Python Applications for Digital Design and Signal Processing

Applications for Digital Design and Signal Processing Session 1

Dan Boschen

While every precaution has been taken in the preparation of this presentation, the author, publisher, and distribution partners assume no responsibility for any errors or omissions, or any damages resulting from the use of any information contained within it.

Course Outline

Session	Topics
1	**Course Intro: Python, Spyder and Jupyter**
2	Core Python
3	Core Python
4	Core Python
5	Python Modules and Packages
6	NumPy
7	NumPy, SciPy
8	Python for Verification, Modelling and Analysis

Session 1 Contents

Goals for this Session: Introduce the course, getting started with the Spyder IDE, Ipython, and Jupyter Notebooks.

Contents	Slides
Why Python – motivations for using Python	6 - 11
Design Examples – First overview of design examples to be done	12-15
Introduction of "Python for Verification"	16
Spyder IDE	17-22
Simple coding example using Spyder	23
Ipython	26-28
Jupyter Notebooks	29-30

SciPy

matplotlib

NumPy

jupyter

SPYDER

ANACONDA

python

Why Python?

Python is

Free, open source

Extremely **user friendly**

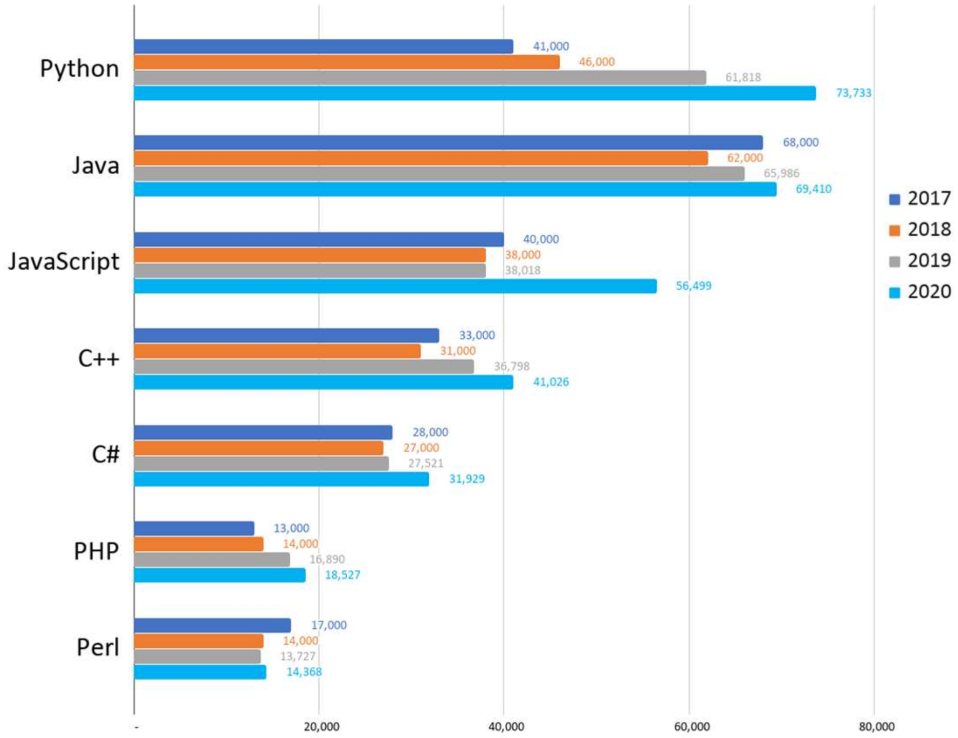
Easy to learn, read and write

Interpreted, **cross-platform**

Supported by a **large community**

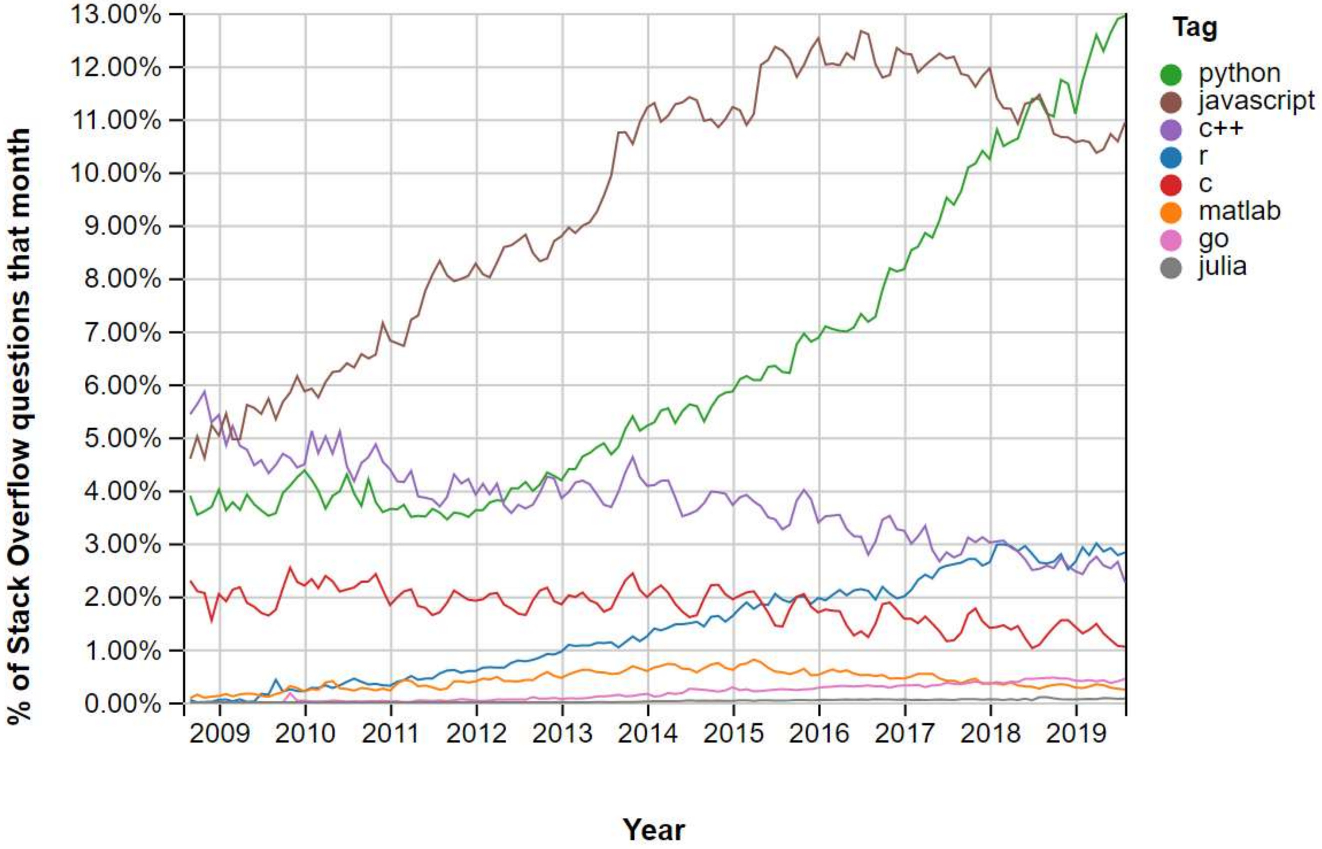
from Codingdojo.com Blog 2/7/2020

How do our usual languages fare?
Worldwide jobs on indeed.com



"Python: The most versatile of the top programming languages of 2020"

