A Blog Article by

Geek Must Have

Python Journeys
Getting Started
GeekMustHave

John HR Schuster

Version 2.1b, 11/07/2018

Table of Contents

It is the day before Thanksgiving, what I have come to call a `Learning Day`. A day where I pick something I need to learn more about and just get started.

Python is everywhere. From the small little Gemma Zero from Adafruit all the way up to the large parallel systems. This should be my second language next to JavaScript/Node.

1. Install Python Windows

Verify Version

Python Version

```
PS D:\Users\jschust2\Dropbox\myDev\Python-Journey\GettingStarted> python --version
Python 3.6.3
```

1.1. Install Latest version

If installed version is much older than current, update it.

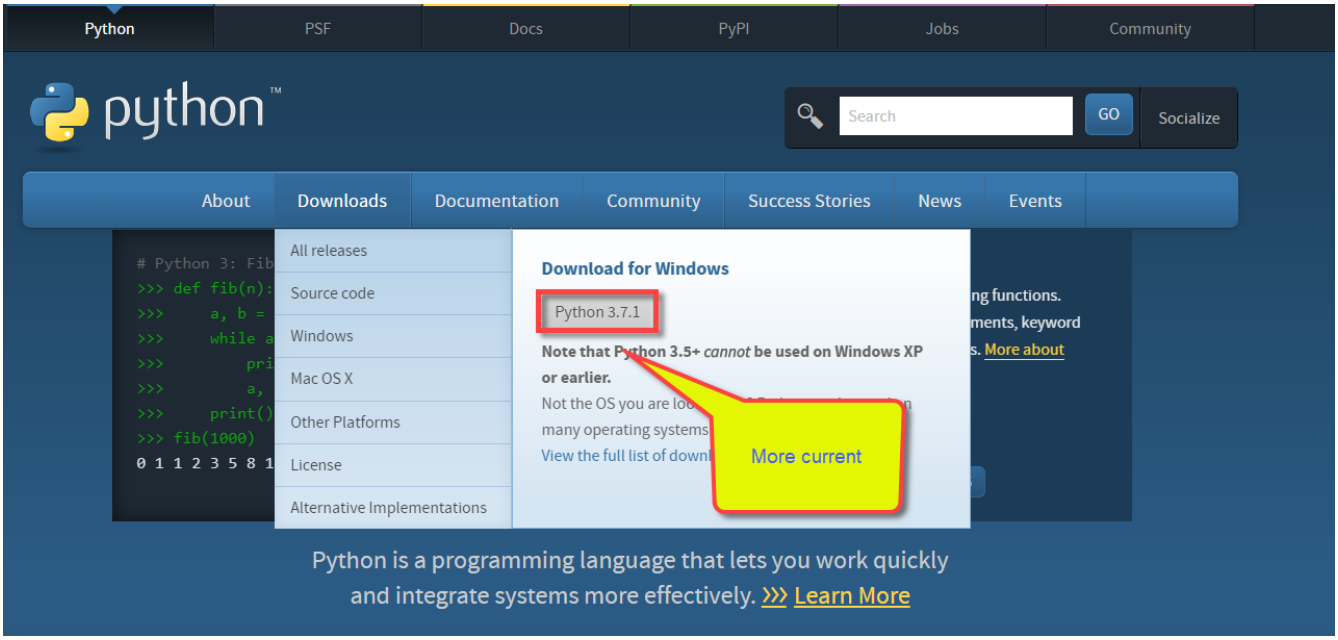

Figure 1. Python Update

Check the Python path option.

