1. Python is a calculator

2. A variable is a container

3. Different types cannot be compared

4. A program is a recipe

**Colvin Run Mill Corn Bread**

1 cup cornmeal
1 cup flour
1/2 teaspoon salt
4 teaspoons baking powder
3 tablespoons sugar
1 egg
1 cup milk
1/4 cup shortening (soft) or vegetable oil

Mix together the dry ingredients. Beat together the egg, milk and shortening/oil. Add the liquids to the dry ingredients. Mix quickly by hand. Pour into greased 8x8 or 9x9 baking pan. Bake at 425 degrees for 20-25 minutes.
1. Python is Like a Calculator
You Type Expressions. Python Computes Their Values.

• 5
• 3+4
• 44/2
• 2**3
• 3*4+5*6
  – If precedence is unclear, use parentheses
• (72 – 32) / 9 * 5
An Expression is Evaluated From the Inside Out

• How many expressions are in this Python code?

\[(72 - 32) / 9.0 * 5\]

\[(40) / 9.0 * 5\]

\[40 / 9.0 * 5\]

\[4.44 * 5\]

\[22.2\]
Another Evaluation Example

\[(72 - 32) / (9.0 \times 5)\]
\[(40) / (9.0 \times 5)\]
\[40 / (9.0 \times 5)\]
\[40 / (45.0)\]
\[40 / 45.0\]
\[.888\]
2. A Variable is a Container
Variables Hold Values

• Recall variables from algebra:
  – Let $x = 2$ ...
  – Let $y = x$ ...

• To assign a variable, use “varname = expression”

  $\pi = 3.14$
  $\pi$

  $\text{var} = 6 \times 10^{23}$

  $22 = x$  # Error!

• Not all variable names are permitted
Changing Existing Variables ("re-binding" or "re-assigning")

x = 2
x
y = 2
y
x = 5
x
y

• "=" in an assignment is not a promise of eternal equality
  – This is different than the mathematical meaning of "=

• Evaluating an expression gives a new (copy of a) number, rather than changing an existing one
How an Assignment is Executed

1. Evaluate the right-hand side to a value
2. Store that value in the variable

```
x = 2
print(x)
y = x
print(y)
z = x + 1
print(z)
x = 5
print(x)
print(y)
print(z)
```

State of the computer:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x: 2</td>
<td>y: 2</td>
</tr>
<tr>
<td>z: 3</td>
<td></td>
</tr>
</tbody>
</table>

Printed output:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

To visualize a program’s execution: [http://pythontutor.com](http://pythontutor.com)
More Expressions: Conditionals (value is True or False)

22 > 4    # condition, or conditional
22 < 4    # condition, or conditional
22 == 4   ...
x == 100  # Assignment, not conditional!
22 = 4    # Error!
x >= 5
x >= 100
x >= 200
not True
not (x >= 200)
3<4 and 5<6
4<3 or 5<6
temp = 72
water_is_liquid = (temp > 32 and temp < 212)

Numeric operators: +, *, **
Boolean operators: not, and, or
Mixed operators: <, >=, ==
More Expressions: strings

• A string represents text
  – 'Python'
  – myString = "BBM 101-Introduction to Programming"
  – ""

• Empty string is not the same as an unbound variable
  – "" and "" are the same

Operations:

• Length:
  – len(myString)

• Concatenation:
  – "Hacettepe" + " " + 'University'

• Containment/searching:
  – 'a' in myString
  – "a" in myString
3. Different Types cannot be Compared

```python
anInt = 2
aString = "Hacettepe"
anInt == aString  # Error
```
Types of Values

• Integers (int): \(-22, 0, 44\)
  – Arithmetic is exact
  – Some funny representations: 12345678901L

• Real numbers (float, for “floating point”): 2.718, 3.1415
  – Arithmetic is approximate, e.g., 6.022*10**23
  – Some funny representations: 6.022e+23

• Strings (str): "I love Python", ""

• Truth values (bool, for “Boolean”): True, False
Operations Behave differently on Different Types

3.0 + 4.0
3 + 4
3 + 4.0
"3" + "4"
# Concatenation
3 + "4"
# Error
3 + True
# Error

Moral: Python only *sometimes* tells you when you do something that does not make sense.
# Operations on Different Types

<table>
<thead>
<tr>
<th>Expression</th>
<th>Python 3.5</th>
<th>Python 2.x</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.0 / 4.0</td>
<td>3.75</td>
<td>3.75</td>
</tr>
<tr>
<td>15 / 4</td>
<td>3.75</td>
<td>3</td>
</tr>
<tr>
<td>15.0 / 4</td>
<td>3.75</td>
<td>3.75</td>
</tr>
<tr>
<td>15 / 4.0</td>
<td>3.75</td>
<td>3.75</td>
</tr>
<tr>
<td>15.0 // 4.0</td>
<td>3.0</td>
<td>3.0</td>
</tr>
<tr>
<td>15 // 4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>15.0 // 4</td>
<td>3.0</td>
<td>3.0</td>
</tr>
<tr>
<td>15 // 4.0</td>
<td>3.0</td>
<td>3.0</td>
</tr>
</tbody>
</table>

Before Python version 3.5, the operand used to determine the type of division.