Stack Overflow Trends

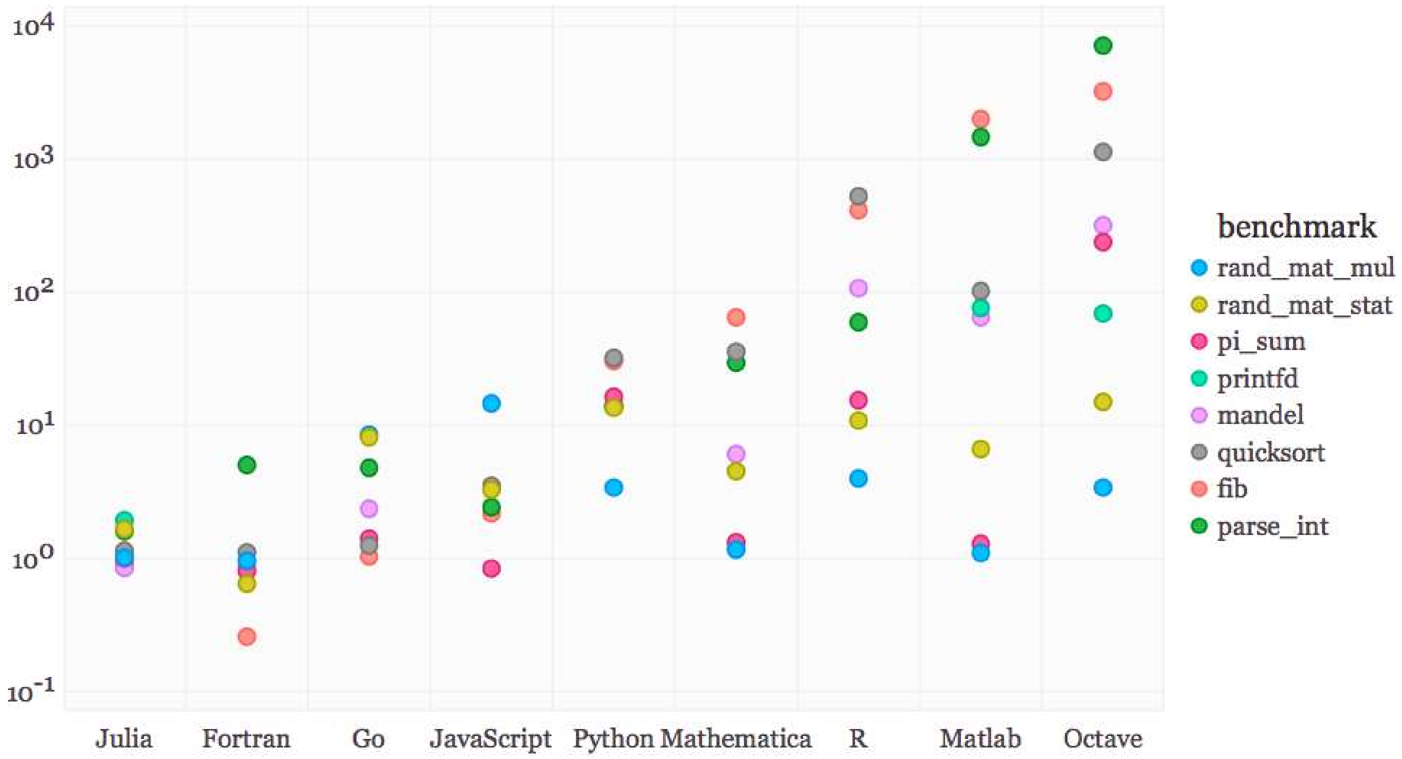


Python is NOT

The fastest run-time

The most memory efficient

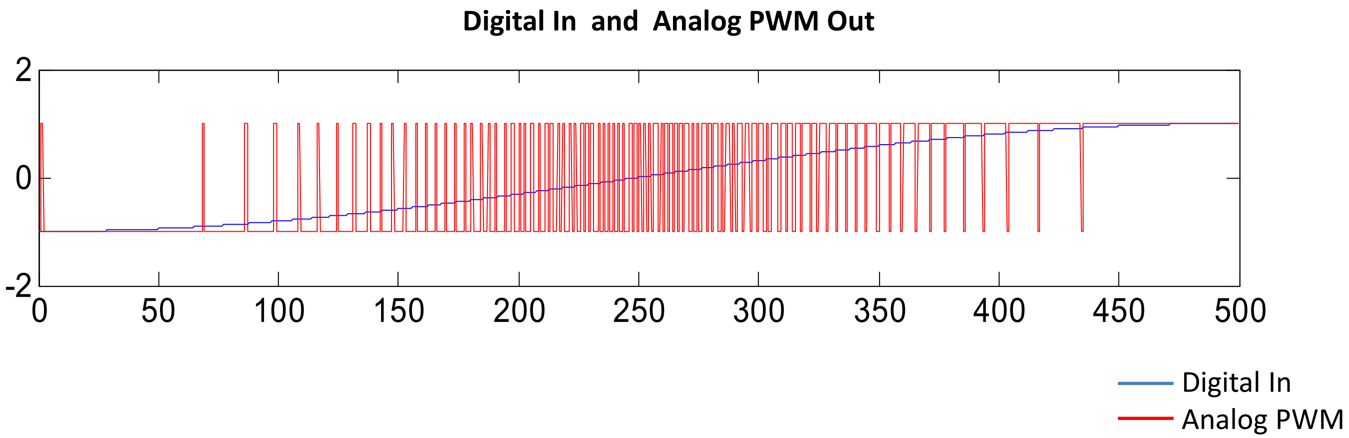
Run Time Comparison

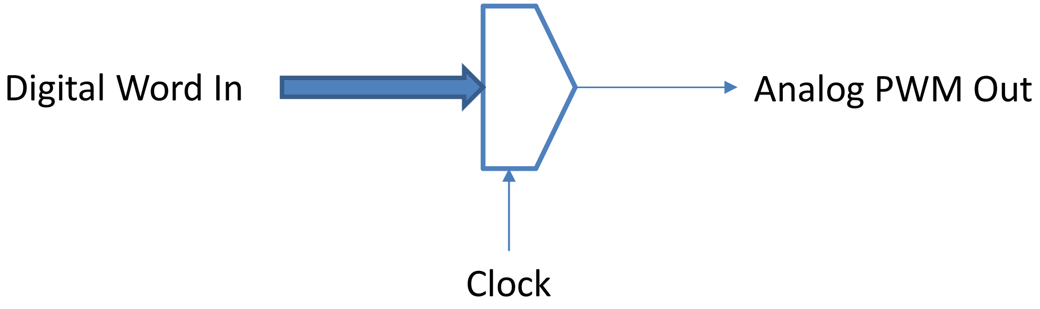


source: https://julialang.org/benchmarks/

Design Examples

Delta Sigma DAC



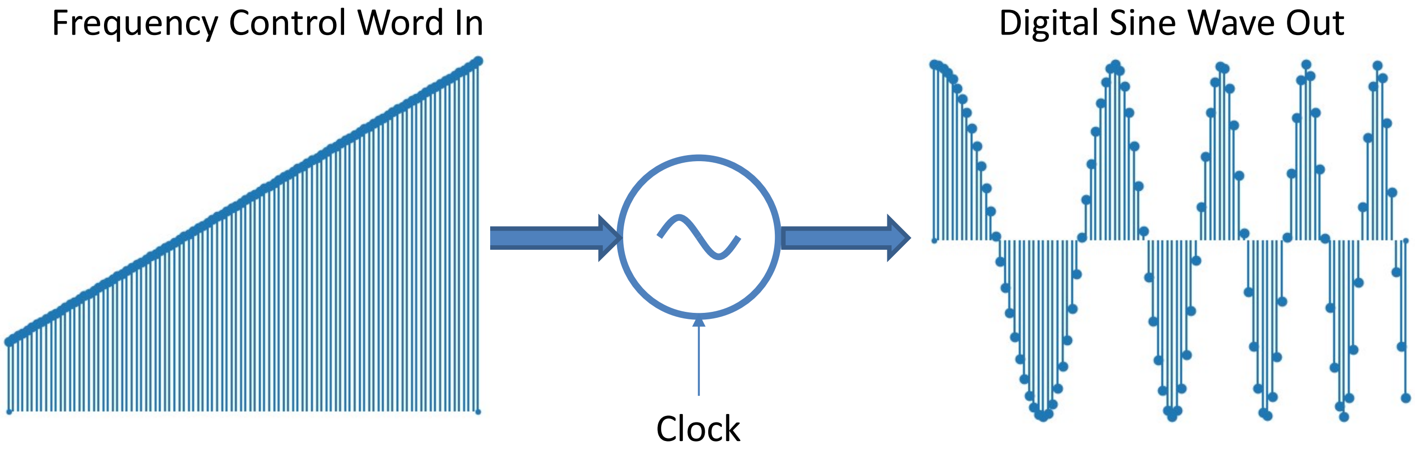


Digital Word In

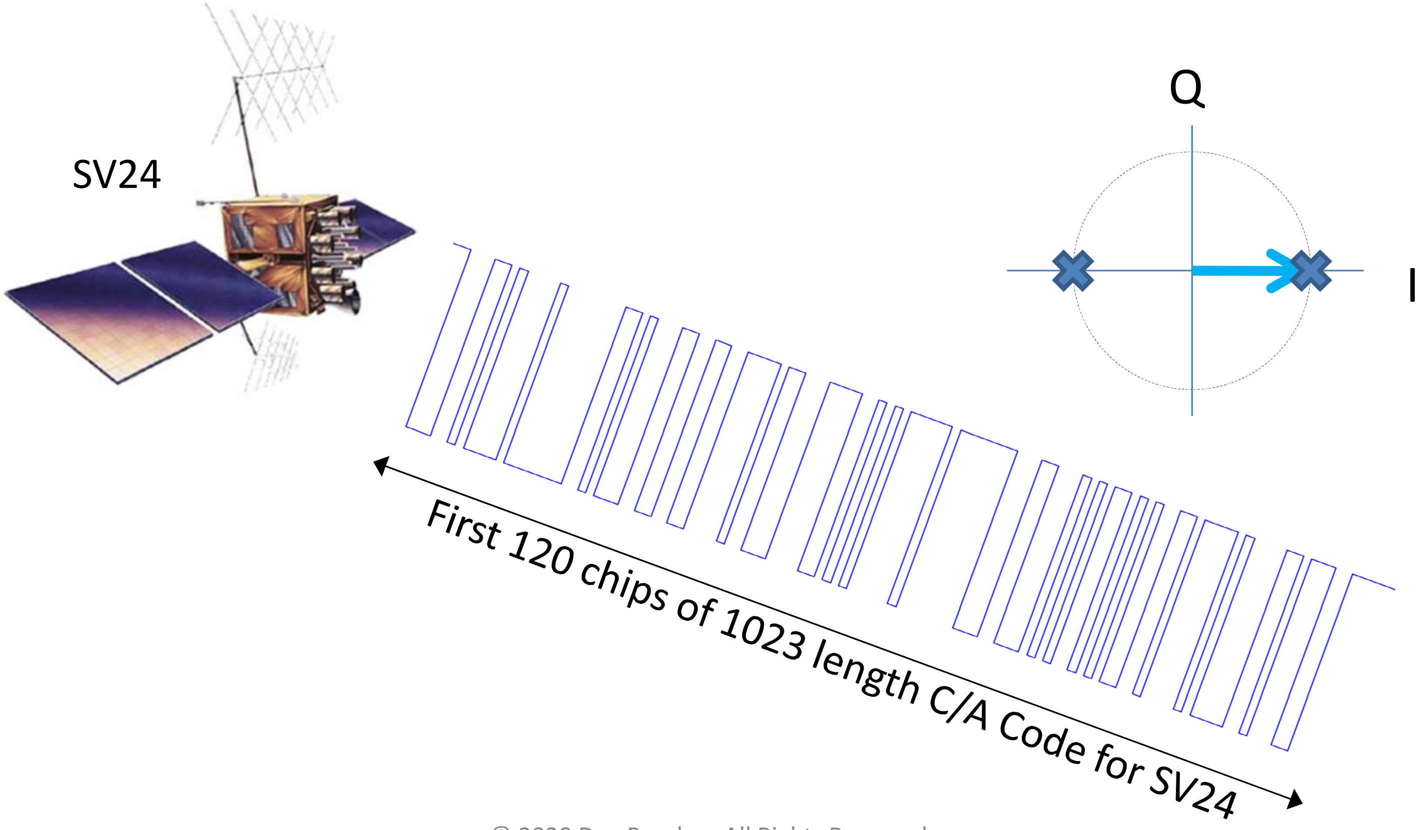
Analog PWM Out

Clock

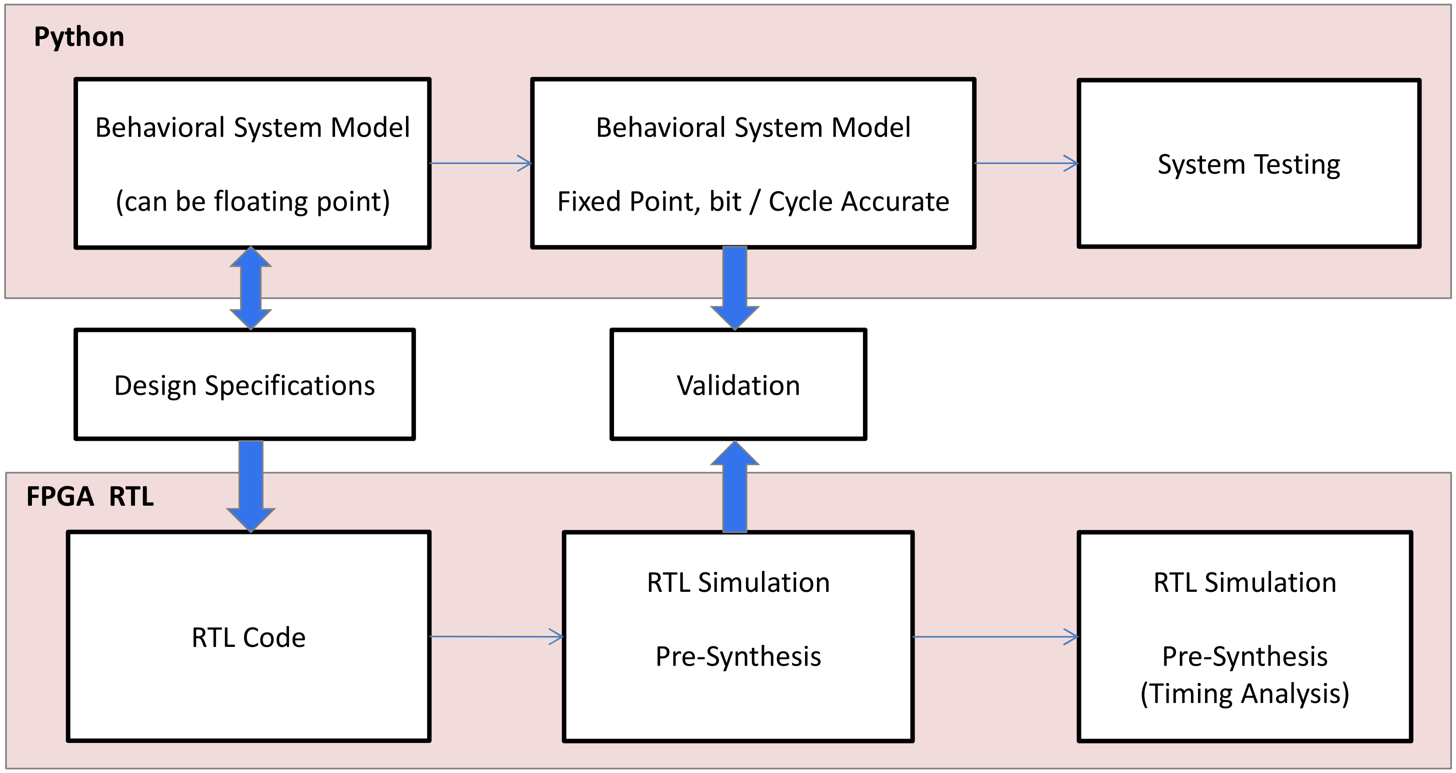
Numerically Controlled Oscillator



GPS C/A Code Generator

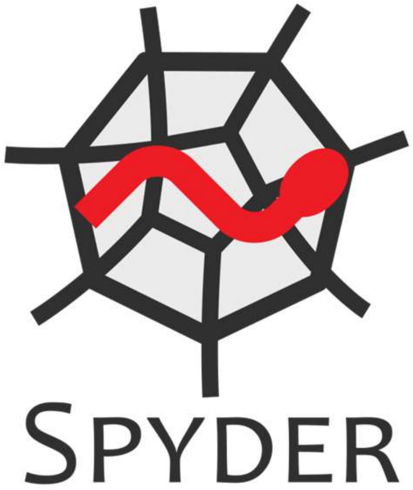


