weightCarbon,
# performs the print statement

print('The weight of oxygen is:', weightOxygen)

weightCO2 = weightCarbon + 2 * weightOxygen
# This first evaluates the right-hand side,
# based on the current values of weightCarbon
# and weightOxygen. This yields 44.
# It then creates the variable weightCO2
# and assign it the value 44.
# Nothing gets printed so far

print('The weight of CO2 is:', weightCO2)
```



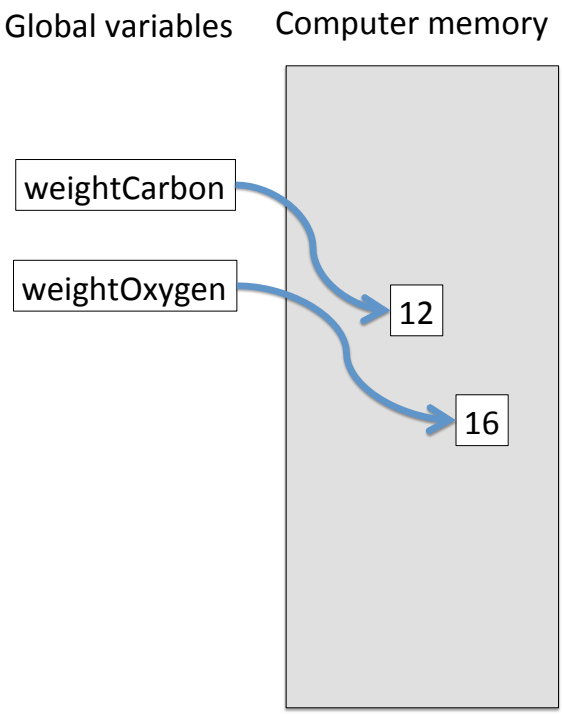
Example of Variable: calculate the molecular mass of CO_2

```
weightCarbon = 12
# This creates a variable weightCarbon,
# assigns it value 12

weightOxygen = 16
# This creates a variable weightOxygen,
# assigns it value 16

print('The weight of carbon is:', weightCarbon)
# This looks up the value of variable weightCarbon,
# performs the print statement

print('The weight of oxygen is:', weightOxygen)

weightCO2 = weightCarbon + 2 * weightOxygen
# This first evaluates the right-hand side,
# based on the current values of weightCarbon
# and weightOxygen.  This yields 44.
# It then creates the variable weightCO2
# and assign it the value 44.
# Nothing gets printed so far

print('The weight of CO2 is:', weightCO2)
```



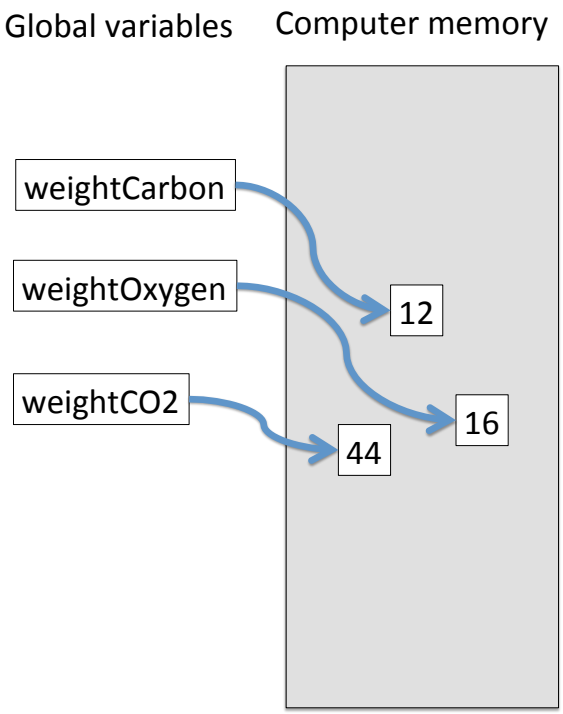
Example of Variable: calculate the molecular mass of CO_2

