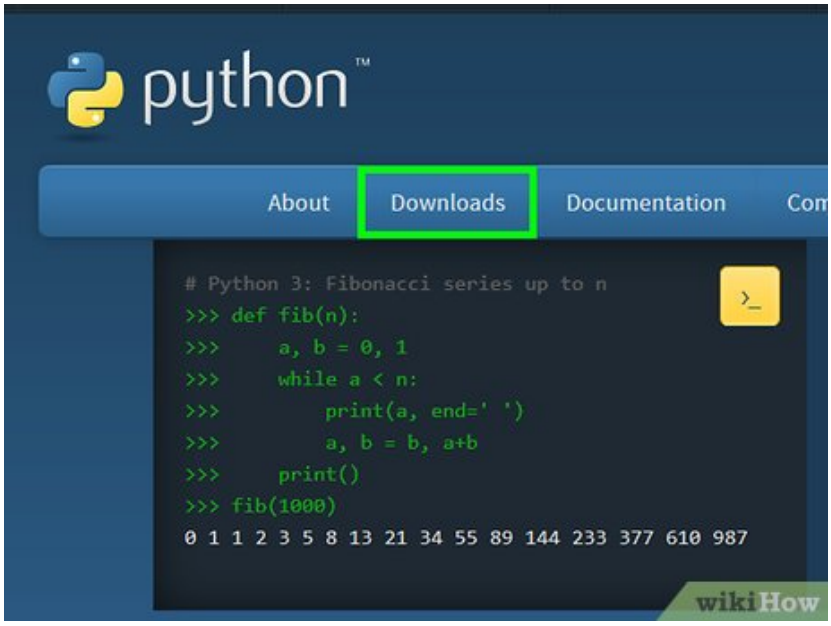
How to Start Programming in Python

Do you want to start learning how to program? Getting into computer programming can be daunting, and you may think that you need to take classes in order to learn. While that may be true for some languages, there are a variety of...

Part 1 of 5:

Installing Python (Windows)

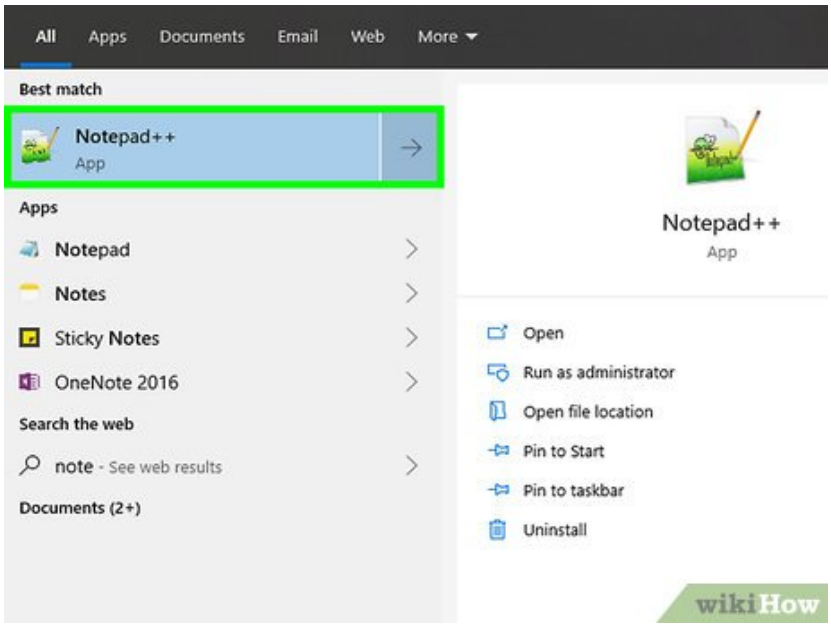
1. 

Download Python for Windows systems. The Windows Python interpreter can be downloaded for free from the Python website. Make sure to download the correct version for your operating system.

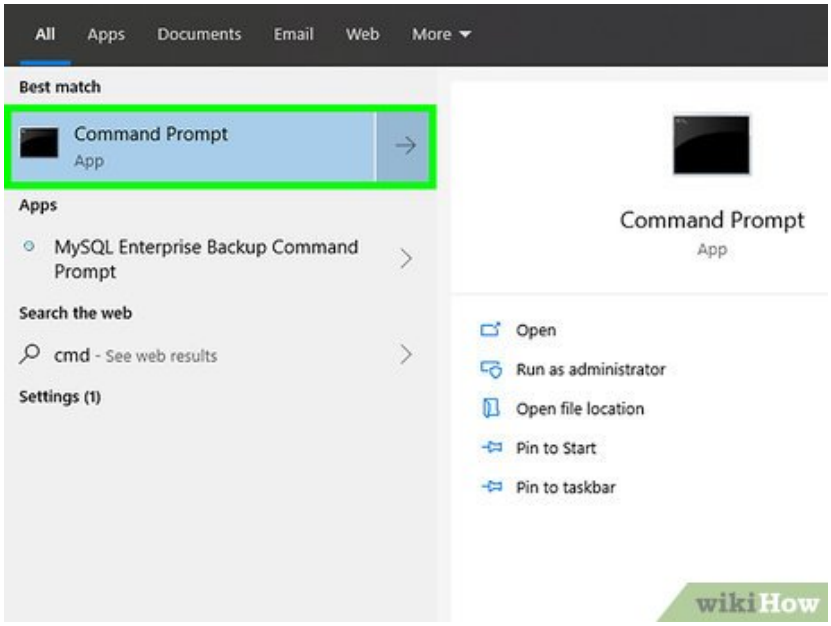
1. You should download the latest version available, which was 3.8 at the time of this writing.
2. OS X and Linux come with Python already installed. You will not need to install any Python-related software, but you may want to install a text editor.
3. Most Linux distributions and OS X versions still use Python 2.X. There are a few minor differences between 2 & 3, most notably the changes to the "print" statement. If you want to install a newer version of Python on OS X or Linux, you can download the files from the Python website.

