



# An Introduction To The Python Programming Language

A presentation by the IEEE's FWCS Computer Society

Dr. Jim Anderson

# Goals For This Presentation



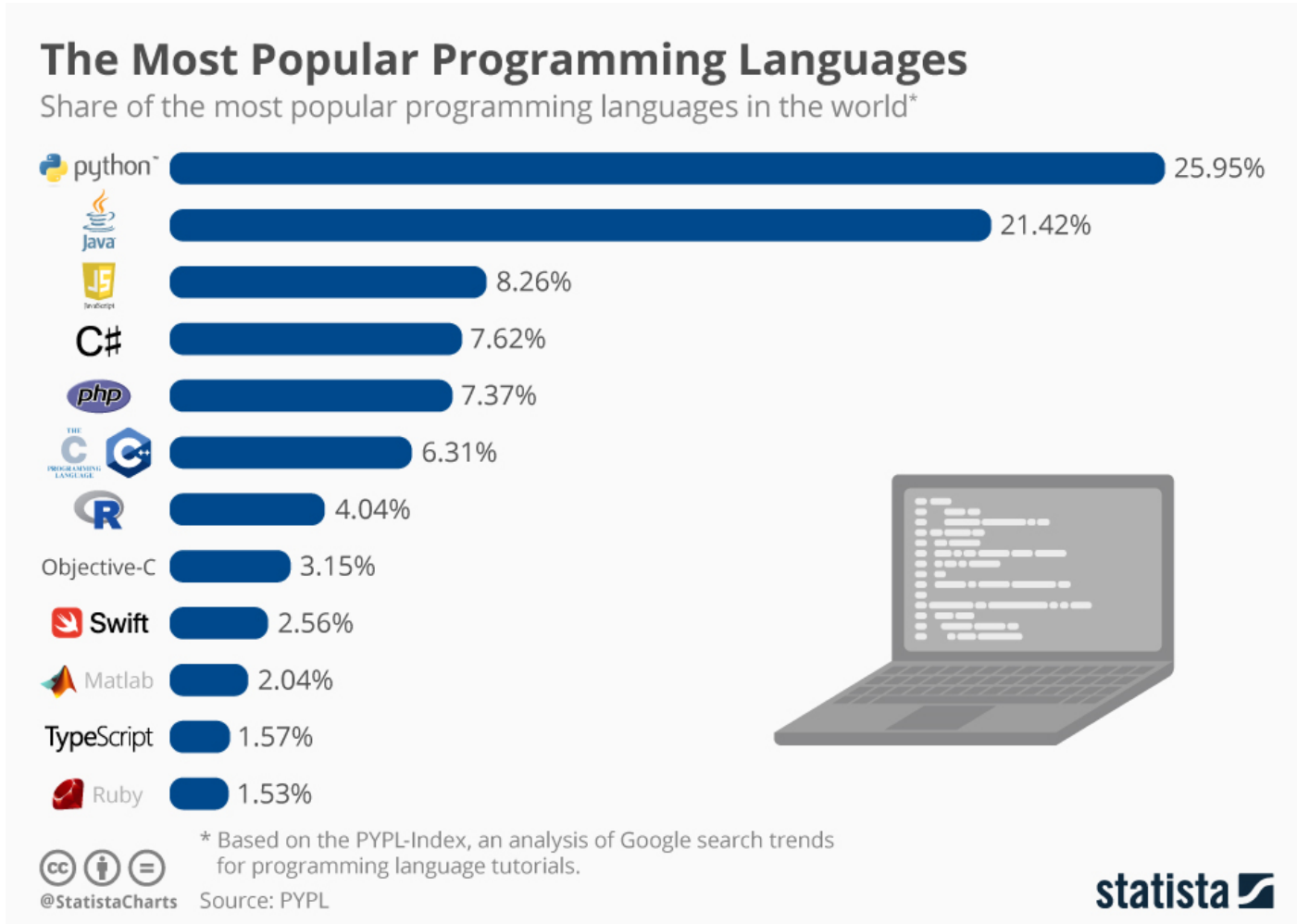
1. Introduce you to the Python programming language
2. Show you how you can install Python on your home computer
3. Explain how variables work in Python
4. Explore how to get input and how to produce output from a program
5. Discover how to control program flow
6. Introduce the concepts of a list and what can be done with it
7. Explore functions and how they can be used to organize code

# Programming Languages That Are Used To Create Software

*Computer programs, known as software, are instructions that tell a computer what to do*



# Top 30 Programming Languages



# What We'll Be Using In This Presentation



Your laptop



The Python computer language  
(free!)

# Who Created Python?

- The Python language was conceived in the late 1980s and its implementation was started in December 1989 by **Guido van Rossum** at CWI in the Netherlands
- Van Rossum needed to carry out repetitive tasks for administering computer systems. He was dissatisfied with other available languages that were optimized for writing large and fast programs.
- He needed to write smaller programs that didn't have to run at optimum speed. It was important to him that he could author the programs quickly and update them quickly as his needs changed.
- Therefore, he designed a language that made it very easy to work with complex data.





# What Is Python?

- Python is a widely used general-purpose, high-level programming language.
- Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in **fewer lines of code** than would be possible in languages such as C++ or Java.
- Python supports multiple programming paradigms, including object-oriented, imperative and functional programming.

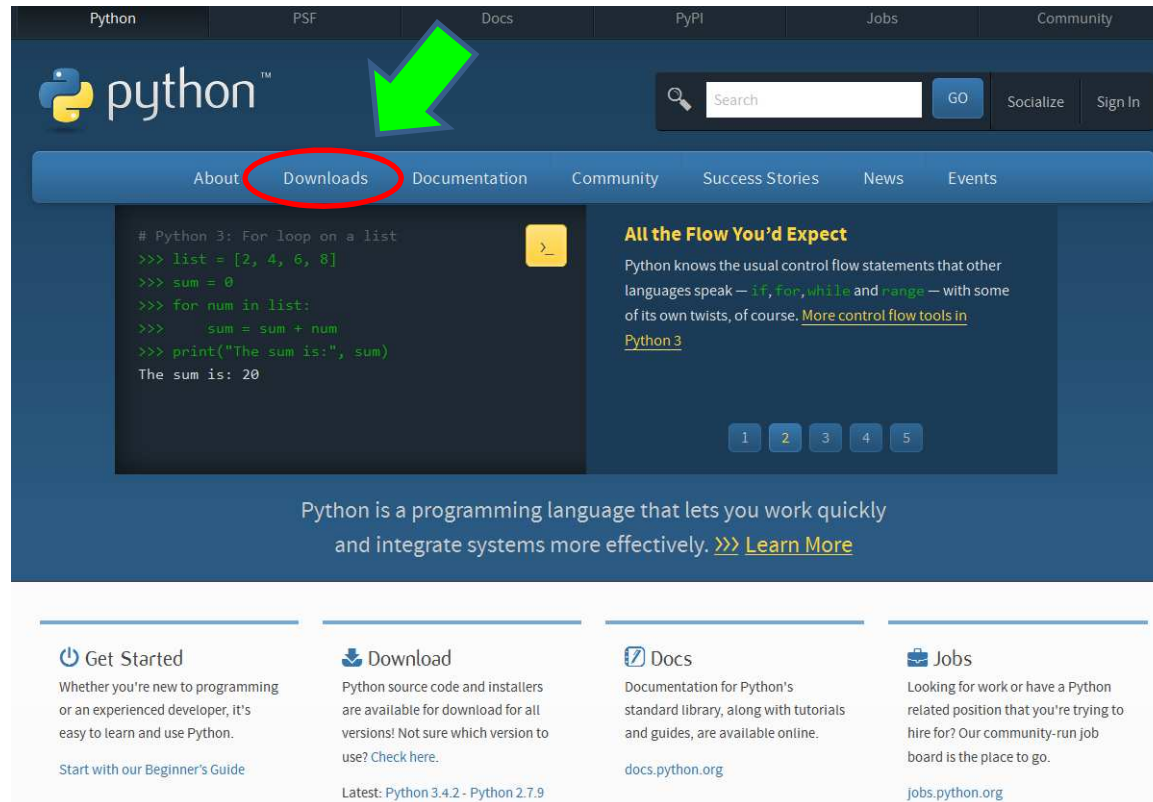
# Why Do People Like Python?

- Python has a much simpler and cleaner syntax than other popular languages such as Java, C, and C++, which makes it easier to learn.
- You can try out short Python programs in an interactive environment, which encourages experimentation and rapid turnaround.
- Python is also very portable between computer systems. The same Python program will run, without change, on Windows, UNIX, Linux, or Macintosh.