```
weightCarbon = 12
# This creates a variable weightCarbon,
# assigns it value 12

weightOxygen = 16
# This creates a variable weightOxygen,
# assigns it value 16

print('The weight of carbon is:', weightCarbon)
# This looks up the value of variable weightCarbon,
# performs the print statement

print('The weight of oxygen is:', weightOxygen)

weightCO2 = weightCarbon + 2 * weightOxygen
# This first evaluates the right-hand side,
# based on the current values of weightCarbon
# and weightOxygen. This yields 44.
# It then creates the variable weightCO2
# and assign it the value 44.
# Nothing gets printed so far

print('The weight of CO2 is:', weightCO2)
```



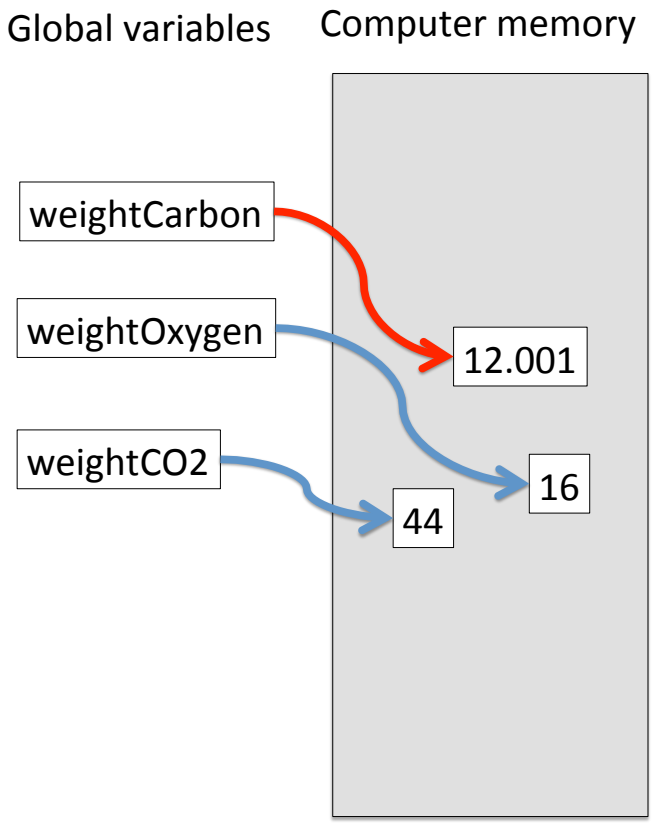
Example of Variable: calculate the molecular mass of CO_2

```
weightCarbon = 12
# This creates a variable weightCarbon,
# assigns it value 12

weightOxygen = 16
# This creates a variable weightOxygen,
# assigns it value 16

print('The weight of carbon is:', weightCarbon)
# This looks up the value of variable weightCarbon,
# performs the print statement

print('The weight of oxygen is:', weightOxygen)

weightCO2 = weightCarbon + 2 * weightOxygen
# This first evaluates the right-hand side,
# based on the current values of weightCarbon
# and weightOxygen.  This yields 44.
# It then creates the variable weightCO2
# and assign it the value 44.
# Nothing gets printed so far

print('The weight of CO2 is:', weightCO2)
```



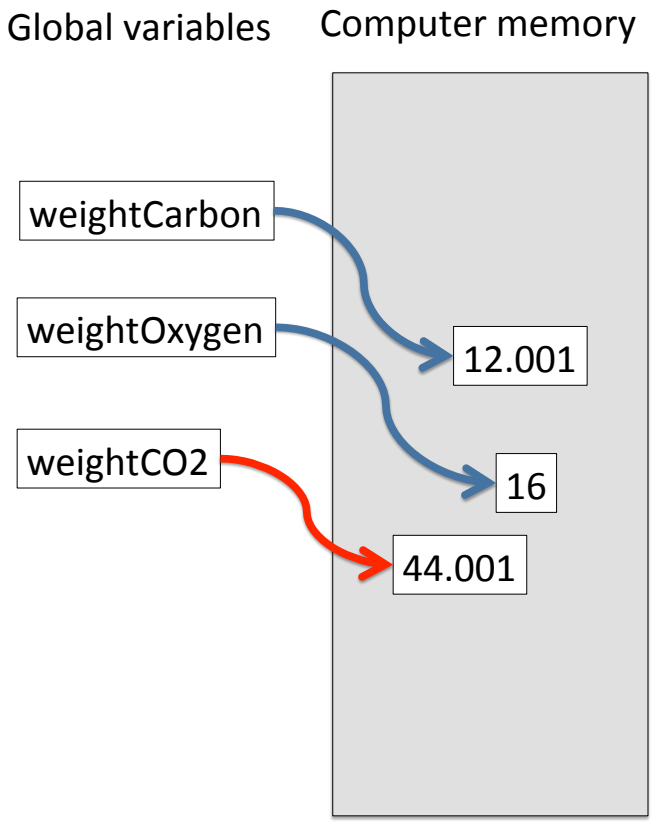
Variables - example