2. 

Install the Python interpreter. Most users can install the interpreter without changing any settings. You can integrate Python into the Command Prompt by enabling the last option in the list of available modules. [2]

3. 

Install a text editor. While you can create Python programs in Notepad or TextEdit, you will find it much easier to read and write the code using a specialized text editor. There are a variety of free editors to choose from such as Notepad++ (Windows), TextWrangler (Mac), or JEdit (Any system).

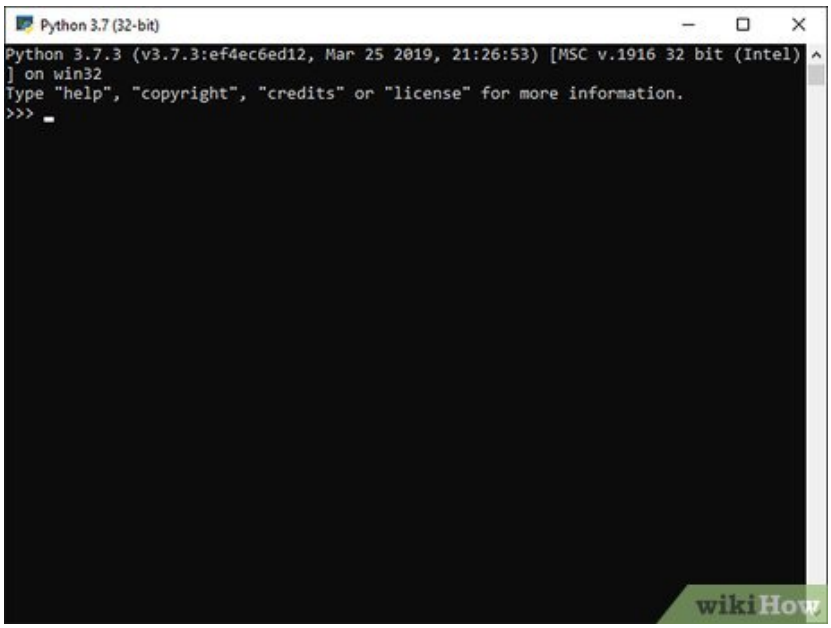
4. 

Test your installation. Open Command Prompt (Windows) of your Terminal (Mac/Linux) and type `python`. Python will load and the version number will be displayed. You will be taken to the Python interpreter command prompt, shown as `>>>`.

1. Type `print("Hello, World!")` and press ? Enter. You should see the text `Hello, World!` displayed beneath the Python command line.

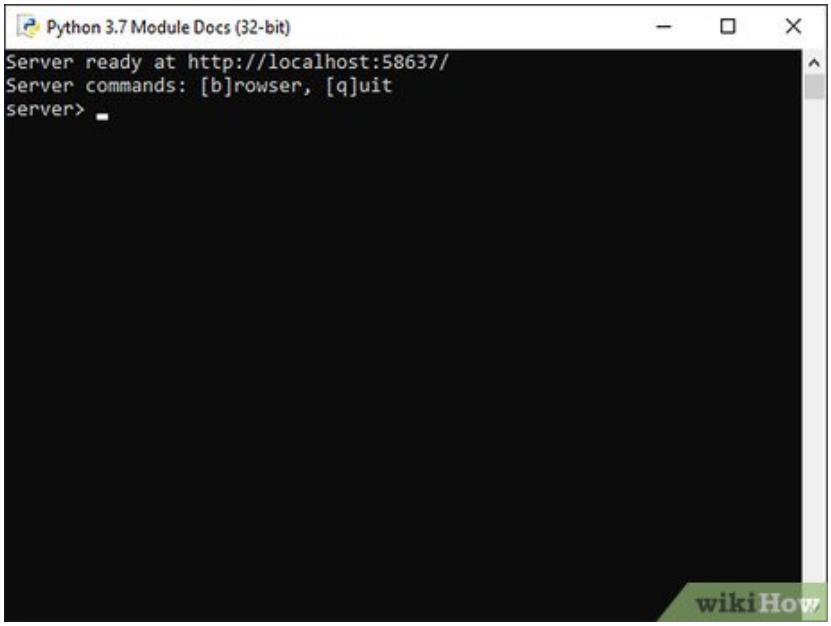
Part 2 of 5:

Learning Basic Concepts

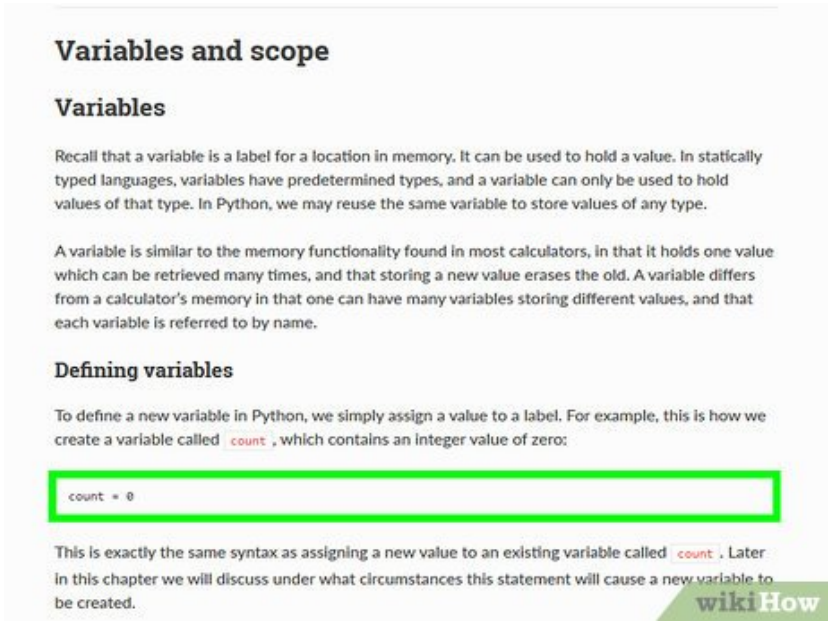
1. 

Understand that Python doesn't need to compile. Python is an interpreted language, which means you can run the program as soon as you make changes to the file. This makes iterating, revising, and

troubleshooting programs much quicker than many other languages.

1. Python is one of the easier languages to learn, and you can have a basic program up and running in just a few minutes.
2. 

Mess around in the interpreter. You can use the interpreter to test out code without having to add it to your program first. This is great for learning how specific commands work, or writing a throw-away program.

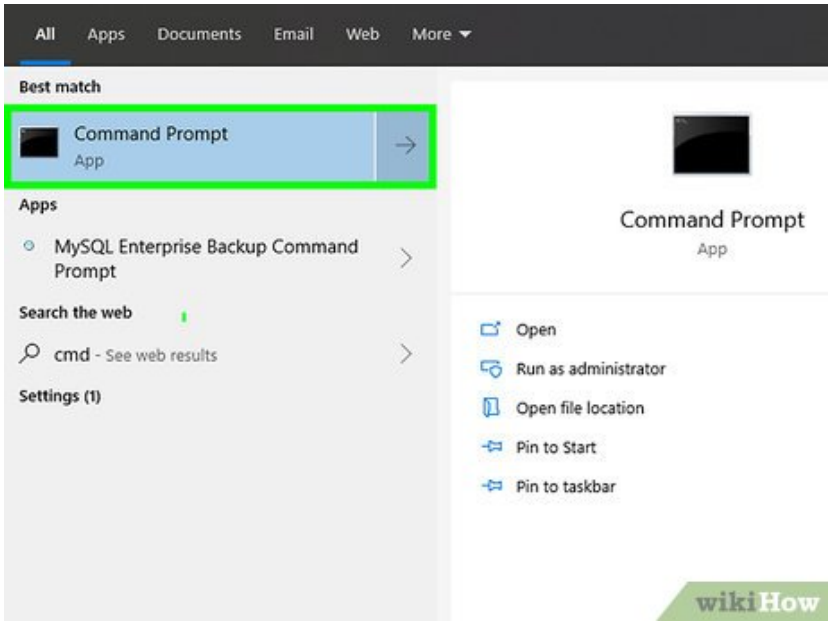
3. 

Learn how Python handles objects and variables. Python is an object-oriented language, meaning everything in the program is treated as an object. Also, you will not need to declare variables at the beginning of your program (you can do it at any time), and you do not need to specify the type of variable (integer, string, etc.).

Part 3 of 5:

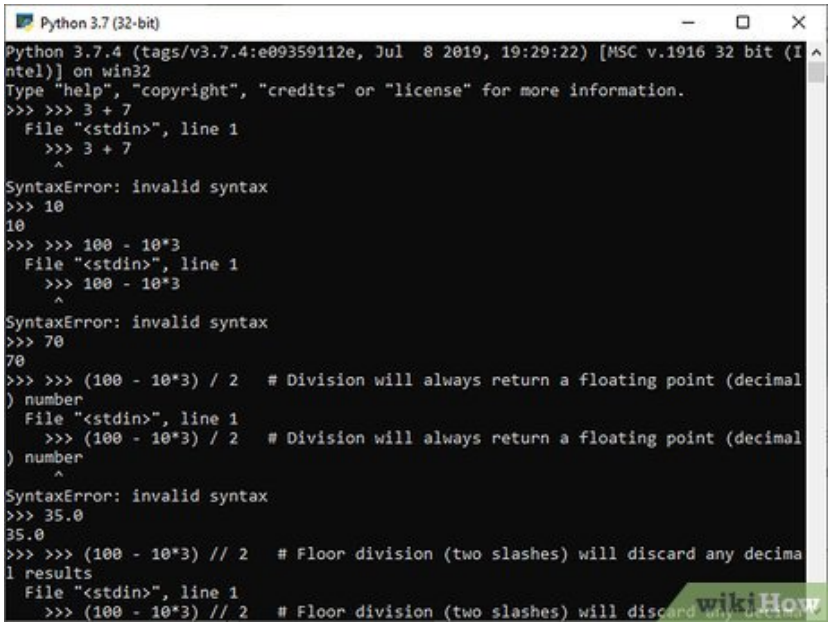
Using the Python Interpreter as a Calculator

Performing some basic calculator functions will help get you familiar with Python syntax and the way numbers and strings are handled.