# Let's Get A Copy Of Python



The screenshot shows the Python.org website. At the top, there is a navigation bar with links for Python, PSF, Docs, PyPI, Jobs, and Community. Below this is a search bar and social media links. A main navigation bar contains links for About, Downloads, Documentation, Community, Success Stories, News, and Events. The 'Downloads' link is circled in red, and a green arrow points to it. The main content area features a code snippet on the left and an article titled 'All the Flow You'd Expect' on the right. At the bottom, there are four columns: 'Get Started', 'Download', 'Docs', and 'Jobs', each with a brief description and a link.

```
# Python 3: For loop on a list
>>> list = [2, 4, 6, 8]
>>> sum = 0
>>> for num in list:
>>>     sum = sum + num
>>> print("The sum is:", sum)
The sum is: 20
```

**All the Flow You'd Expect**  
Python knows the usual control flow statements that other languages speak — `if`, `for`, `while` and `range` — with some of its own twists, of course. [More control flow tools in Python 3](#)

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> [Learn More](#)

Get Started	Download	Docs	Jobs
Whether you're new to programming or an experienced developer, it's easy to learn and use Python. <a href="#">Start with our Beginner's Guide</a>	Python source code and installers are available for download for all versions! Not sure which version to use? Check here.  Latest: Python 3.4.2 - Python 2.7.9	Documentation for Python's standard library, along with tutorials and guides, are available online.  <a href="http://docs.python.org">docs.python.org</a>	Looking for work or have a Python related position that you're trying to hire for? Our community-run job board is the place to go.  <a href="http://jobs.python.org">jobs.python.org</a>

Go to the Python Software Foundation's website:  
<https://www.Python.org/>

# Let's Get A Copy Of Python 3.10.4

The screenshot shows the Python Software Foundation website. The 'Downloads' menu is open, and the 'Python 3.7.4' option is highlighted with a red circle. A green arrow points to the 'Downloads' menu item. The website content includes a search bar, navigation links (About, Downloads, Documentation, Community, Access Stories, News, Events), and a main section with 'Get Started', 'Download', 'Docs', and 'Jobs' links. The 'Download' section mentions 'Latest: Python 3.7.4'. The footer includes 'Latest News' and 'Upcoming Events' links.

```
# Python 3: Sia
>>> print("Hello, I'm Python")
Hello, I'm Python

# Input, assign
>>> name = input("What is your name? ")
>>> print("Hi, Python.")
Hi, Python.
```

Python is a programming language that lets you work quickly and integrate systems more effectively. [»»» Learn More](#)

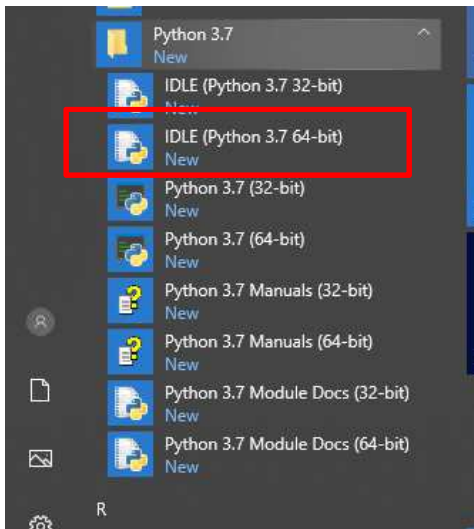
**Get Started**  
Whether you're new to programming or an experienced developer, it's easy to learn and use Python.  
Start with our [Beginner's Guide](#)

**Download**  
Python source code and installers are available for download for all versions!  
Latest: Python 3.7.4

**Docs**  
Documentation for Python's standard library, along with tutorials and guides, are available online.  
[docs.python.org](https://docs.python.org)

**Jobs**  
Looking for work or have a Python related position that you're trying to hire for? Our **retelaunched community-run job board** is the place to go.  
[jobs.python.org](https://jobs.python.org)

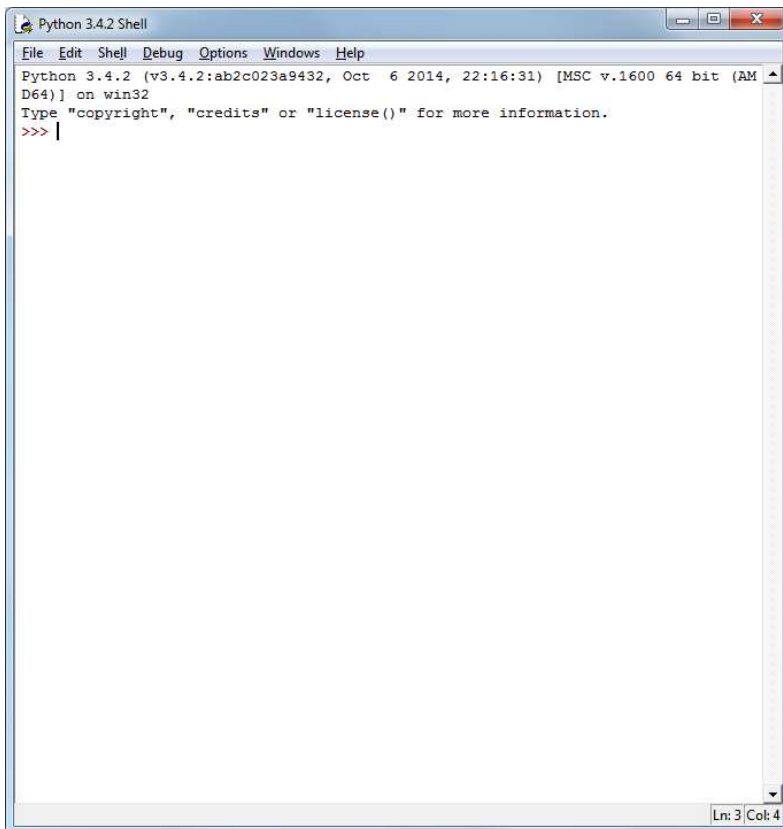
[»»» More](#) Latest News [»»» More](#) Upcoming Events [»»» More](#)



# Say Hello To IDLE



- IDLE (Integrated DeveLopment Environment) is an integrated development environment for Python.
- IDLE is intended to be a simple IDE and suitable for beginners, especially in an educational environment.
- Since van Rossum named the language Python partly to honor British comedy group Monty Python, the name IDLE was probably also chosen partly to honor Eric Idle, one of Monty Python's founding members.

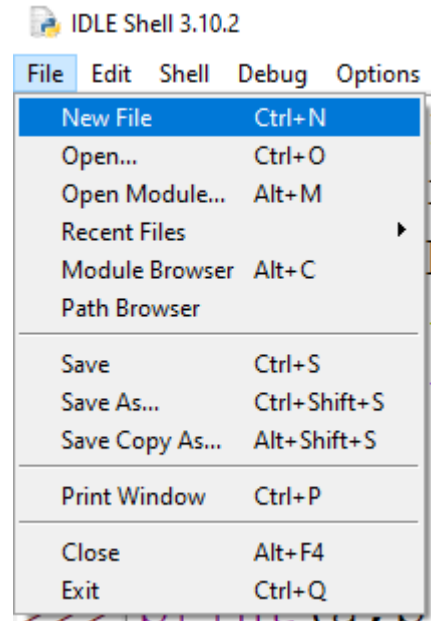


# Using IDLE

```
IDLE Shell 3.10.2
File Edit Shell Debug Options Window Help
Python 3.10.2 (tags/v3.10.2
AMD64)] on win32
Type "help", "copyright", "
>>> print("Hello World")
Hello World
>>> a = 2
>>> b = 3
>>> c = a + b
>>> print(a,b,c)
2 3 5
>>> |
```