```
weightCarbon = 12
weightOxygen = 16
print('The weight of carbon is:', weightCarbon)
print('The weight of oxygen is:', weightOxygen)
weightCO2 = weightCarbon + 2 * weightOxygen
print('The weight of CO2 is:', weightCO2)

# Improved measurement of atomic masses
weightCarbon = 12.001
print('The weight of CO2 is:', weightCO2)
# weightCO2 remains 44
```



Variables - example

```
weightCarbon = 12
weightOxygen = 16
print('The weight of carbon is:', weightCarbon)
print('The weight of oxygen is:', weightOxygen)
weightCO2 = weightCarbon + 2 * weightOxygen
print('The weight of CO2 is:', weightCO2)

# Improved measurement of atomic masses
weightCarbon = 12.001
print('The weight of CO2 is:', weightCO2)
# weightCO2 remains 44

weightCO2 = weightCarbon + 2 * weightOxygen
# now weightCO2 becomes 44.001
print('The weight of CO2 is:', weightCO2)
```



Changing the value of a variable (weighCarbon) does not affect the value of other variables (weightCO2) unless we explicitly recompute that variable.

Live Demo in Spyder

Variables - example 2

Goal: Write a program that computes the body mass index (BMI) of a person: $BMI = weight/(height^2)$

```
weight = 69
height = 1.8
BMI = weight/(height**2)
print('A person with weight', weight, 'and height',
height, 'has BMI =', BMI)

weight = 74 # suppose the weight changes
# The value of BMI still has not changed
print('A person with weight', weight, 'and height',
height, 'has BMI =', BMI)

# We need to recalculate BMI to get the correct BMI
BMI = weight/(height**2)
print('A person with weight', weight,
'and height', height, 'has BMI =', BMI)
```

Live Demo in Spyder

Variables - example 3 (user input)

Goal: Write a program that asks the user for their weight and height and then computes BMI.

How? Use the input(String) function, which prompts the user to enter data, and returns the string that was typed.

```
weight = input('Please enter your weight (in kg): ')
height = input('Please enter your height (in m): ')
BMI = weight/(height**2)
print('Your BMI is', BMI)
```

Problem: We get a *runtime error*:

TypeError: unsupported operand type(s) for ** or pow(): 'str' and 'int'

Use the Python shell to find out what the type of the weight and height variables are.

```
type(weight) # Aha, it's a String, not an integer
type(height) # and this one too!
```

That's because the *input* function always produces a string, irrespective of what is actually typed by the user.

Converting between types

Python allows data to be converted from one type to another using type conversion functions:

```
int(someObject) # convert someObject to an integer
float(someObject) # convert someObject to a float
str(someObject) # convert someObject to a string
```

Example,

```
name='Yue' # name is a String
weight='66' # weight is a String
height='1.8' # height is a String
weightInt = int(weight) # weightInt is an integer 68
heightFloat = float(height) # heightInt is a float 1.8
heightInt = int(height) # heightInt is an integer 1
#Note: int() truncates decimal values
nameInt = int(name) # this causes an error, because
# the content of name cannot be converted to number
```

BMI program corrected

We use the type conversion functions to convert the output of the input function to float.

```
weight = input('Please enter your weight (in kg): ')
weightFloat= float(weight)
height = input('Please enter your height (in m): ')
heightFloat= float(height)
BMI = weightFloat/(heightFloat**2)
print('Your BMI is ' ,BMI)
```

Or more succinctly, we directly convert the output of the input function to a float, without saving the String in a variable:

```
weight=float(input('Please enter your weight (in kg): ')
height=float(input('Please enter your height (in m): '))
BMI = weight/(height**2)
print('Your BMI is ' ,BMI)
```

Live Demo in Spyder