1. 

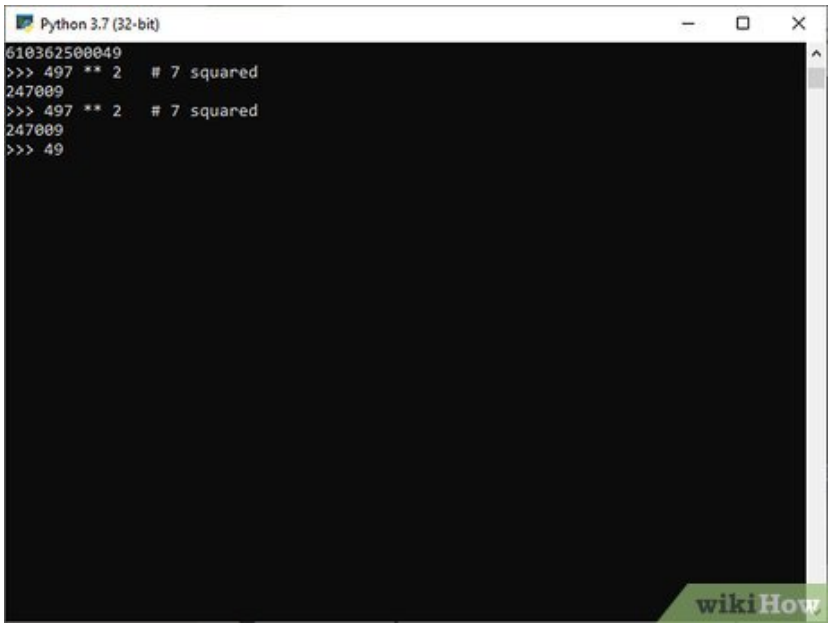
Start the interpreter. Open your Command Prompt or Terminal. Type `python` at the prompt and press ? Enter. This will load the Python interpreter and you will be taken to the Python command prompt (>>>).

1. If you didn't integrate Python into your command prompt, you will need to navigate to the Python directory in order to run the interpreter.

2. 

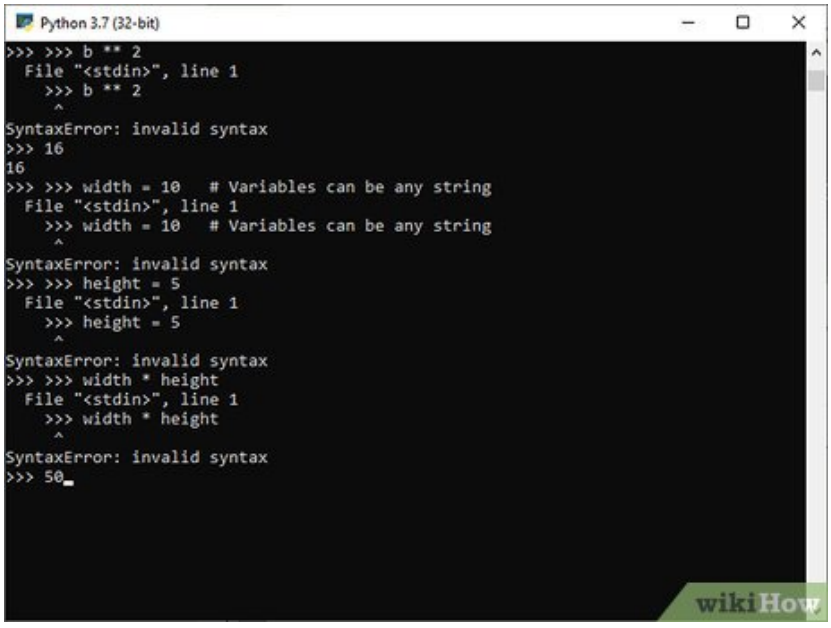
Perform basic arithmetic. You can use Python to perform basic arithmetic with ease. See the box below for some examples on how to use the calculator functions. Note: # designates comments in Python code, and they are not passed through the interpreter.

```
>>> 3 + 7 10 >>> 100 - 10*3 70 >>> (100 - 10*3) / 2
# Division will always return a floating point (decimal) number 35.0
>>> (100 - 10*3) // 2
# Floor division (two slashes) will discard any decimal results 35 >>>
23 % 4 # This calculates the remainder of the division 3 >>> 17.53 *
2.67 / 4.1 11.41587804878049
```

3. 

Calculate powers. You can use the `**` operator to signify powers. Python can quickly calculate large numbers. See the box below for examples.

```
>>> 7 ** 2 # 7 squared 49 >>> 5 ** 7 # 5 to the power of 7 78125
```

4. 

Create and manipulate variables. You can assign variables in Python to perform basic algebra. This is a good introduction to how to assign variables within Python programs. Variables are assigned by using the `=` sign. See the box below for examples.