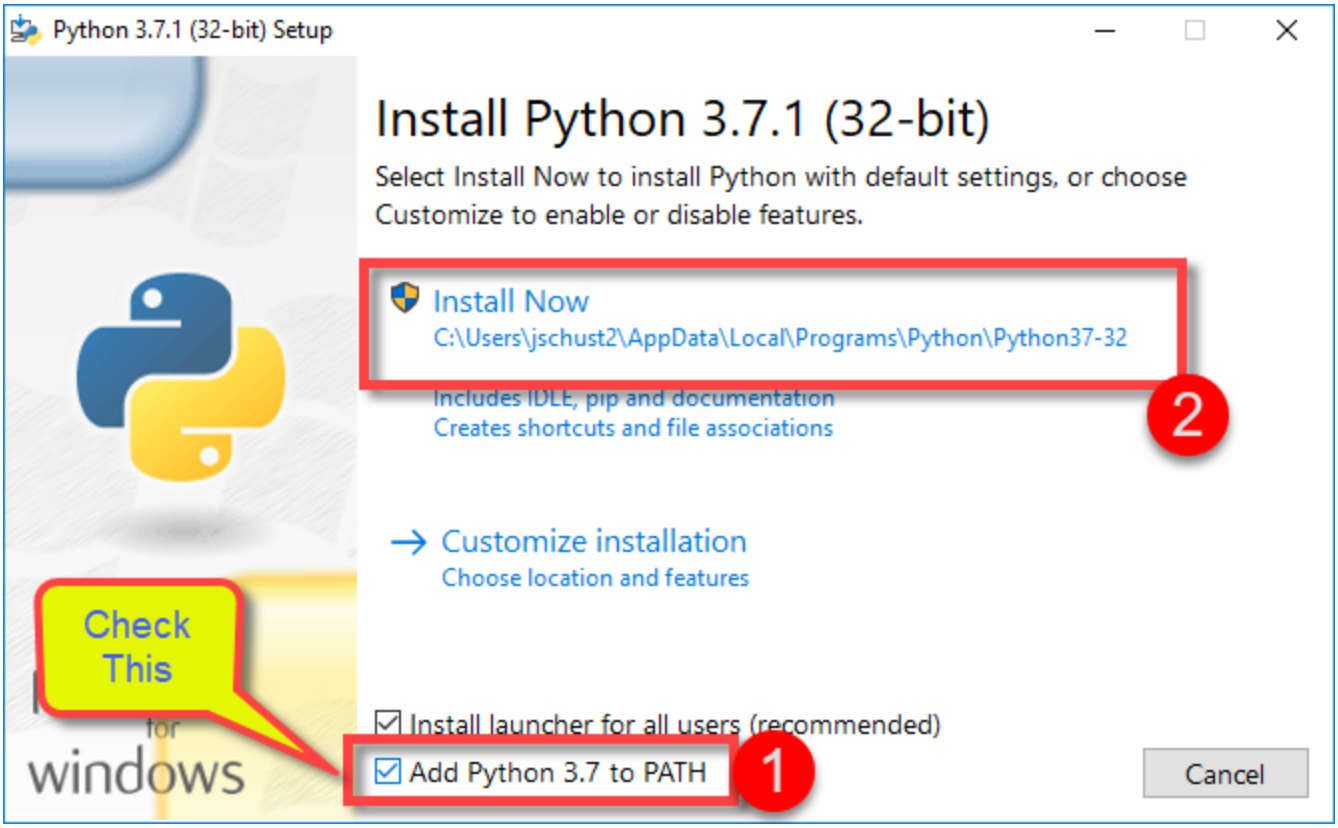

Figure 2. Python Options

If you are running Python from within vsCode you may have to restart vsCode to get the most current version of Python.

2. vsCode and Python

Microsoft has an vsCode extension with 23 million downloads, sounds safe.