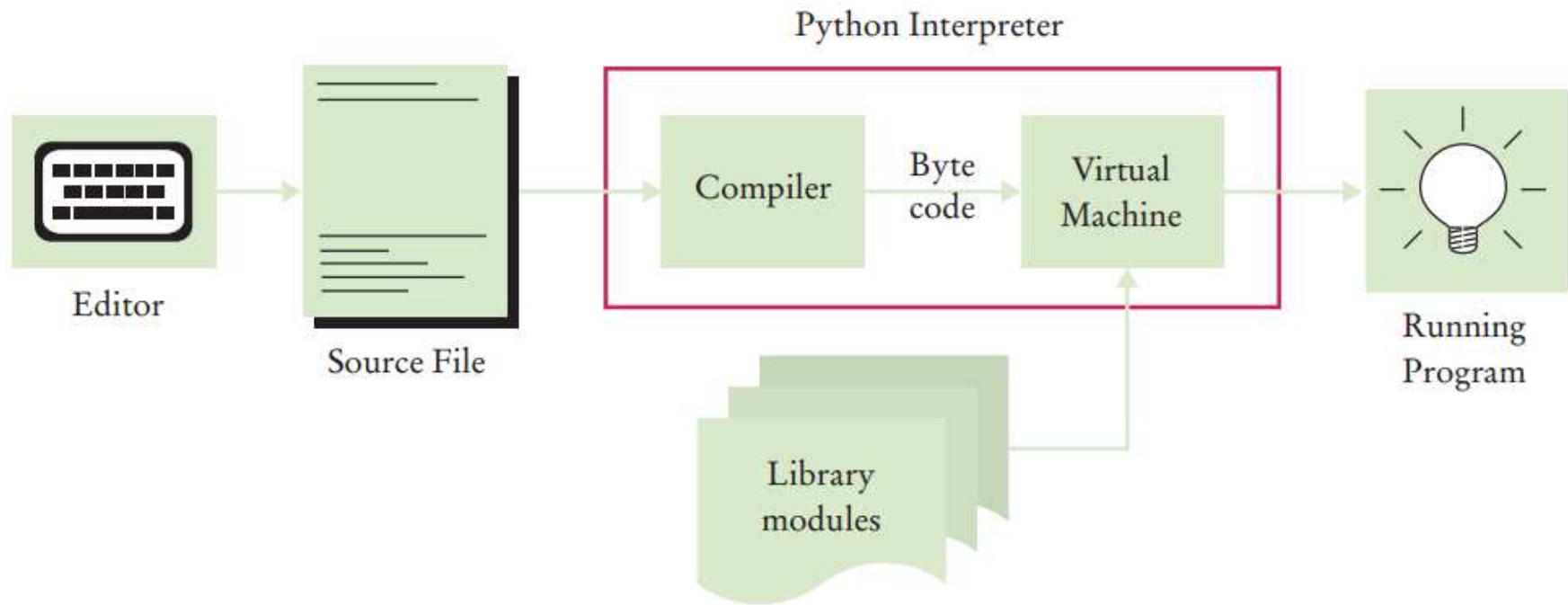
Python IDLE Interpreter

Python IDLE File



```
*untitled*
File Edit Format Run Options Window Help
print("Hello World")
a = 2
b = 3
c = a + b
print(a,b,c)
```

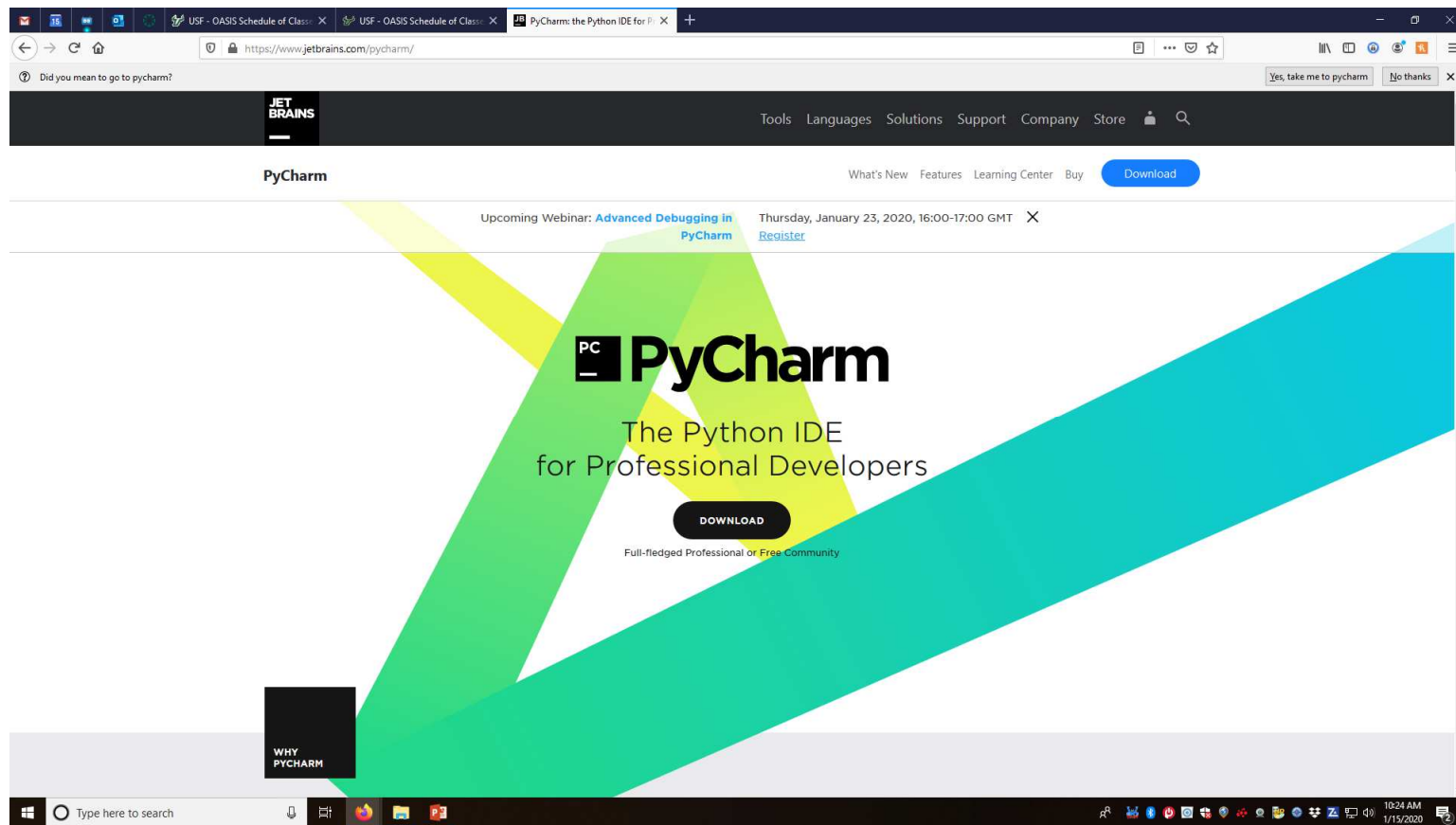
# How The Python Interpreter Works



- An interpreter is a program that translates one computer language to another.
- The Python interpreter translates your code to a byte code that gets executed by a virtual machine.
- Note that Python code never gets compiled – it never gets changed into the native language of your computer’s CPU.

# Getting Even More Software! (PyCharm)

<https://www.jetbrains.com/pycharm/>



The screenshot shows the PyCharm website homepage in a browser window. The browser's address bar displays the URL <https://www.jetbrains.com/pycharm/>. The website features a dark navigation bar with the JetBrains logo and links for Tools, Languages, Solutions, Support, Company, and Store. Below the navigation bar, the word "PyCharm" is prominently displayed, along with a "Download" button. A banner for an upcoming webinar titled "Advanced Debugging in PyCharm" is visible, scheduled for Thursday, January 23, 2020, from 16:00-17:00 GMT. The main content area features the PyCharm logo and the text "The Python IDE for Professional Developers", with a "DOWNLOAD" button and the note "Full-fledged Professional or Free Community". A "WHY PYCHARM" section is partially visible at the bottom left. The browser's taskbar at the bottom shows the Windows Start button, a search bar, and several application icons, including Firefox, File Explorer, and PyCharm. The system tray on the right indicates the time as 10:24 AM on 1/15/2020.

# You Want The “Community Edition”

The screenshot shows the JetBrains PyCharm download page. The browser address bar displays the URL: <https://www.jetbrains.com/pycharm/download/#section=windows>. The page features the PyCharm logo on the left, which includes the version information: Version: 2019.3.1, Build: 193.5662.61, and the release date: 18 December 2019. Below the logo are links for System requirements, Installation Instructions, and Other versions. The main content area is titled "Download PyCharm" and offers two options: "Professional" (Free trial) and "Community" (Free, open-source). The "Community" option is circled in red, and a green arrow points to its "Download" button. At the bottom of the page, there is a banner for the "Toolbox App" and a footer with social media icons and legal links.

# The Problem



I need to know what is going on in my program – it needs to be able to talk to me:

- How much is the loan going to cost?
- How much inventory do we have?
- How many chemicals were used to make the mixture?
- How many artifacts have been collected?



# The Solution: Python's Print Statement

*Syntax*    `print()`  
`print(value1, value2, ..., valuen)`

All arguments are optional. If no arguments are given, a blank line is printed.

```
print("The answer is", 6 + 7, "!")
```

The values to be printed, one after the other, separated by a blank space.