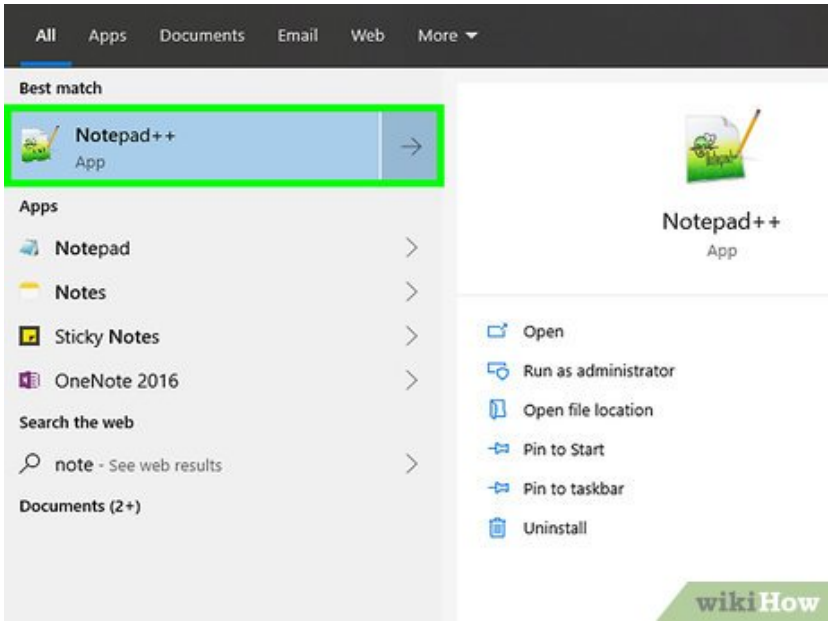
```
>>> a = 5 >>> b = 4 >>> a * b 20 >>> 20 * a // b 25 >>> b ** 2 16 >>> width = 10 # Variables can be any string >>> height = 5 >>> width * height 50
```

5. 

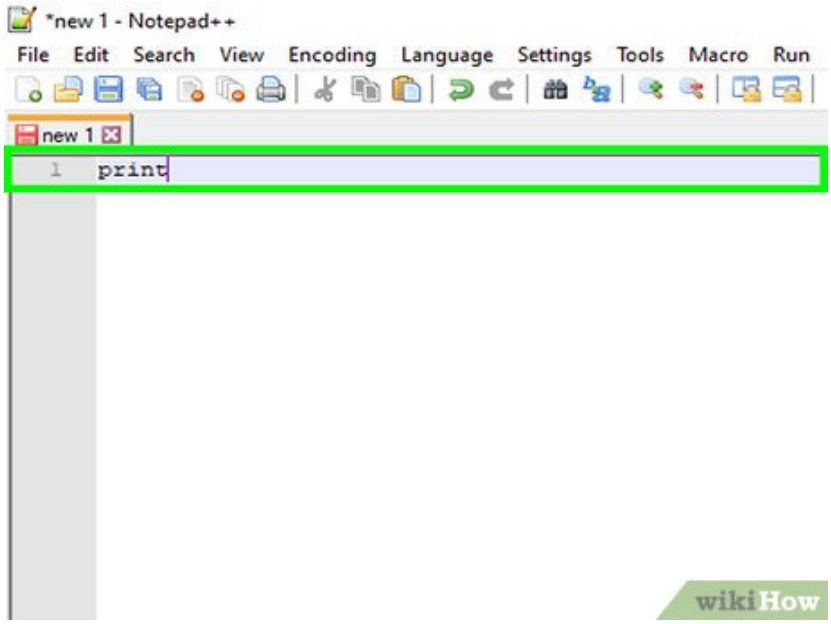
Close the interpreter. Once you are finished using the interpreter, you can close it and return to your command prompt by pressing Ctrl + Z (Windows) or Ctrl + D (Linux/Mac) and then pressing ? Enter. You can also type `quit()` and press ? Enter.

Part 4 of 5:

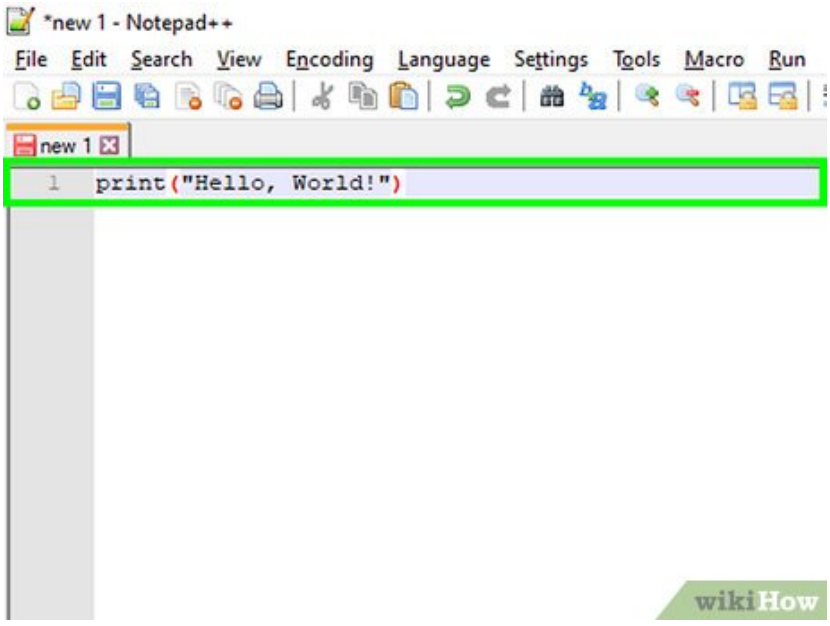
Creating Your First Program

1. 

Open your text editor. You can quickly create a test program that will get you familiar with the basics of creating and saving programs and then running them through the interpreter. This will also help you test that your interpreter was installed correctly.

2. 