Python for Verification

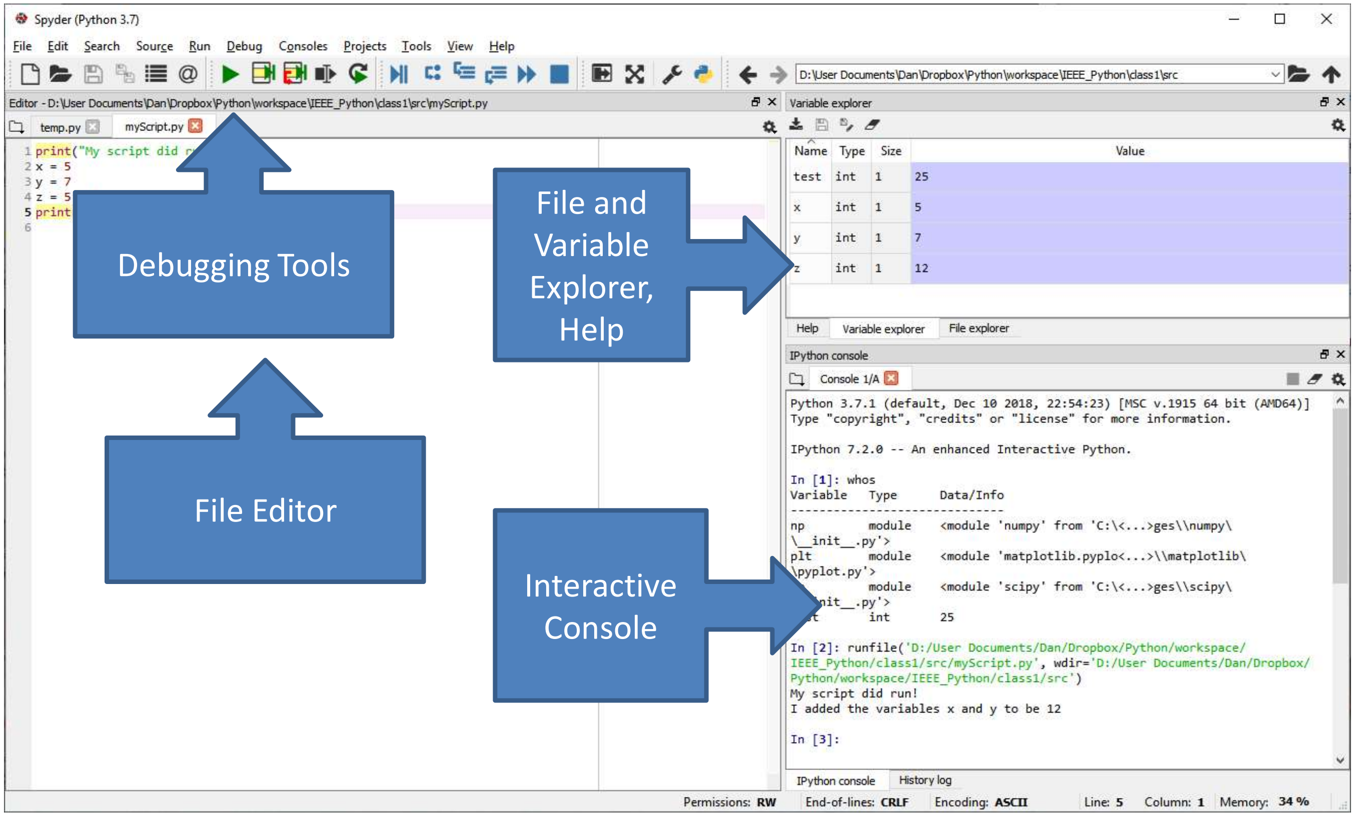


Spyder IDE

Python IDEs



Spyder IDE


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\User Documents\Dan\Dropbox\Python\workspace\IEEE_Python\class1\src
Editor - D:\User Documents\Dan\Dropbox\Python\workspace\IEEE_Python\class1\src\myScript.py
temp.py
myScript.py
Variable explorer
Help
Variable explorer
File explorer
IPython console
Console 1/A
IPython console
History log
Permissions: RW
End-of-lines: CRLF
Encoding: ASCII
Line: 5
Column: 1
Memory: 34 %

Debugging Tools

File and Variable Explorer, Help

File Editor

Interactive Console

```
print("My script did r
x = 5
y = 7
z = 5
print
```

Name	Type	Size	Value
test	int	1	25
x	int	1	5
y	int	1	7
z	int	1	12

```
Python 3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.2.0 -- An enhanced Interactive Python.

In [1]: whos
Variable   Type      Data/Info
------------------------------
np         module    <module 'numpy' from 'C:\<...>ges\\numpy\
\__init__.py'>
plt        module    <module 'matplotlib.pyplo<...>\\matplotlib\
\pyplot.py'>
           module    <module 'scipy' from 'C:\<...>ges\\scipy\
  nit__.py'>
  t        int       25

In [2]: runfile('D:/User Documents/Dan/Dropbox/Python/workspace/
IEEE_Python/class1/src/myScript.py', wdir='D:/User Documents/Dan/Dropbox/
Python/workspace/IEEE_Python/class1/src')
My script did run!
I added the variables x and y to be 12

In [3]:
```

Projects

Setting up a "Project" is completely optional

Project will save all associated IDE configuration settings

A .spyproject directory is created in the root folder of the project with .ini files of all related configuration settings.