/ : Division
//: Integer Division
Type Conversion

float(15)  15.0
int(15.0)  15
int(15.5)  15
int("15")  15
str(15.5)  15.5
float(15) / 4  3.75
A Program is a Recipe

Coffee
- Start
- Boil water
- Add coffee
- Add water to cup
- Milk \( \leftarrow \) Value

Milk = TRUE ?
- Y
  - Add milk
  - Drink coffee
  - Stop
- N

Milk?
Design the Algorithm Before Coding

• We should think (design the algorithm) before coding

• Algorithmic thinking is the logic. Also, called problem solving

• Coding is the syntax

• Make this a habit

• Some students do not follow this practice and they get challenged in all their courses and careers!
What is a Program?

• A program is a sequence of instructions

• The computer executes one after the other, as if they had been typed to the interpreter

• Saving your work as a program is better than re-typing from scratch

```
x = 1
y = 2
x + y
print(x + y)
print("The sum of", x, "and", y, "is", x+y)
```
The **print**() Statement

- The **print** statement always prints one line
  - The next print statement prints below that one

- Write 0 or more expressions after **print**, separated by commas
  - In the output, the values are separated by spaces

- Examples:
  ```python
  x = 1
  y = 2
  print(3.1415)
  print(2.718, 1.618)
  print()
  print(20 + 2, 7 * 3, 4 * 5)
  print("The sum of", x, "and", y, "is", x+y)
  ```

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1415</td>
<td>The sum of 1 and 2 is 3</td>
</tr>
<tr>
<td>2.718 1.618</td>
<td></td>
</tr>
<tr>
<td>22 21 20</td>
<td></td>
</tr>
</tbody>
</table>
  }

- The sum of 1 and 2 is 3
Exercise: Convert Temperatures

- Make a temperature conversion chart as the following
- Fahrenheit to Centigrade, for Fahrenheit values of: -40, 0, 32, 68, 98.6, 212
- \[ C = (F - 32) \times \frac{5}{9} \]

<table>
<thead>
<tr>
<th>Fahrenheit</th>
<th>Centigrade</th>
</tr>
</thead>
<tbody>
<tr>
<td>-40</td>
<td>-40.0</td>
</tr>
<tr>
<td>0</td>
<td>-17.7778</td>
</tr>
<tr>
<td>32</td>
<td>0.0</td>
</tr>
<tr>
<td>68</td>
<td>20.0</td>
</tr>
<tr>
<td>98.6</td>
<td>37.0</td>
</tr>
<tr>
<td>212</td>
<td>100.0</td>
</tr>
</tbody>
</table>

- You have created a Python program!
- (It doesn’t have to be this tedious, and it won’t be.)
Expressions, Statements, and Programs

• An expression evaluates to a value
  
  \[3 + 4\]
  \[\pi \times r^{**2}\]

• A statement causes an effect
  
  \[\pi = 3.14159\]
  \[\text{print}(\pi)\]

• Expressions appear within other expressions and within statements
  
  \[(\text{fahr} - 32) \times (5.0 / 9)\]
  \[\text{print}(\pi \times r^{**2})\]

• A statement may not appear within an expression
  
  \[3 + \text{print}(\pi)\]  # Error!

• A program is made up of statements
  
  – A program should do something or communicate information
print() Function

```python
print(3 * 2)
print(3 * "abc")
print(True)
print(False)
print(None)
```
1. Python is a calculator

2. A variable is a container

3. Different types cannot be compared

4. A program is a recipe
Programming Languages

• A programming language is a “language” to write programs in, such as Python, C, C++, Java

• The concept of programming languages are quite similar

• Python:

```python
print("Hello, World!")
```

• Java:

```java
public static void main(String[] args) {
    System.out.println("Hello, World");
}
```

• Python is simpler! That’s why we are learning it first 😊
Evolution of Programming Languages

Assembly 1949

Fortran 1954

Algol 1958

Sketchpad 1963

Smalltalk 1971

Lisp 1958

Basic 1964

Objective-C 1983

Perl 1987

Self 1987

Tcl 1988

Python 1991

Dylan 1991

Visual Basic 1991

Java 1991

C++ 1983

C 1969

PHP 1995

Javascript 1995

Ruby 1993

C# 2000
Most Popular Coding Languages of 2015

- C++ 9.8%
- Ruby 7.1%
- Javascript 6.5%
- Python 31.2%
- Java 19.6%
- C# 7.4%
- PHP 3.6%
- C 6.1%
- Go 2.3%
- Bash .4%
-Objective C 1%
- Haskell 1.5%
- Lua .3%
-Clojure .2%
- R .1%
-TCL .07%
-VB .NET .04%