# Comments



- Comments are a critical part of every program
  - They are not executed (they don't do anything)
  - They serve to remind you what you were trying to do
  - When you come back 2 years later, they will be vital to understanding
  - In this class, comments are how you talk to the teacher / TAs
- Comments are anything that occurs after a “#”
  - Comments in Python start with the hash character, #, and extend to the end of the physical line.
  - A comment may appear at the start of a line or following whitespace or code, but not within a string literal.
  - A hash character within a string literal is just a hash character.
  - Example:

`# This is Homework #1`

`print("# This is not a comment")`

# Our First Program: Hello World!

- Python is case sensitive. You must enter upper- and lowercase letters exactly as they appear in a program listing.
- You must type `"print"`, you cannot type `"Print"` or `"PRINT"`.

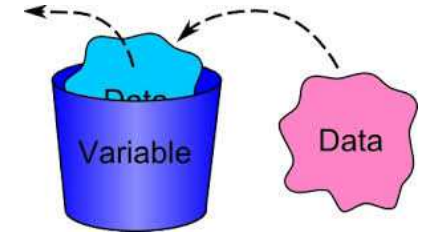


# Our First Python Program

```
# My first Python program  
print("Hello, World!")
```



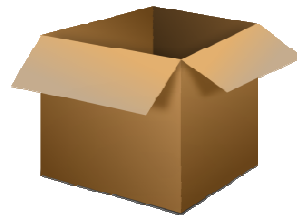
# Let's Talk About Variables...



- Variables are reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.
- Variables in Python consist of an alphanumeric name beginning in a letter. Variable names are case sensitive (scores not same as Scores or SCORES).
- Python variables do not have to be explicitly declared to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.
- The operand to the left of the = operator is the name of the variable and the operand to the right of the = operator is the value stored in the variable.

- Examples:

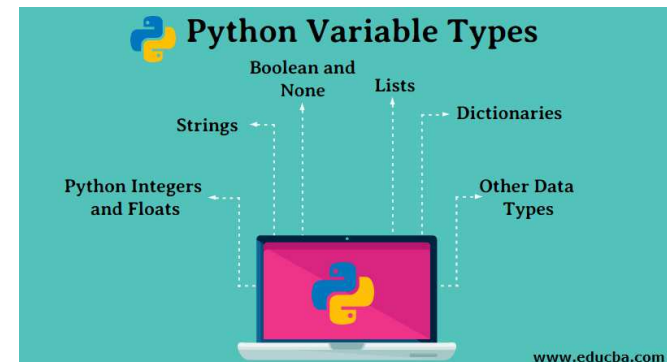
```
counter = 100  
miles = 1000.0  
name = "John"
```



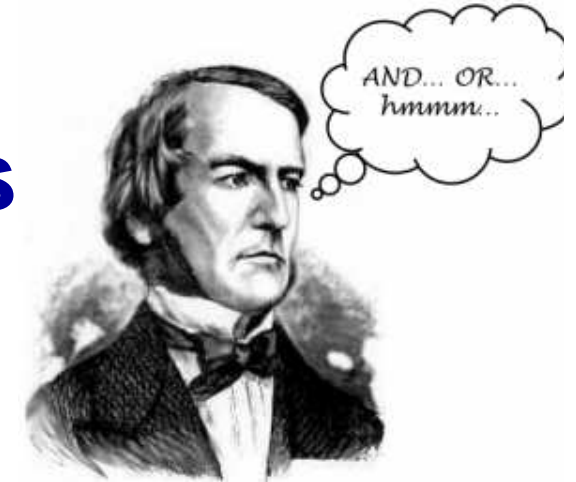
*Python variable names can have unlimited length – but keep them short!*

# Types Of Variables In Python

- **Integers**: Integers are a number that can be positive or negative or 0, but they cannot have a decimal point: 100, 5, 7, 32, 65
- **Floats**: Floats are decimal - they have a decimal point: 3.14, 7.3, 100.123
- **Strings**: a contiguous set of characters represented in the quotation marks. Python allows either pair of single or double quotes: "Hello World"
- **Boolean**: either **True** or **False**
- List
- Dictionary



# Boolean Variables



- Sometimes, you need to evaluate a logical condition in one part of a program and use it elsewhere.
- To store a condition that can be true or false, you use a *Boolean variable*.
- In Python, the **bool** data type has exactly two values, denoted **False** and **True**. Note that we capitalize the use of the False/True words.
- These values are not strings or integers; they are special values, just for Boolean variables.
- Example:
  - Here is the initialization of a variable set to True:  
**failed = True**
  - You can use the value later in your program to make a decision:  
**if failed: # Only executed if failed has been set to true**

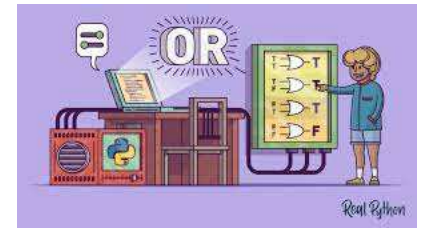
# Boolean Operators



- When you make complex decisions, you often need to combine Boolean values.
- An operator that combines Boolean conditions is called a **Boolean operator**.
- In Python, the *and* operator yields True only when both conditions are true.
- The *or* operator yields True if at least one of the conditions is true.



# Boolean Operators



Operator	Description	Example
and	Returns True if both statements are true	<code>x &lt; 5 and x &lt; 10</code>
or	Returns True if one of the statements is true	<code>x &lt; 5 or x &lt; 4</code>
not	Reverse the result, returns False if the result is true	<code>not(x &lt; 5 and x &lt; 10)</code>

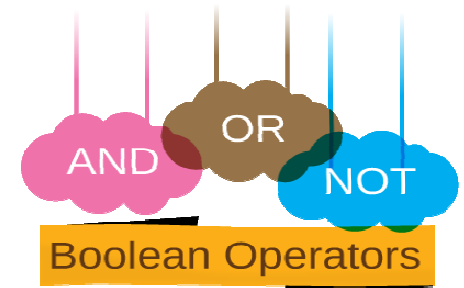
# Invert A Condition

- Sometimes you need to *invert* a condition with the **not** Boolean operator.
- The not operator takes a single condition and evaluates to **True** if that condition is false and to **False** if the condition is true.



```
if not frozen :  
    print("Not frozen")
```

# Python Logical Operators



Operator	Description	Example
and	Returns True if both statements are true	<code>x &lt; 5 and x &lt; 10</code>
or	Returns True if one of the statements is true	<code>x &lt; 5 or x &lt; 4</code>
not	Reverse the result, returns False if the result is true	<code>not(x &lt; 5 and x &lt; 10)</code>

```
if age > 21 and id = 1:  
    beerBought = 1  
else:  
    beerBought = 0
```

```
if hungry = 1 or bored = 1:  
    eatNow = 1  
else:  
    goRunning = 1
```

```
if not(age > 65):  
    socialSecurity = 0
```



# Say Hello To Strings!

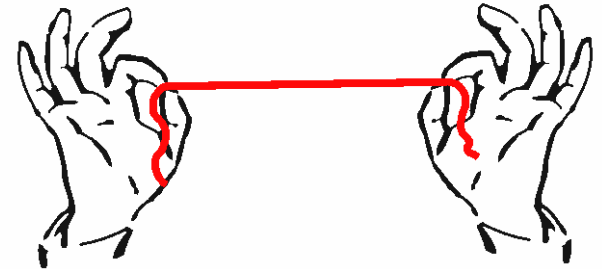
- Your computer programs will have to process text in addition to numbers.
- Text consists of **characters**: letters, numbers, punctuation, spaces, and so on.
- A **string** is a sequence of characters.
- Example: "John Smith"



# How Do We Deal With Strings?

- Strings can be stored in variables:

```
answer = "a"
```



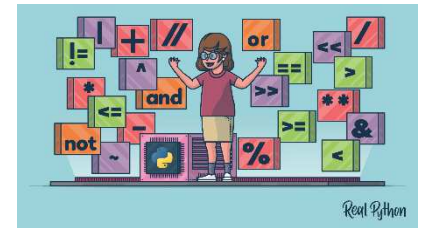
- A **string literal** denotes a particular string:

```
"Mississippi"
```

- In Python, string literals are specified by enclosing a sequence of characters within a matching pair of either single or double quotes.

```
"fire truck" or 'fire truck'
```

# Python Math Operators



Operator	Name	Example
+	Addition	$x + y$
-	Subtraction	$x - y$
*	Multiplication	$x * y$
/	Division	$x / y$
%	Modulus	$x \% y$
**	Exponentiation	$x ** y$
//	Floor division	$x // y$

$5 \% 2 = 1$

$5 // 2 = 2$

# Python Assignment Operators



Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3 <b>8</b>	x = x + 3
-=	x -= 3 <b>2</b>	x = x - 3
*=	x *= 3 <b>15</b>	x = x * 3
/=	x /= 3 <b>1.6667</b>	x = x / 3
%=	x %= 3 <b>2</b>	x = x % 3
//=	x //= 3 <b>1</b>	x = x // 3
**=	x **= 3 <b>125</b>	x = x ** 3
&=	x &= 3 <b>1</b>	x = x & 3
=	x  = 3 <b>7</b>	x = x   3
^=	x ^= 3 <b>6</b>	x = x ^ 3
>>=	x >>= 3 <b>0</b>	x = x >> 3
<<=	x <<= 3 <b>40</b>	x = x << 3

# Input and Output

- When a program asks for user input, it should first print a message that tells the user which input is expected. Such a message is called a **prompt**. In Python, displaying a prompt and reading the keyboard input is combined in one operation.
  - `first = input("Enter your first name: ")`
- The input function displays the string argument in the console window and places the cursor on the same line, immediately following the string.
  - Enter your first name: █





# Numerical Input



- The input function can only obtain a **string of text** from the user.
- To read an integer value, first use the input function to obtain the data as a string, then convert it to an integer using the int function.

```
numBottles = input("Please enter the number of bottles: ")  
bottles = int(numBottles)
```

```
bottlePrice = input("Enter price per bottle: ")  
price = float(bottlePrice)
```

# Sample Program



- Write a program that reads a number between 1,000 and 999,999 from the user, where the user enters a comma in the input. Then print the number without a comma.