Project Explorer allows for navigating the project
(same view as file explorer but limited to project root and lower)

Other tips

Highlight any item and press control-I to get help

In editor, can run complete code ▶
or run highlighted code or cells

In console: enter "clear" or "cls" to clear screen (does not clear variables)
%reset clears entire namespace (and does not reload startup file)

Alt-Up and Alt-Down to move line or selected multiple lines

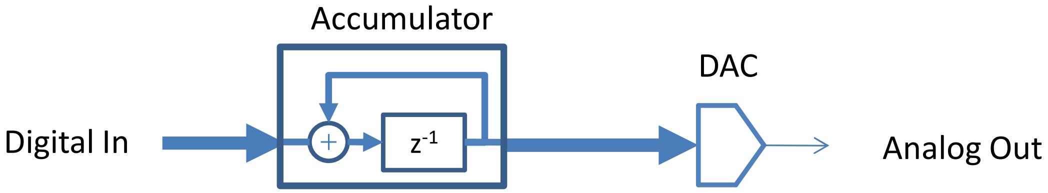
Divide code into cells using `#  %%   optional section name`
View cell sections by section name using View – Panes -Outline

More Help on Spyder

Primary Docs: https://docs.spyder-ide.org

First Coding Example

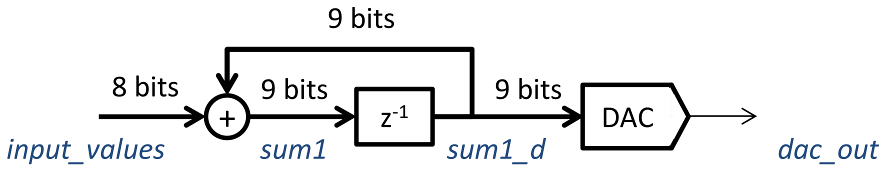
Accumulator



Accumulator Implementation

All *values* are represented as signed integers from -2^{b-1} to $+2^{b-1}-1$

values wrap on overflow



IPython

IPython

Command shell for interactive computing

Combines introspection, tab completion, history, rich media

Adds "magics"

Console for Spyder IDE

Python Kernel for Jupyter Notebook

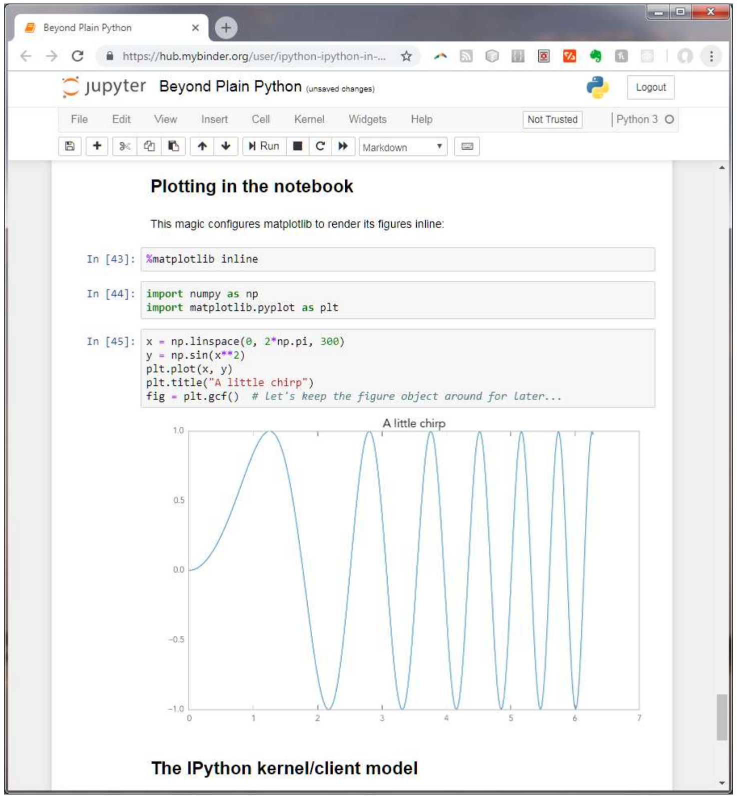
Ipython can also be invoked from terminal by typing "ipython" instead of "python".

More Help on IPython

Primary Docs: https://ipython.readthedocs.io/en/stable/

Jupyter Notebooks

Jupyter Notebooks



Plotting in the notebook

This magic configures matplotlib to render its figures inline:

```
In [43]: %matplotlib inline
```

```
In [44]: import numpy as np
         import matplotlib.pyplot as plt
```

```
In [45]: x = np.linspace(0, 2*np.pi, 300)
         y = np.sin(x**2)
         plt.plot(x, y)
         plt.title("A little chirp")
         fig = plt.gcf()  # Let's keep the figure object around for later...
```

The IPython kernel/client model

Open source, interactive application for creating and sharing documents that combine code, graphics, equations and text.