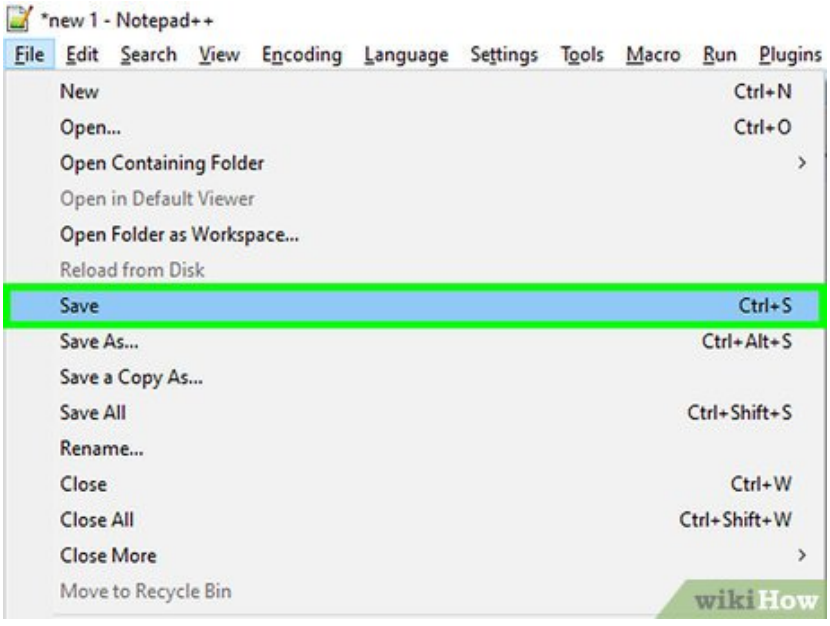
Create a "print" statement. "Print" is one of the basic functions of Python, and is used to display information in the terminal during a program. Note: "print" is one of the biggest changes from Python 2 to Python 3. In Python 2, you only needed to type "print" followed by what you wanted displayed. In Python 3, "print" has become a function, so you will need to type "print()", with what you want displayed inside the parentheses.

3. 

Add your statement. One of the most common ways to test a programming language is to display the text "Hello, World!" Place this text inside of the "print()" statement, including the quotation marks:

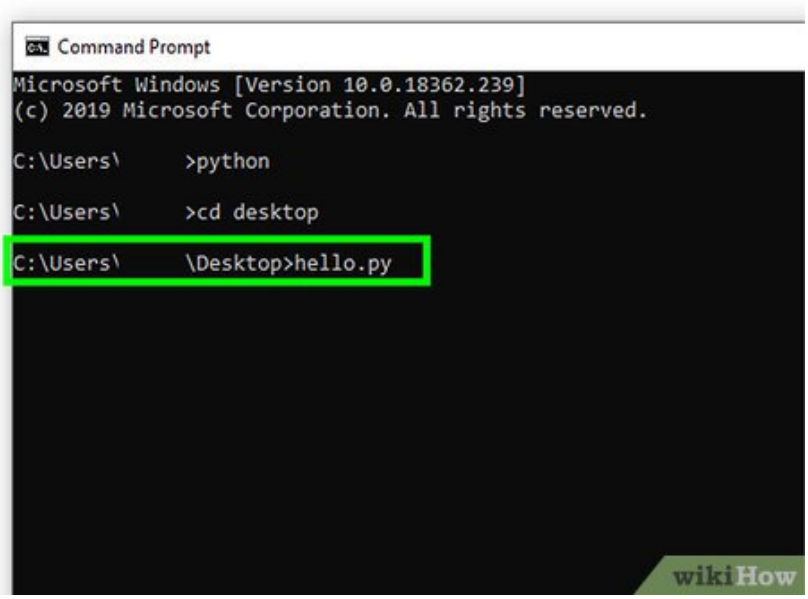
```
print("Hello, World!")
```

1. Unlike many other languages, you do not need to designate the end of a line with a `;`. You also will not need to use curly braces (`{}`) to designate blocks. Instead, indenting will signify what is included in a block.

4. 

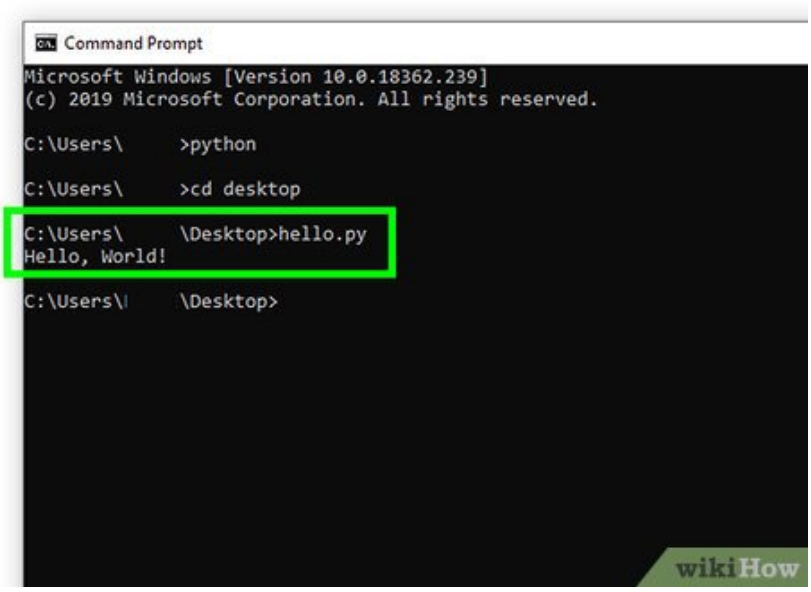
Save the file. Click the File menu in your text editor and select Save As. In the dropdown menu beneath the name box, choose the Python file type. If you are using Notepad (not recommended), select "All Files" and then add ".py" to the end of the file name.