- Here is a sample dialog; the user input is in color:

Please enter an integer between 1,000 and 99,999:

23,456

23456

# Formatted Output: New School

- The `format()` method was added in Python 2.6.
- Format method of strings requires more manual effort.
- User uses `{}` to mark where a variable will be substituted and can provide detailed formatting directives, but user also needs to provide the information to be formatted.
- This method lets us concatenate elements within an output through positional formatting.



# Formatted Output

- I would like write a program to tell the world that my name is John Smith, I own 27 televisions, and I am currently 4/5 of the way done with my college education.

- Items to be printed:

- John Smith
- 27
- 4/5



- I will store my name in a variable called "name", I will store the number of televisions that I own in a variable called "televisions".
- Basic: `print("I am"+name+", I own "+televisions+" televisions and I am "+4/5+"of the way through college")`
- Better: `print("I am {0}, I own {1} televisions and I am {2} of the way through college").format(name, televisions,4/5)`
- Flipped: `print(" I own {1} televisions, I am {2} of the way through college, I am {0}").format(name, televisions,4/5)`

# Decisions!

- One of the essential features of computer programs is their ability to make decisions
- Example: what a car does at a stoplight depends on the color of the light.
- A program can take different actions depending on inputs and other circumstances.

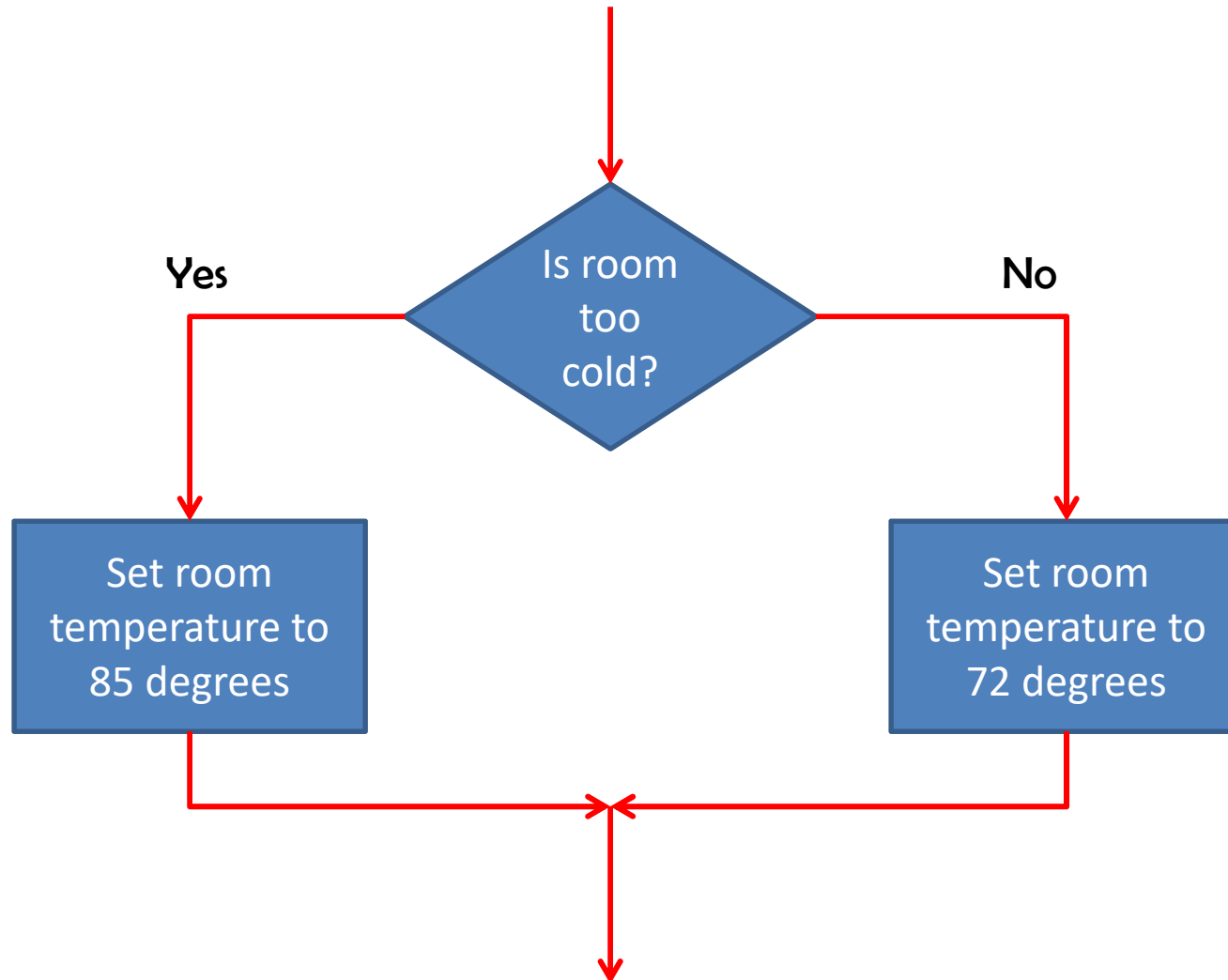


# Say Hello To The “IF” Statement

- In Python, an **IF** statement is used to implement a decision.
- When a condition is fulfilled, one set of statements is executed. Otherwise, another set of statements is executed



# Example Of Using IF



# Syntax Of Python IF Statement

**Syntax**    `if condition :`  
                  `statements`

`if condition :`  
                  `statements1`  
                  `else :`  
                  `statements2`

A condition that is true or false.  
Often uses relational operators:  
`== != < <= > >=`  
(See page 98.)

Omit the else branch  
if there is nothing to do.

The if and else  
clauses must  
be aligned.

The colon indicates  
a compound statement.

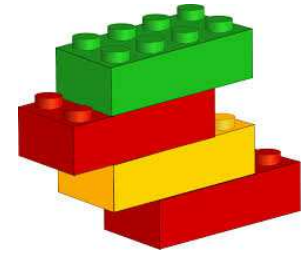
```
if floor > 13 :  
    actualFloor = floor - 1  
else :  
    actualFloor = floor
```

If the condition is true, the statement(s)  
in this branch are executed in sequence;  
if the condition is false, they are skipped.

If the condition is false, the statement(s)  
in this branch are executed in sequence;  
if the condition is true, they are skipped.