(Overview using live notebook in class)

BACK-UP SLIDES

Debugging with Jupyter & Spyder

Can connect both to same Kernel:

In Spyder Ipython Console - Gear - Connect to an Existing Kernel

Keep other console open- need to use for integrated debug

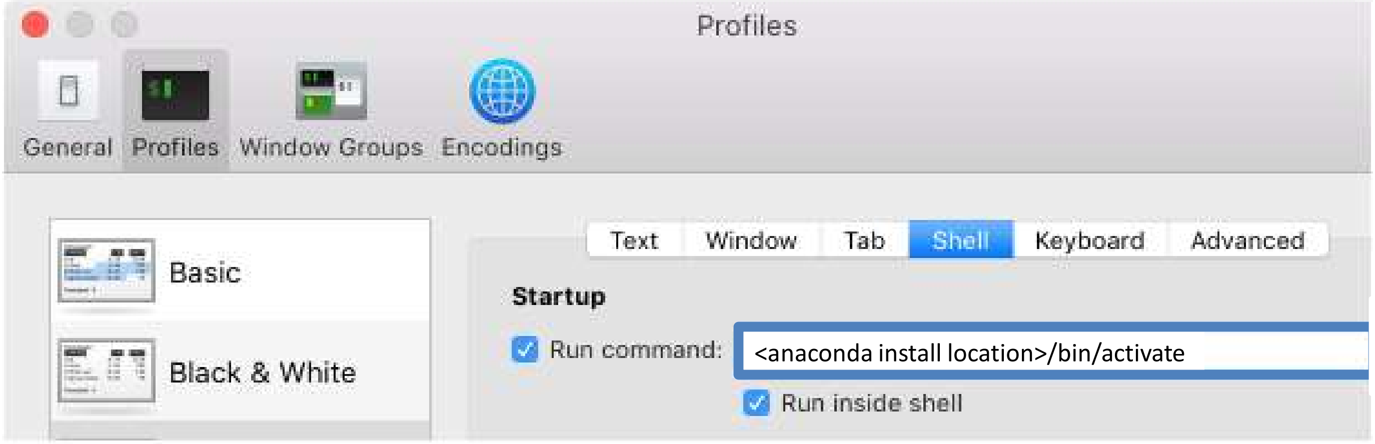
Reload Magic in Ipython (to force the reloading of a module after changes):

```
%load_ext autoreload     (if extension not yet loaded)
%autoreload
%autoreload 0            to disable
```

See https://ipython.org/ipython-doc/3/config/extensions/autoreload.html

Anaconda Prompt for Mac

1. Open **Terminal** → **Preferences** → **Profiles**.
2. Select the profile you use, then switch to the **Shell** tab.
3. Under **Startup**, enter the command you wish to run as the **Run command**.