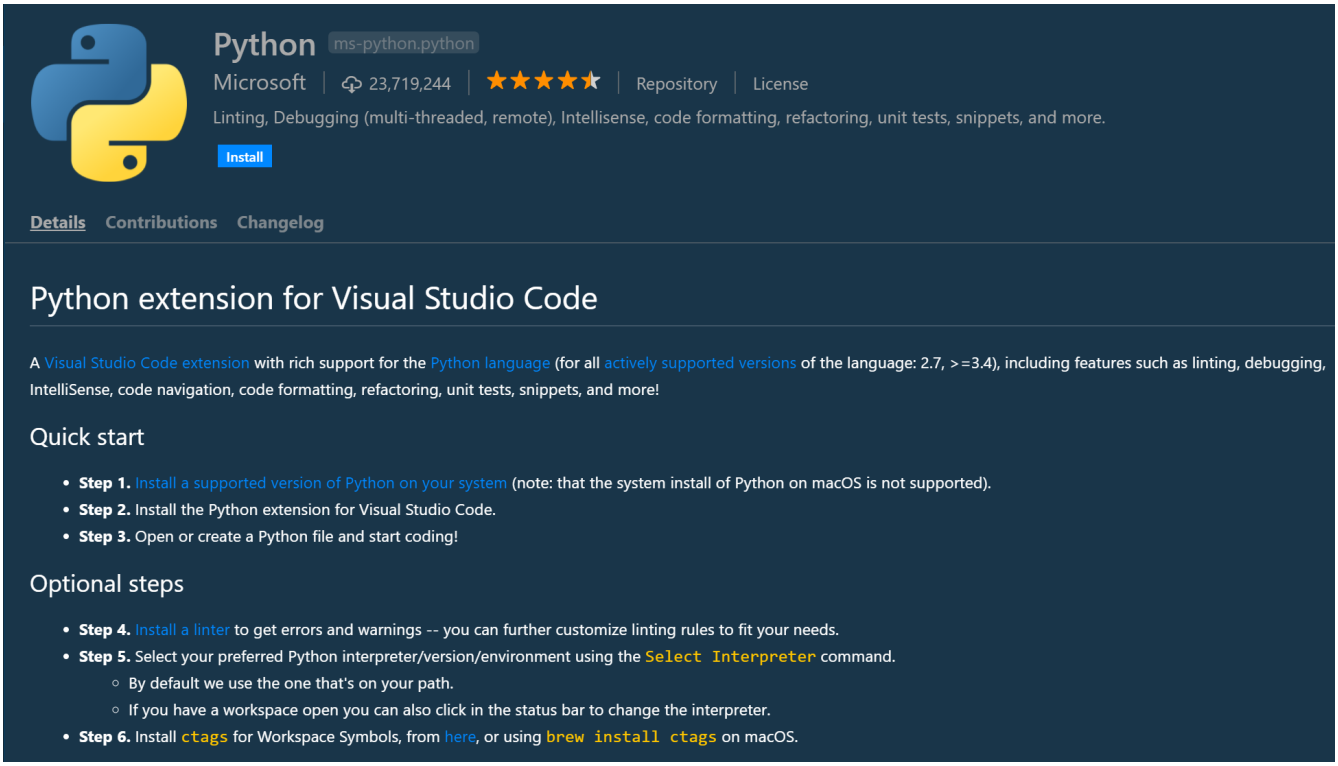

Figure 3. Microsoft Python Extension

2.1. Install the linter

Install Python Linter

```
python -m pip install -U pylint --user
```

3. Python basics

3.1. Syntax

- Python has no mandatory statement termination characters and blocks are specified by indentation.
- **Indent to begin a block**, dedent to end one. Statements that expect an indentation level end in a colon (:).
- Comments start with the pound (#) sign and are single-line
- Multi-line strings are used for multi-line comments.
- Values are assigned (in fact, objects are bound to names) with the equals sign (`=`)
- Equality testing is done using two equals signs (`==`).
- Increment/decrement values using the `+=` and `-=` operators respectively by the right-hand amount.
- Keywords are the reserved words in Python. Keywords are case sensitive.
- Statements can extend over multiple lines with the line continuation character (`\`).

- Boolean `True` is a `1`, 'False` is a `0`
- `Null` is now `None`

3.2. Object-Orientated Programming

Object-Oriented Programming (OOP) comes built into the Python language. It lays down a model which derive a solution by creating objects, defining relationships, and binding data. However, the procedural programming methodology takes on a top-down approach and solves one problem at a time while splitting it into smaller ones. On the other hand, OOP is a bottom-up problem-solving technique which seeks a blueprint of the solution at the onset and leaves implementation for the later.

3.3. Clear Screen

This is much more complicated than it needed to be. Whats wrong with `clear`?

Clear Screen

```
import os
_=os.system("clear")
```

3.4. Literal Collections

- **List** - `fruits = ['apple', 'orange', 'bannana']`
- **Tuple** - `numbers = (1, 2, 3, 5, 8, 13)`
- **Dictionary** -`ham = {'a':'Alpha', 'b': 'Bravo', 'c': 'Charlie'}`
- **Set** - `vowels = {'a', 'e', 'i', 'o', 'u'}`

3.5. Data types

Every value in Python has a datatype. Since everything is an object in Python programming, data types are actually classes and variables are instance (object) of these classes.