# Tabs



- Python requires blockstructured code as part of its syntax. The alignment of statements within a Python program specifies which statements are part of a given statement block.
- How do you move the cursor from the leftmost column to the appropriate indentation level? A perfectly reasonable strategy is to hit the space bar a sufficient number of times.
- With most editors, you can use the Tab key instead. A tab moves the cursor to the next indentation level. Some editors even have an option to fill in the tabs automatically

# Tabs



- Blockstructured code has the property that nested statements are indented by one or more levels:

```
if totalSales > 100.0 :
```

```
    discount = totalSales * 0.05
```

```
    totalSales = totalSales - discount
```

```
    print("You received a discount of ${0:%.2f}".format(discount))
```

```
else :
```

```
    diff = 100.0 - totalSales
```

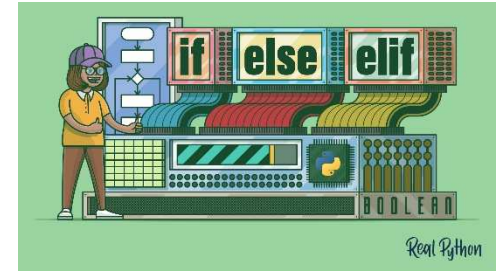
```
    if diff < 10.0 :
```

```
        print("purchase our item of the day & you can receive a 5% discount.")
```

```
    else :
```

```
        print("You need to spend ${0:%.2f} more to receive a 5%  
            discount.".format(diff))
```

# Python Comparison Operators



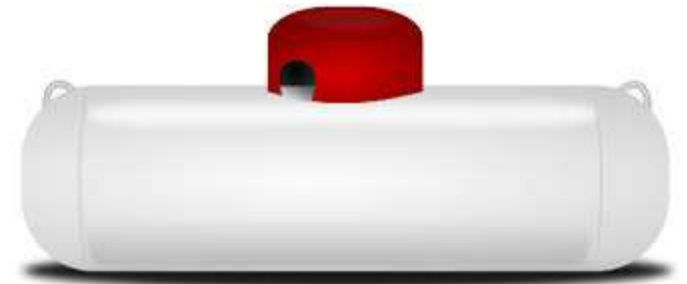
Equality Testing Requires two “=”

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

**Note: relational operators have a lower precedence than arithmetic operators**

# Sample IF Problem

- The variables `fuelAmount` and `fuelCapacity` hold the actual amount of fuel and the size of the fuel tank of a vehicle.
- If less than 10 percent is remaining in the tank, a status light should show a red color; otherwise, it shows a green color.
- Simulate this process by printing out either "red" or "green"



# A Better Way: elif

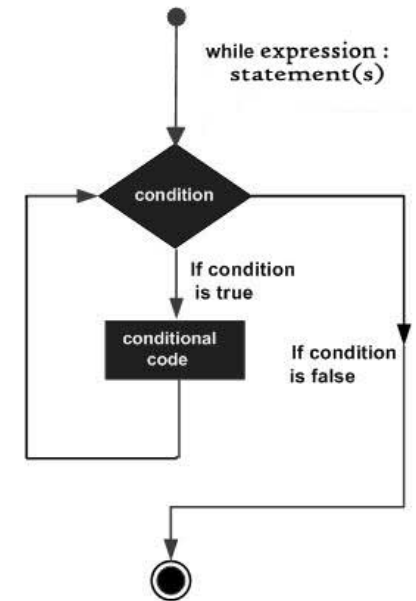
```
if (classScore >=90):  
    print("You got an A!")  
elif (classScore >= 80) :  
    print("You did ok, you got a B!")  
elif (classScore >=70) :  
    print("So-so, you got a C")  
elif (classScore >= 60) :  
    print("Oh –oh, you got a D")  
else:  
    print("Dang it, you got an F")
```



**Note that you have to test the more specific conditions first.**

# The While Statement

- A while loop in Python repeatedly executes a target statement as long as a given condition is true.
- The syntax of a while loop in Python programming language is:  
`while expression:  
 statement(s)`
- Here, `statement(s)` may be a single statement or a block of statements. The condition may be any expression, and true is any non-zero value. The loop iterates while the condition is true.
- When the condition becomes false, program control passes to the line immediately following the loop.
- In Python, all the statements indented by the same number of character spaces after a programming construct are considered to be part of a single block of code. Python uses indentation as its method of grouping statements.
- Here, a key point of the while loop is that ***the loop might not ever run.***

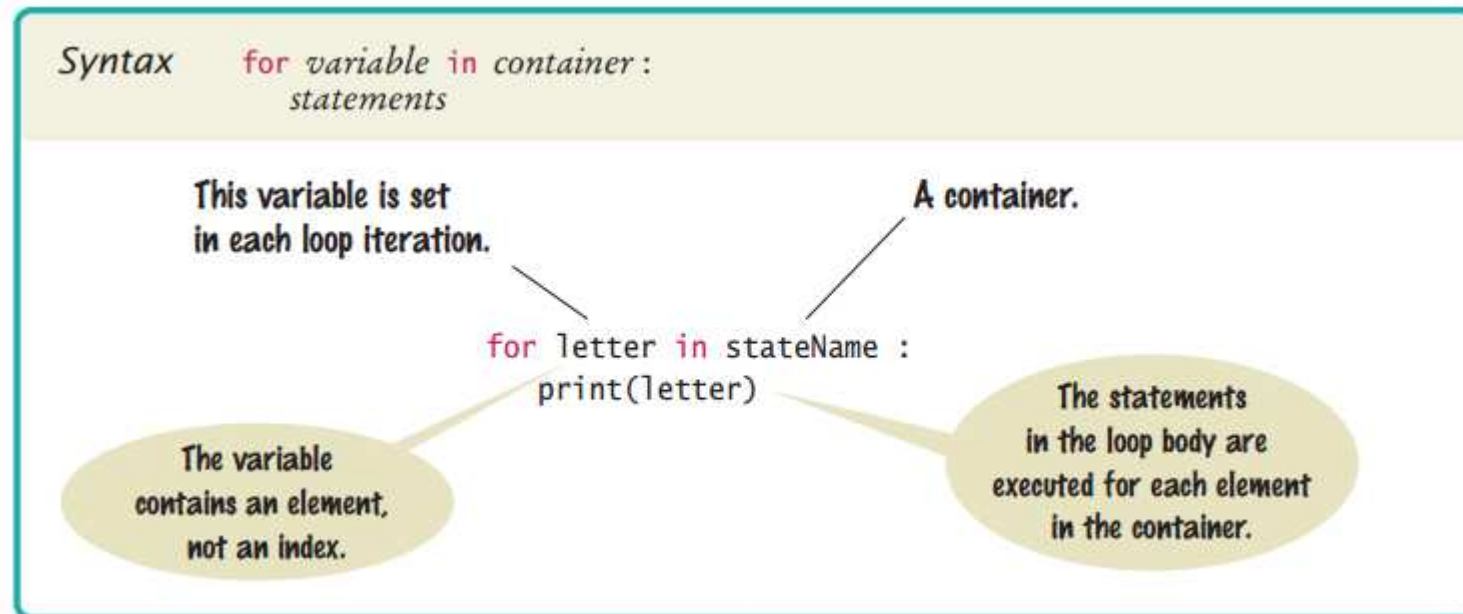


```
x=5  
while (x > 0):  
    print(x)  
    x = x-1
```



```
5  
4  
3  
2  
1
```

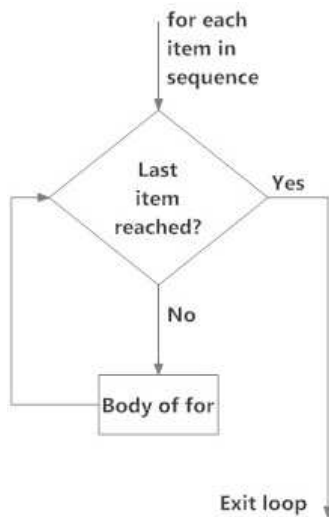
# The For Statement



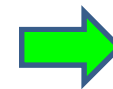
The for loop can be used to iterate over the contents of any **container**, which is an object that contains or stores a collection of elements. Thus, a string is a container that stores the collection of characters in the string.

# The For Statement

- **for** loops are traditionally used when you have a block of code which you want to repeat a **fixed number of times**.
- The Python **for** statement iterates over the members of a sequence in order, executing the block each time.
- Loop continues until we reach the last item in the sequence. The body of for loop is separated from the rest of the code using indentation.



```
for i in range(0,11):  
    print(i)  
print("all done")
```



```
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
all done
```



# The Range Function

- Count-controlled loops that iterate over a range of integer values are very common.
- To simplify the creation of such loops, Python provides the **range** function for generating a sequence of integers that can be used with the for loop.
- The loop code:

```
for i in range(1, 10) : # i = 1, 2, 3, ..., 9  
    print(i)
```

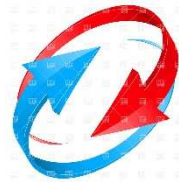


prints the sequential values from 1 to 9. The range function generates a sequence of values based on its arguments.

- The first argument of the range function is the first value in the sequence.
- **Values are included in the sequence while they are less than the second argument**

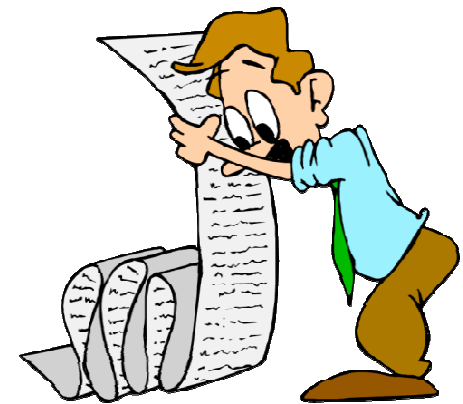
# Difference Between For and While

- A while loop uses a **boolean test** – you don't know how many times it's going to loop.
- A while loop is good when you don't know how many times to loop and just want to keep going while some condition is true.
- But if you know how many times to loop (e.g. the length of an array, 7 times, etc.), a for loop is cleaner.