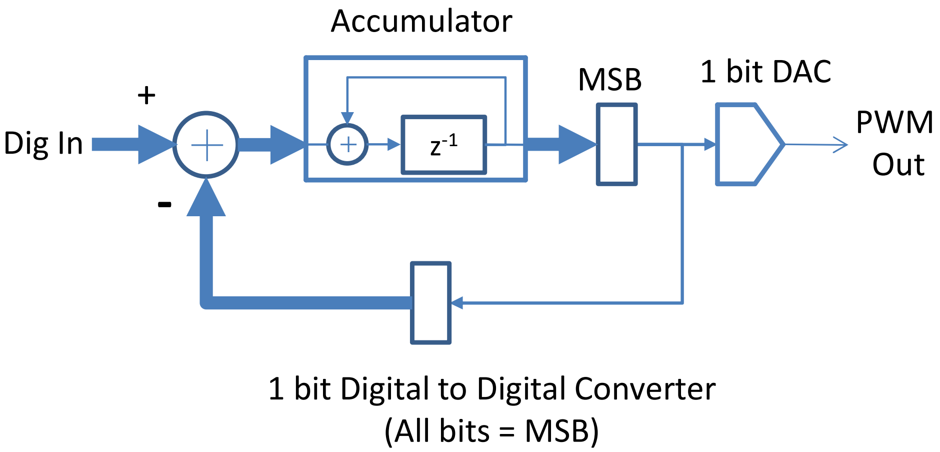


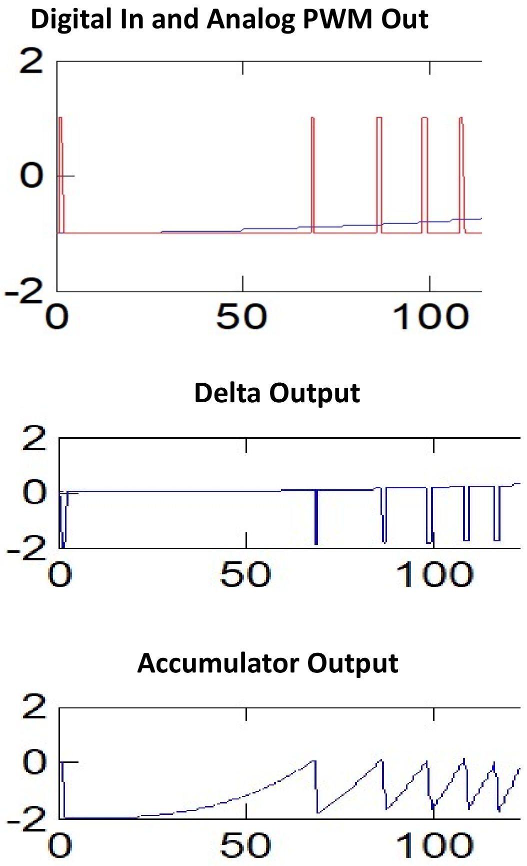
Startup

☑ Run command: <anaconda install location>/bin/activate

☑ Run inside shell

Delta Sigma Waveforms

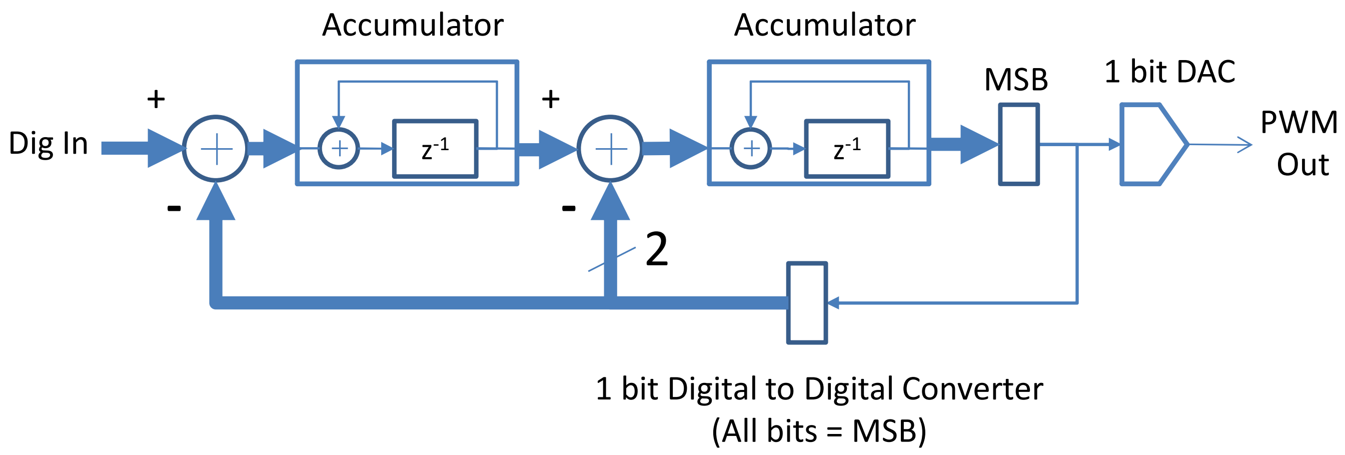



Digital In and Analog PWM Out

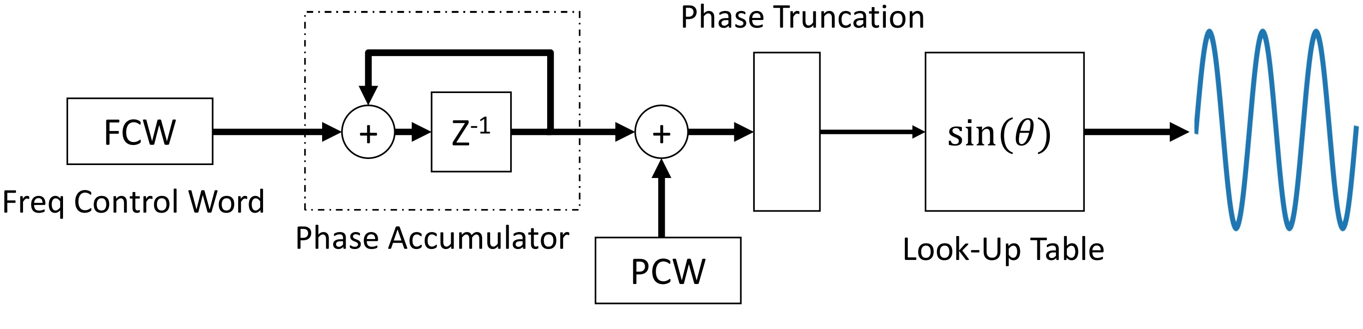
Delta Output

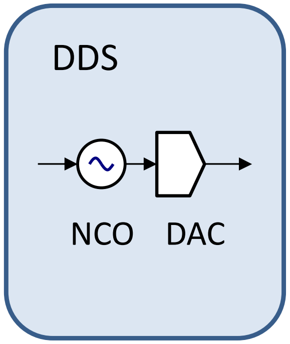
Accumulator Output

2nd Order Delta Sigma DAC



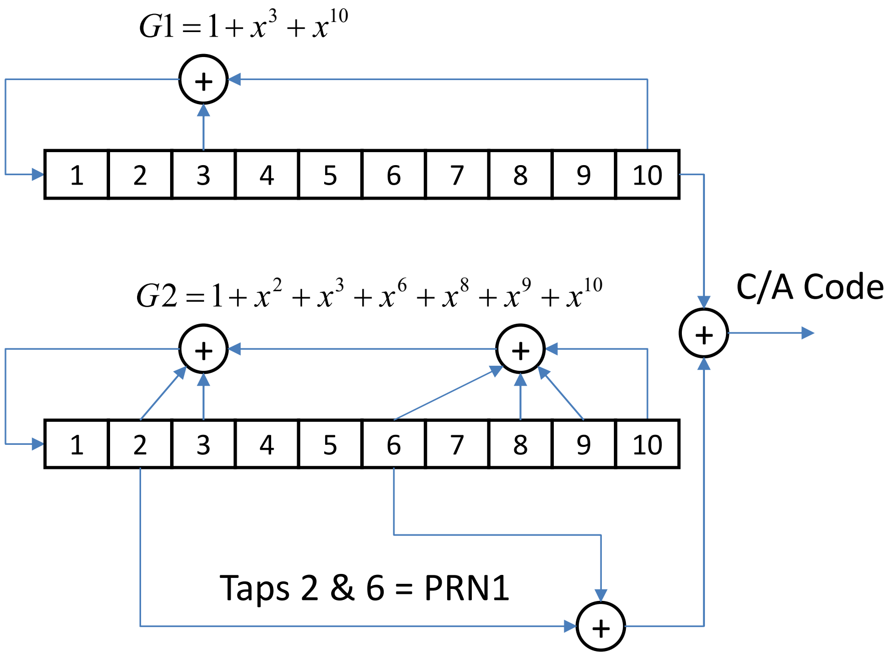
Numerically Controlled Oscillator





GPS C/A Code Generator

PRN ID	G2 Taps	PRN ID	G2 Taps
1	2 & 6	17	1 & 4
2	3 & 7	18	2 & 5
3	4 & 8	19	3 & 6
4	5 & 9	20	4 & 7
5	1 & 9	21	5 & 8
6	2 & 10	22	6 & 9
7	1 & 8	23	1 & 3
8	2 & 9	24	4 & 6
9	3 & 10	25	5 & 7
10	2 & 3	26	6 & 8
11	3 & 4	27	7 & 9
12	5 & 6	28	8 & 10
13	6 & 7	29	1 & 6
14	7 & 8	30	2 & 7
15	8 & 9	31	3 & 8
16	9 & 10	32	4 & 9



A different C/A code is generated by selecting different taps off of G2, which results in delaying the G2 code relative to G1

Reference: "GPS SPS Signal Specification"
https://www.gps.gov/technical/ps/1995-SPS-signal-specification.pdf