Integers - as big a number as you need

Float - up to 15 decimal points

Complex - `x + yj` where `x` is the real part, `y` is the imaginary part???

Lists - `a = [5,10,12,32,118]` then `a[3]` would be `32` indexing starts at zero, a[3]=666 replaces item in list. Very much like a simple array

Tuple - Tuple is an ordered sequence of items same as list.The only difference is that tuples are immutable. Tuples once created cannot be modified.

Tuples are used to write-protect data and are usually faster than list as it cannot change

dynamically.

`stuff = ('Diodes', 45, 1.25)`, then `stuff[1]` would be `45`.

Sets - Unordered collection of unique items. Set is defined by values separated by comma inside braces { }. Items in a set are not ordered. Since, set are unordered collection, indexing has no meaning. Hence the slicing operator [] does not work. Set have unique values. They eliminate duplicates.

`a = {1,2,2,3,3,3,4,4,5}` then `a` becomes `{1,2,3,4,5}`

Dictionary - Dictionary is an unordered collection of key-value pairs. It is generally used when we have a huge amount of data. Dictionaries are optimized for retrieving data. We must know the key to retrieve the value.

So `morse = {'a': ".-", 'b': "..", 'c': "._."}` and `morse['a']` would return `.-`

4. `constants.py` Program

constants.py program code

```
# --- Typical usage is a constant.py file for constants????

PI = 3.1415
GRAVITY = 9.8

# --- Used in other files
#
# import constant
# print(constant.PI)
```

5. `python1.py` Program

python1.py program code

```
# --- GeekMustHave Python - Getting Started

# --- All this just to clear the screen???
import os
_=os.system("clear")

message = "GeekMustHave, Python Journeys - Getting started "
message += "11/21/2018 Happy Learning Day"
print(message)

# --- Def is the equilivant of function
#     DocString can be first line of Def triple double-quotes """Description"""

def double(num):
```

```
  """Function to double the value sent to it"""
  return num*2

print(double(4))  #--- Run the function
print(double.__doc__)  #--- Show documentation about function

# --- Multiple variables, one line
firstName, lastName, age, pi = 'John', 'Schuster', 65, 3.1415

print("First name: " + firstName)
print("Last Name: " + lastName)
print("My age: " + str(age))  #--- standard (str) function
print("Piece of Pi: " + str(pi))
print("Pi Age: " + str(age * pi))

import constant
print(constant.PI)   #--- Not the same as lowercase (pi)

# --- Lists [] [] [] [] []
a = [5,10,12,32,118]
print('Forth item in list: ' + str(a[3]))  #--- should be 32 as lists are zero based
index
a[3]=988
print('Forth item in list: ' + str(a[3]))  #--- should be 988 as lists are zero based
index

# --- Tuples - Can't be changed () () () ()
vowels = ('a', 'e', 'i', 'o', 'u')
print("Second item in set: " + vowels[1])

# --- Sets - Unordered lists, not reordered, dups removed {} {} {} {}
ids = {192, 188, 203, 188, 144, 192, 188}
print("Sets: ")
print(ids)

# --- Dictionary - Unordered key:value  {:} {:} {:}, no indexed 0,1,2 access, key only
morse = {'a': ".-", 'b': "_..", 'c': "_._."}
print("Morse for 'a': " + morse['a'])
```

5.1. python1.py results

Here are the results from the python program

Python Results

```
GeekMustHave, Python Journeys - Getting started 11/21/2018 Happy Learning Day
8
Function to double the value sent to it
First name: John
Last Name: Schuster
My age: 65
Piece of Pi: 3.1415
Pi Age: 204.19750000000002
3.1415
Forth item in list: 32
Forth item in list: 988
Second item in set: e
Sets:
{192, 203, 188, 144}
Morse for 'a': .-
```

6. Resources

Python Wiki

https://wiki.python.org/moin/BeginnersGuide/Programmers