# Python Is All About Lists

- What lists do you use in your life today?
- Shopping list: milk, banana, apples, oranges, coral, napkins, cups, oil, grapes
- To-Do List: pick up dry cleaning, go shopping, get gas, pay bills, get mail
- Christmas List: Mom, sister, brother, wife, kids, boss



# 4 Steps To Creating A Python List

## IEEE Florida West Coast Section 2022

1. **Convert** each of the names into strings by surrounding the data with quotes.

"IEEE" "Florida West Coast Section" 2022

2. **Separate** each of the list items from the next with a comma.

"IEEE", "Florida West Coast Section", 2022

3. **Surround** the list of items with opening and closing square brackets.

["IEEE", "Florida West Coast Section", 2022]

4. **Assign** the list to an identifier using the assignment operator (=).

prerequisites = ["IEEE", "Florida West Coast Section", 2022]



# What Does Your List Look Like Inside Of The Computer?

```
prerequisites = ["IEEE", "Florida West Coast Section", 2022]
```

The computer stores your list data in what is called an “array”.



Each item in a list has an index number associated with it...



2

1

0

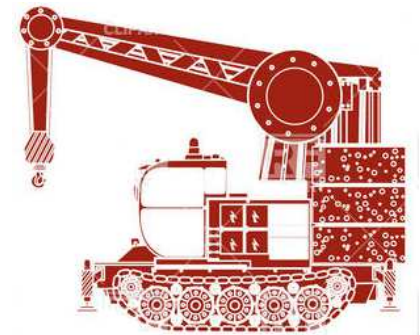
Python Starts Counting At 0

# How To Access A List

- A list is a sequence of *elements*, each of which has an integer position or *index*.
- To access a list element, you specify which index you want to use.
- That is done with the subscript operator `[]` in the same way that you access individual characters in a string.
- For example:

```
values = [9,8,7,6,5,4,3,2,1,0]
```

```
print(values[3]) # Prints the element at index 3 - 6
```



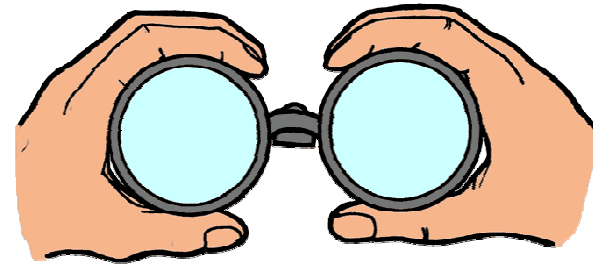
# Access Your List Data Using The Square Bracket Notation

```
prerequisites = ["IEEE", "Florida West Coast Section", 2022]
```

```
print (prerequisites[0]) → IEEE
```

```
print (prerequisites[1]) → Florida West Coast Section
```

```
print (prerequisites[2]) → 2022
```



# Calculating The Length Of A List

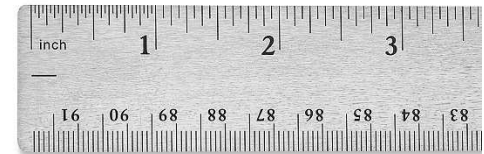
- Python has a built-in function `len()` for getting the total number of items in a list. The `len()` method takes an argument where you provide a list and it returns the length of the given list.

- This is how you use the `len()` method of Python:

```
len(object)
```

- For example, we have a list:

```
myList = [5, 10, 15, 20, 'a', 'b', 'c']
```



- This is how you use the `len()` method for getting the length of that list:

```
list_length = len(myList)  
print(list_length)
```

Answer: 7



# You Can Create Multidimensional Lists

- Lists can hold data of mixed type.
- But it gets even better than that: lists can hold collections of anything, *including other lists*.
- Simply **embed** the *inner* list within the *enclosing* list as needed.

multiDim = [[1,2,3],[4,5,6],[7,8,9]] =



1 2 3  
4 5 6  
7 8 9

# Lists Are Much More Than Just Simple Arrays

- Lists in Python are full-blown Python collection objects.
- This means that lists come with ready-to-use functionality in the form of **list methods**.

BIF  
len



## Methods

1. append
2. pop
3. remove
4. insert
5. extend

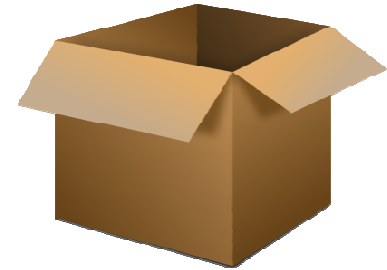
# Playing With Lists: Append

```
prerequisites = ["CHM 2045" , "Chemistry 1" , 3]
```

- Add teacher: Dr. Jim Anderson
- Add year: 2022

```
prerequisites.append("Dr. Anderson")
```

```
prerequisites.append(2022)
```



```
CHM 2045 , Chemistry 1 , 3 , Dr. Anderson , 2022
```

**Note:** Python lists can contain data of **mixed types**. You can mix strings with numbers within the *same* Python list. You can mix more than just strings and numbers -- you can store data of *any type* in a single list.

# Playing With Lists: Pop

- Allows you to remove the item that is at the end of a list or a given index value:

`prerequisites.pop()`

CHM 2045 , Chemistry 1 , 3 , Dr. Anderson , 2022 **X**

`prerequisites.pop(2)`

CHM 2045 , Chemistry 1 , **X** , Dr. Anderson , 2022



# Playing With Lists: Remove

- Allows you to specify which list item you want to remove no matter where in the list it is located

```
prerequisites.remove("Dr. Anderson")
```

CHM 2045 , Chemistry 1 , 3 , ~~Dr. Anderson~~



# Playing With Lists: Insert

- Allows you to add an item to a list in a specified location on the list

```
prerequisites.insert(2,"Really Hard")
```

```
CHM 2045 , Chemistry 1 , Really Hard , 3
```



# Playing With Lists: Extend

- Allows multiple list items to be added to an existing list

```
prerequisites.extend(["Dr. Anderson", "2022"])
```

CHM 2045 , Chemistry 1 , Really Hard , 3 , Dr. Anderson , 2022



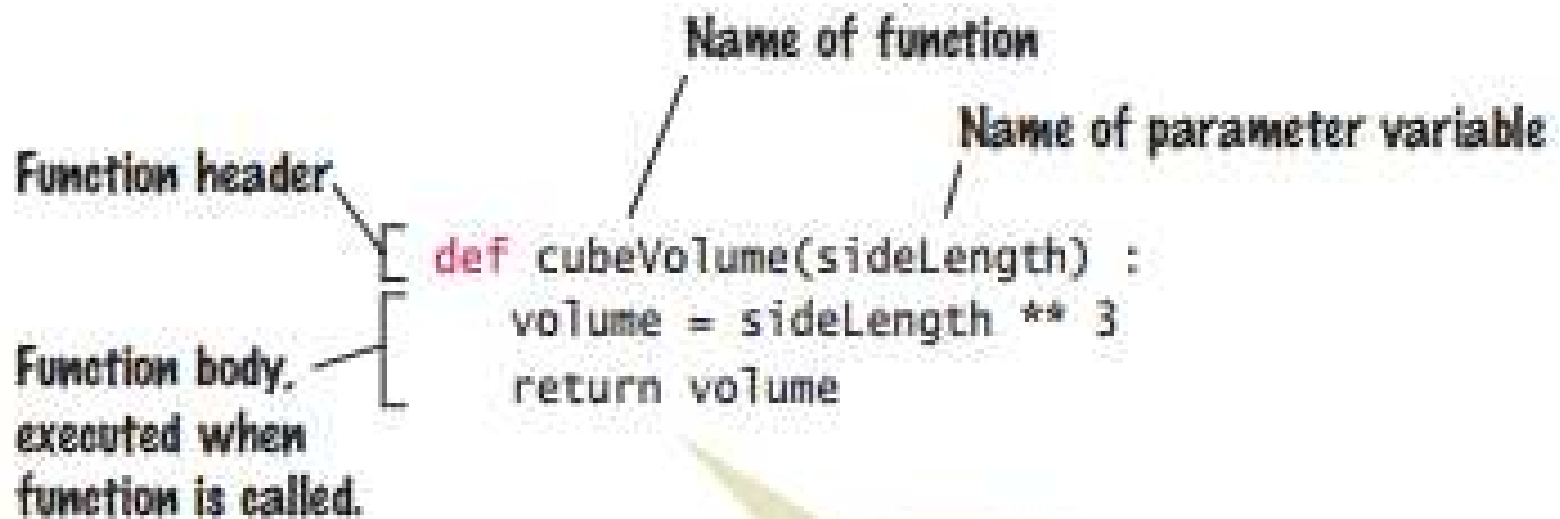
# The Dysfunctional Calendar Program



- You are in the process of creating a calendar program. But it's not going very well for you.
- You've been given the string ("Monday", "Tuesday", "Tuesday", "Thursday", "Friday")
- Turn this string into a list.
- Append "Saturday" and "Sunday" to the end of the list
- Insert "Wednesday" into the list in the correct position
- Check to make sure that "Friday" is in the list
- Remove the second "Tuesday" from the list



