Template Examples - Do Not Modify

Boaty McBoatface

Compiled on: November 17, 2022

Table of Contents

CHAPTER 1

Chapter With a Listing

Listing 1.1: Overly Complicated RGB type specification in F#

```
let bounded e =
  let max = 255
  match e < 0 , e > max with
  | true, false -> 0
  | false, true -> max
  | _, _ -> e
let divide e1 e2 =
  match e2 = 0 with
  | true -> 0
  | false -> e1 / e2
type RGB =
  struct
    val R: int
    val G: int
    val B: int
    new(r:int, g:int, b:int) = {R = bounded r; G = bounded g; B = bounded b}
    static member (+) (e1:RGB, e2:RGB) = RGB((e1.R + e2.R), (e1.G + e2.G), (e1.B + e2.B))
    static member Succ (e1:RGB) = RGB((e1.R + 1), (e1.G + 1), (e1.B + 1))
    static member (-) (e1:RGB, e2:RGB) = RGB((e1.R - e2.R), (e1.G - e2.G), (e1.B - e2.B))
    static member (*) (e1:RGB, e2:RGB) = RGB((e1.R * e2.R), (e1.G * e2.G), (e1.B * e2.B))
    static member (/) (e1:RGB, e2:RGB) = RGB((divide e1.R e2.R), (divide e1.G e2.G), (
        divide e1.B e2.B))
  end
```

CHAPTER 2

1970-01-01

2.1 Old Business

2.2 Progress

2.3 Agreements

2.4 Next Steps

CHAPTER 3

Including Code with Minted

my-pagebreak.lua

```
-- insert page break before every level 1 header
-- intended use for tex input, docx output

function Pandoc(doc)
  local hblocks = {}
  for i,el in pairs(doc.blocks) do
      if (el.t == "Header" and el.level == 1) then
         pandoc.List.insert(hblocks,
            pandoc.RawBlock(
              'openxml',
              '<w:p><w:r><w:br w:type="page"/></w:r></w:p>'
            ))
      end
      pandoc.List.insert(hblocks, el)
  end
  return pandoc.Pandoc(hblocks, doc.meta)
end
```

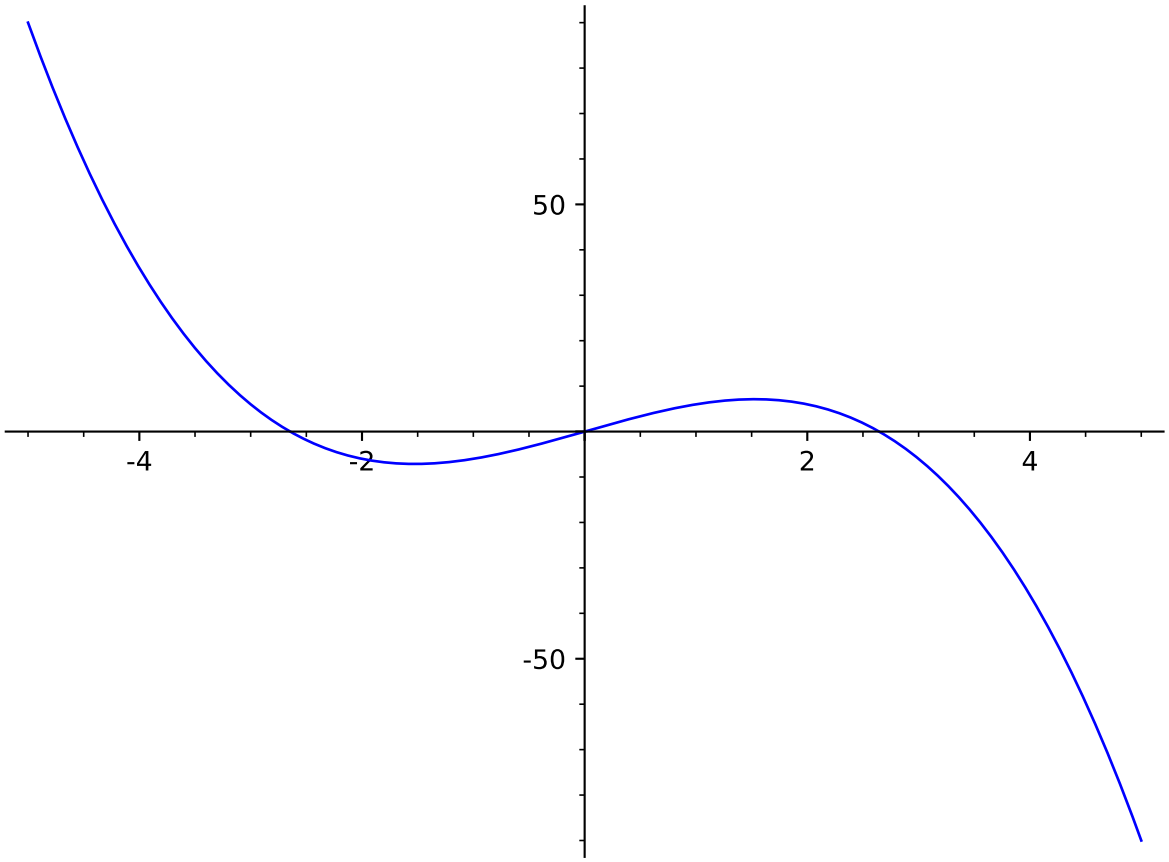
automoderator.yaml

```
    ---

    # Remove Affiliate Links
    # Automatically removes Amazon affiliate links.
    standard: amazon affiliate links
    action: remove
    action_reason: 'Amazon affiliate link'
```

CHAPTER 4

Sageplot Example

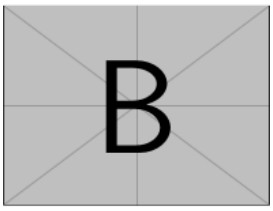


CHAPTER 5

Chapter with a Subcaption

Subcaption example

(a) caption a



(b) caption b

Figure 5.1: a shows Figure 1 and b shows Figure 2.