1. Make sure to save the file somewhere easy to access, as you will need to navigate to it in the command prompt.
2. For this example, save the file as "hello.py".

5. 

Run the program. Open your Command Prompt or Terminal and navigate to the location where you saved your file. Once you are there, run the file by typing `hello.py` and pressing ? Enter. You should see the text `Hello, World!` displayed beneath the command prompt.

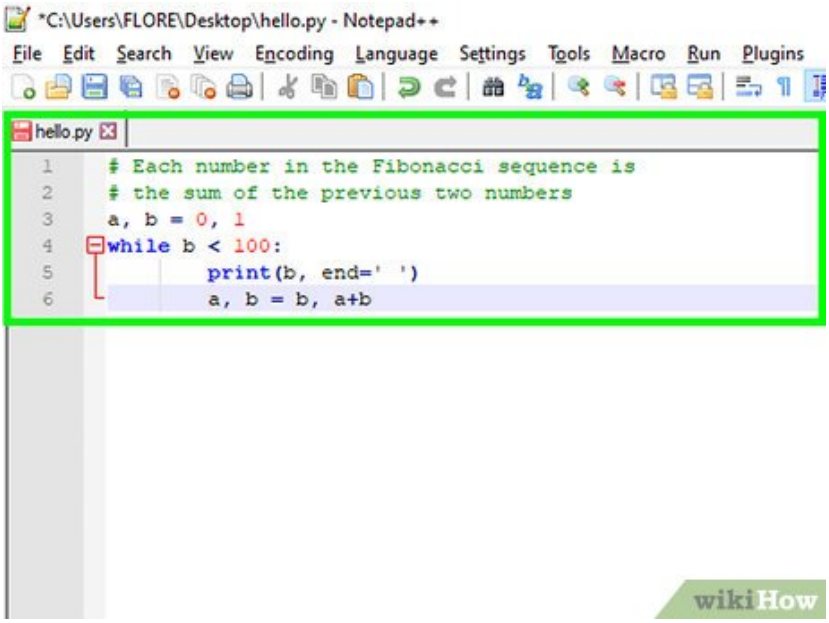
1. Depending on how you installed Python and what version it is, you may need to type `python hello.py` or `python3 hello.py` to run the program.

6. 

Test often. One of the great things about Python is that you can test out your new programs immediately. A good practice is to have your command prompt open at the same time that you have your editor open. When you save your changes in your editor, you can immediately run the program from the command line, allowing you to quickly test changes.

Part 5 of 5:

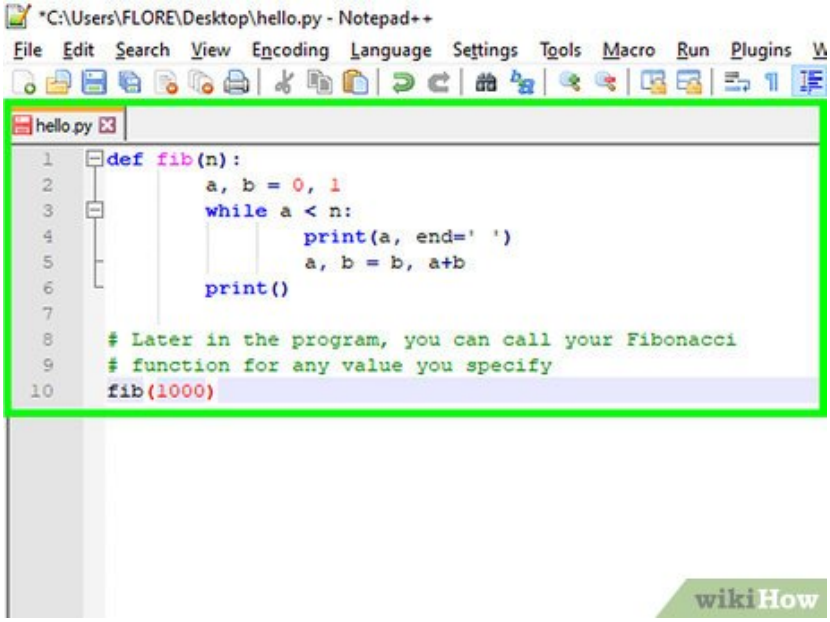
Building Advanced Programs

1. 