# Definition Of A Function

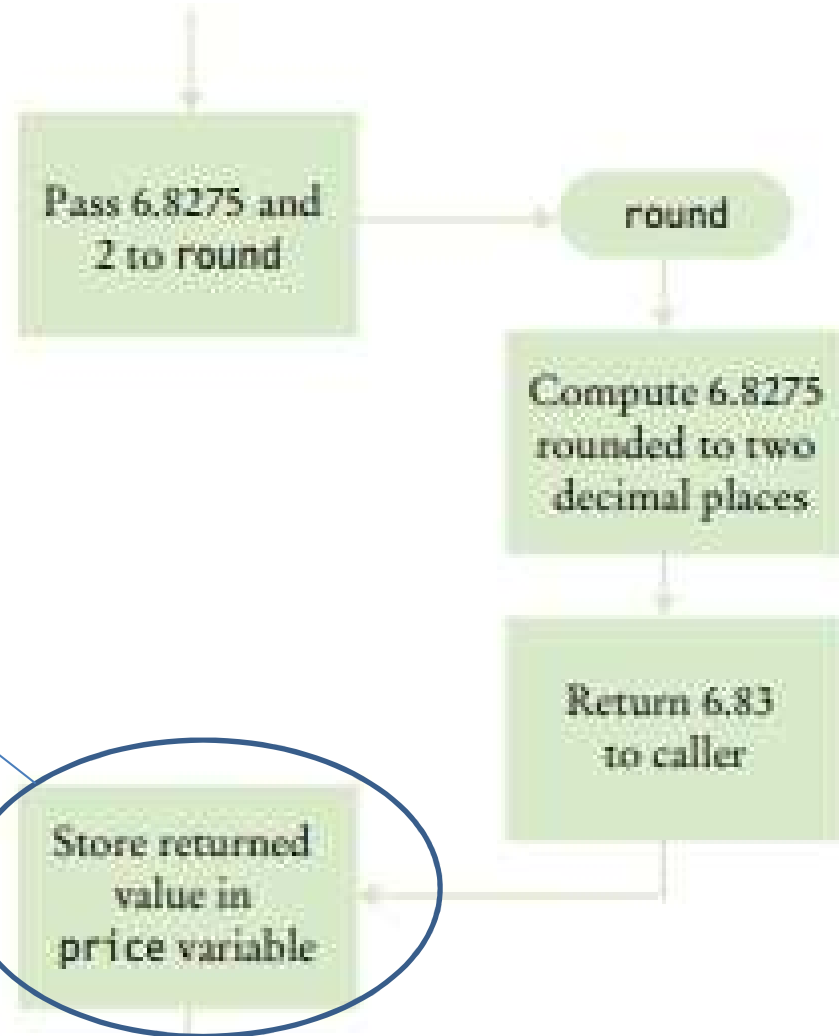


return statement  
exits function and  
returns result.

# What Happens When You Call A Function?

“Arguments”

price = round(6.8275, 2)



**Note:** Multiple arguments can be passed to a function. Multiple values can be returned.

# How To Create A Function

- When writing this function, you need to
  - Pick a name for the function (`selectFlavor`).
  - Define a variable for each argument (`yogurt`).
- These variables are called the **parameter variables**.
- Put all this information together along with the **def** reserved word to form the first line of the function's definition:

`def selectFlavor(yogurt) :`



- This line is called the **header** of the function.

# Create The Body Of The Function

- The body contains the statements that are executed when the function is called.

**def selectFlavor :**

Body {  
    selection = input("Please enter the flavor of yogurt you would like:")  
    print("Flavor selected", selection)



# Send The Result Back

- In order to return the result of the function, use the **return** statement:

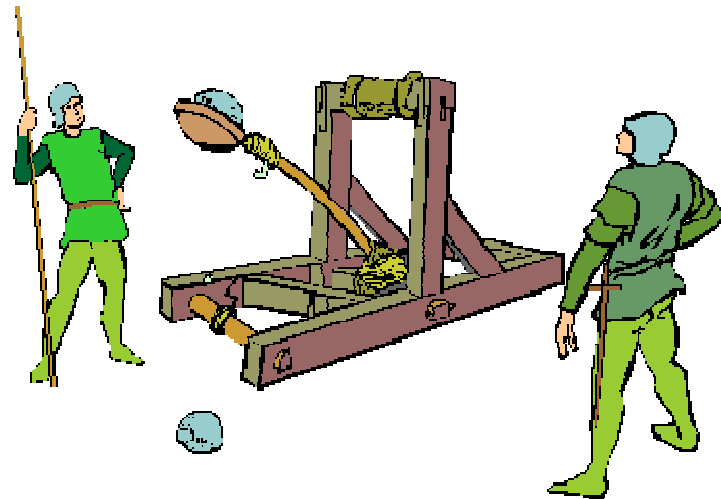
```
def getInput():  
    result = 1  
    return result
```

```
answer = getInput()  
print(answer)
```

- If you want, you can return multiple values:

```
def getInput():  
    result = 1  
    quantity = 10  
    return result, quantity
```

```
answer, amount = getInput()  
print(answer, amount)
```



If you forget to include a "return" statement, the function will return "none".

# Final Form Of Our Function

```
def selectFlavor() :  
    listOfFlavors = ["vanilla","chocolaate","lime","ginger",  
    print("Please choose from these flavors:")  
    for flavor in range(len(listOfFlavors)):  
        print(listOfFlavors[flavor])  
    print()  
  
    selection = input("Please enter the flavor of yogurt you would like: ")  
    print("Selected flavor: ",selection)  
  
    return (selection)
```



**Note:** A function is a compound statement, which requires the statements in the body to be indented to the same level.

# Order Matters!

- Python is an interpreted language. This means that it needs to “see” a function before you call it...



**YES**

```
def selectFlavor(yogurt) :  
    listOfFlavors = checkInventory()  
    while flavor in listOfFlavors  
        print(listOfFlavors[flavor])  
  
    if (yogurt == 1) :  
        selection = input("Please enter the flavor of yogurt you would like:")  
    else :  
        selection = input("Please enter the flavor of ice cream you would like:")  
  
    return selection
```

➔ **myChoice = selectFlavor(0)**



**NO**

➔ **myChoice = selectFlavor(0)**

```
def selectFlavor(yogurt) :  
    listOfFlavors = checkInventory()  
    while flavor in listOfFlavors  
        print(listOfFlavors[flavor])  
  
    if (yogurt == 1) :  
        selection = input("Please enter the flavor of yogurt you would like:")  
    else :  
        selection = input("Please enter the flavor of ice cream you would like:")  
  
    return selection
```



# Functions & Parameters



# Function Parameter Passing

- The values that are supplied to the function when it is called are the **arguments** of the call.
- When a function is called, variables are created for receiving the function's arguments.
- In the function these variables are called **parameter variables**.
- Each parameter variable is initialized with the corresponding argument.



# Parameter Passing: An Example

## 1. Function Call

```
myFlavor = selectFlavor(yogurt)
```

## 2. Initializing argument variable

```
myFlavor = selectFlavor(yogurt)
```

## 3. About to return to the caller

```
selection = input("Please enter flavor of yogurt you would like:")  
return selection
```

## 4. After function call

```
myFlavor = selectFlavor(yogurt)
```



# Return Values

- You use the return statement to specify the result of a function.
- The return statement can return the value of any expression.
- Instead of saving the return value in a variable and returning the variable, it is often possible to eliminate the variable and return the value of a more complex expression:



```
return ((numConesWanted*5)**2)
```

# Multiple Returns

- When the return statement is processed, the function exits *immediately*.
- Every branch of a function should return a value.

```
def selectFlavor(yogurt):  
    if not(yogurt) :  
        return (noFlavor)  
    else:  
        myFlavor = input("Enter your yogurt flavor")  
        return (myFlavor)
```



# Goals For This Presentation



1. Introduce you to the Python programming language
2. Show you how you can install Python on your home computer
3. Explain how variables work in Python
4. Explore how to get input and how to produce output from a program
5. Discover how to control program flow
6. Introduce the concepts of a list and what can be done with it
7. Explore functions and how they can be used to organize code