CHAPTER 6

Timeline Example

Tables aren't so confusing when you align the text!

Table 6.1: Summary of Data Sources Investigated

Name	Source	Climate Station	Type	10 Minute	Hourly	Daily
CliFlo10min	CliFlo	Albany	2015 Observations	Yes	Yes	Yes
CliFloHourly	CliFlo	Albany	2015 Observations	No	Yes	Yes
CliFloDaily	CliFlo	Albany	2015 Observations	No	No	Yes
NIWA12x24	SolarView	Takapuna	Idealised Year	No	Yes	Yes
NIWA8760	SolarView	Takapuna	Idealised Year	No	Yes	Yes

Table 6.2: Daily Cumulative Solar Irradiation Statistics for September and December (Wh/m2)

	September			December		
Min	780.6		678	1872		1726
Q. 1	1697.2		1756	5739		5148
Med	2136.1		2068	6592		6848
Mean	2111.3	2770	2130	6447	5213	6313
Q. 3	2519.5		2612	7545		8170
Max	3261.1		3414	8889		8901
SD	715.5		698.4	1687.7		2204.6
Name	CliFloHourly	NIWA12x24	NIWA8760	CliFloHourly	NIWA12x24	NIWA8760

Timeline Example

2021	ResBaz 2021 takes place
1984	LaTeX Initial Release
	LaTeX was originally written in the early 1980s by Leslie Lamport at SRI International

CHAPTER 7

Markdown Example

Here is a simple footnote[1].
A footnote can also have multiple lines[2].
You can also use words, to fit your writing style more closely[3].

[1] My reference.

[2] Every new line should be prefixed with 2 spaces.
This allows you to have a footnote with multiple lines.

[3] Named footnotes will still render with numbers instead of the text but allow easier identification and linking.
This footnote also has been made with a different syntax using 4 spaces for new lines.

Appendices

A section in my appendices!

/Users/eper363/Projects/Poolean/Poolean/HuffmanDomain.fs

```
module Poolean.HuffmanDomain

open NucleotideDomain
open CompressionDomain

module HuffmanDomain =

    type ASCIICount = { Key: char; Value: int} // could change for n-grams (string) and frequency (float)

    /// 1's based Heap implementation (Priority Queue)
    /// - Smallest (integer) items have the highest priority
    ///
    /// See Kleinberg & Tardos for more details (2nd edition, p. 60)
    /// - This implementation is not thread-safe
    type Heap() =
        let mutable size = 0
        let mutable queue = [| {Key = '\000'; Value = 0} |] // first index is a placeholder, don't use it

        let swap e1 e2 =
            if (queue.[e2]).Value < (queue.[e1]).Value then
                let tmp = queue.[e1]
                queue.[e1] <- queue.[e2]
                queue.[e2] <- tmp

        member self.Print() = printfn "Queue:\t%A\nSize:\t%d" queue size

        /// find parent index of node i
        member self.Parent(i) = floor((i |> float) / 2.0) |> int

        /// bubble values up the heap
        member self.HeapifyUp(i) =
            if i > 1 then
                let j = self.Parent(i)
                swap j i
                self.HeapifyUp(j) // more efficient if I checked i < j, this will go up the tree with NOPs

        /// bubble values down the heap
        member self.HeapifyDown(i) =
            let n = size
            match (2*i > n), (2 * i < n), (2 * i = n) with
            | _, true, _ ->
                let left = 2*i
                let right = 2*i + 1
                (match (queue.[left]).Value < (queue.[right]).Value with | true -> left | false -> right) |> Some
            | _, _, true -> 2*i |> Some
            | _, _, _ -> None
            |> Option.bind (fun j -> swap i j; self.HeapifyDown(j); Some j) |> ignore

        /// insert value into heap H
        /// use heapify-up to repair damaged heap structure after each call
        /// new elements get appended to the end of the internal array
        member self.Insert(v) =
            queue <- Array.append queue [|v|]
            size <- size + 1
            self.HeapifyUp(size)

        /// if heap contains elements, return minimum element
        member self.FindMin() = match size >= 1 with | true -> Some queue.[1] | false -> None

        /// Delete element in heap position i
        /// use heapify-down to repair damaged heap structure after each call
        member self.Delete(i) =
            queue <- Array.append queue.[0..i - 1]  queue.[i+1 .. size]
            size <- size - 1
            self.HeapifyDown(i)

        /// identify and delete element with minimum key value
        member self.ExtractMin() = self.FindMin() |> Option.bind (fun min -> self.Delete(1); Some min)
```

Page 1 of 1

Figure 7.1: A priority queue is required for the Huffman coding algorithm. This is one possible implementation, from Kleinberg and Tardos, 2006.

Bibliography

Kleinberg, J., & Tardos, É. (2006). Algorithm design. Pearson.