Experiment with a basic flow control statement. Flow control statements allow you to control what the program does based on specific conditions.[3] These statements are the heart of Python programming, and allow you to create programs that do different things depending on input and conditions. The `while` statement is a good one to start with. In this example, you can use the `while` statement to calculate the Fibonacci sequence up to 100:

```
# Each number in the Fibonacci sequence is
# the sum of the previous two numbers  a, b = 0, 1 while b  100: print(
b, end=' ') a, b = b, a+b
```

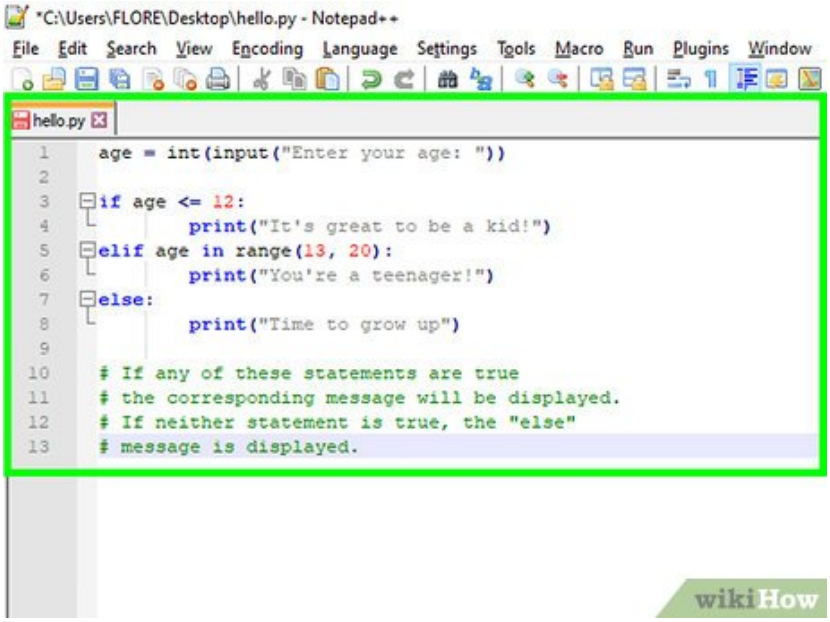
1. The sequence will run as long as (while) *b* is less than () 100.
2. The output will be `1 1 2 3 5 8 13 21 34 55 89`
3. The `end=' '` command will display the output on the same line instead of putting each value on a separate line.
4. There are a couple things to note in this simple program that are critical to creating complex programs in Python:
    1. Make note of the indentation. A `:` indicates that the following lines will be indented and are part of the block. In the above example, the `print(b)` and `a, b = b, a+b` are part of the `while` block. Properly indenting is essential in order for your program to work.
    2. Multiple variables can be defined on the same line. In the above example, *a* and *b* are both defined on the first line.
    3. If you are entering this program directly into the interpreter, you must add a blank line to the end so that the interpreter knows that the program is finished.

2. 

Build functions within programs. You can define functions that you can then call on later in the program. This is especially useful if you need to use multiple functions within the confines of a larger program. In the following example, you can create a function to call a Fibonacci sequence similar to the one you wrote earlier:[4]

```
def fib(n): a, b = 0, 1 while a  n: print(a, end=' ') a, b = b, a+b
print() # Later in the program, you can call your Fibonacci
# function for any value you specify fib(1000)
```

1. This will return `0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987`

3. 

Build a more complicated flow control program. Flow control statements allow you to set specific conditions that change how the program is run. This is especially important when you are dealing with user input. The following example will use the `if`, `elif` (else if), and `else` to create a simple

program that evaluates the user's age.[5]

```
age = int(input("Enter your age: ")) if age = 12: print(
"It's great to be a kid!") elif age in range(13, 20): print(
"You're a teenager!") else: print("Time to grow up")
# If any of these statements are true
# the corresponding message will be displayed.
# If neither statement is true, the "else" # message is displayed.
```

1. This program also introduces a few other very important statements that will be invaluable for a variety of different applications:
    1. `input()` - This invokes user input from the keyboard. The user will see the message written in the parentheses. In this example, the `input()` is surrounded by an `int()` function, which means all input will be treated as an integer.
    2. `range()` - This function can be used in a variety of ways. In this program, it is checking to see if the number in a range between 13 and 20. The end of the range is not counted in the calculation.

4. *Conditional Expressions.* [6]

Meaning	Symbol	Python Symbol
Less than	<	<
Greater than	>	>
Less than or equal	≤	<=
Greater than or equal	≥	>=
Equals	=	==
Not equal	≠	!=

wikiHow

Learn the other conditional expressions. The previous example used the "less than or equal" (=) symbol to determine if the input age met the condition. You can use the same conditional expressions that you would in math, but typing them is a little different:*Conditional Expressions.* [6]MeaningSymbolPython Symbol Less than>Less than or equal?=Greater than or equal?>=Equals===Not equal?!=

5. 

Continue learning. These are just the basics when it comes to Python. Although it's one of the simplest languages to learn, there is quite a bit of depth if you are interested in digging. The best way to keep learning is to keep creating programs! Remember that you can quickly write scratch programs directly in the interpreter, and testing your changes is as simple as running the program from the command line again.

1. There are lots of good books available for Python programming, including, "Python for Beginners", "Python Cookbook", and "Python Programming: An Introduction to Computer Science".
2. There are a variety of sources available online, but many are still geared towards Python 2.X. You may need to make adjustments to any examples that they provide.
3. If you want to run python online but wish to run python 3, Repl[1] has a python interpreter that uses virtual linux machines. Another good online resource for a future "pythonista" (well-versed python programmer) is thinkfunctional[2]. For bigger challenges, "Automate the Boring Stuff"[3] and Project Euler[4] are also available.
4. Many local schools offer classes on Python. Oftentimes Python is taught in introductory classes as it is one of the easier languages to learn.

Sample Programs

Picture 24 of How to Start Programming in Python
Sample Python Interpreter Startup Code

Picture 25 of How to Start Programming in Python
Sample Python Calculator Code

Picture 26 of How to Start Programming in Python
Sample Easy Python Program

tips and guides. Thank you for reading and for following us regularly.