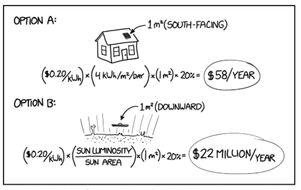
In [1]:
```
use Physics::Unit;
use Physics::Measure :ALL;
```

In [2]:
```
# reset rounding
$Physics::Measure::round-val = Nil;
```



SOLAR ENERGY TIP: TO MAXIMIZE SUN EXPOSURE, ALWAYS ORIENT YOUR PANELS DOWNWARD AND INSTALL THEM ON THE SURFACE OF THE SUN.

Option A - on the roof

To break this down into pieces, we have:

In [3]:
```
#( 0.20 $/kWh ) * ( ( 4kWh / m2 ) per day ) * ( 1 m2 ) * 20%
#      -----          ---------    -------     ----    ---
#        [1]             [2]        [3]        [4]     [5]
```

Out[3]:
```
()
```

In [4]:
```
#[1] Dollars per kWh

# First we need to create a custom Unit and a custom Measure type
# we do not have currency units (yet) so I will ignore the USD currency in the definition

Unit.new(defn => '1 / kWh', names => ['$/kWh']);           #dd GetUnit('$/kWh');

# And a new custom measurement type...
class DollarPerKWH is Measure {
    has $.units where *.name eq ('$/kWh').any;
}

# And then link them with Get Measurement Unit...
GetMeaUnit('$/kWh').NewType('DollarPerKWH');

my $cost-per-kwh = 0.2 $/kWh';
```

Out[4]:
```
0.2$/kWh
```

In [5]:
```
#[2] Insolation

my $energy-per-area = (4 kWh/m2');
```

Out[5]:
```
4kWh/m2
```

In [6]:
```
$energy-per-area.^name;
```

Out[6]:
```
Physics::Measure::Insolation
```

In [7]:
```
#[3] Irradiance

my $power-per-area-on-roof = $energy-per-area / 1 day';
```

Out[7]:
```
0.000046kW/m^2
```

In [8]:
```
$power-per-area-on-roof.units.type;
```

Out[8]:
```
Irradiance
```

In [9]:
```
#[4] we can just do 1 m2 like this...

say 1 m2';
```

```
1m^2
```

In [10]:
```
#[5] and a percent % like this...

say 20 %';    #libra format '' always needs a space between the number and the unit string
```

```
20%
```

Putting it all together:

In [12]:
```
my $earnings-per-year-on-roof = $cost-per-kwh * ( $power-per-area-on-roof * 1 m2' ) * 20 %' * 1 year';
```

Out[12]:
```
58.44①
```

the ① indicates that the result is dimensionless

Option B - on the sun

To break this one down into pieces, we have:

In [13]:
```
#( 0.20 $/kWh ) * ( ( Sun Luminosity ) / ( Sun Area ) ) * ( 1 m2 ) * 20%
#      -----          --------------       --------        ----    ---
#        [1]              [2]               [3]           [4]     [5]
```

Out[13]:
```
()
```

In [14]:
```
# set rounding to cope with big values
$Physics::Measure::round-val = 100;
```

Out[14]:
```
100
```

In [15]:
```
#[2] we can use the postfix style for SI units

my $solar-luminosity = 3.828e26W;
```

Out[15]:
```
382800000000000026004684800W
```

In [16]:
```
$solar-luminosity.norm;     # and we can auto normalize SI units to the best SI prefix
```

Out[16]:
```
400YW
```

In [19]:
```
#[3] and the area (also from https://en.wikipedia.org/wiki/Sun)

my $solar-area = 6.09e12 km2';
$solar-area.in: <m2>
```

Out[19]:
```
6090000000000000000m^2
```

In [20]:
```
$solar-area.in: <peta m2>;
```

Out[20]:
```
6100peta m2
```

In [22]:
```
#combining [2] and [3] gives:

my $power-per-area-on-sun = $solar-luminosity / $solar-area;
```

Out[22]:
```
62857100W/m^2
```

In [23]:
```
$power-per-area-on-sun.in: <kW/m^2>;
```

Out[23]:
```
62900kW/m^2
```

Putting it all together:

In [25]:
```
my $earnings-per-year-on-sun = $cost-per-kwh * ( $power-per-area-on-sun * 1 m2' ) * 20 %' * 1 year';
```

Out[25]:
```
79344822857100①
```

In [26]:
```
($earnings-per-year-on-sun.value / 1_000_000_000_000).fmt("%d billion USD per year");
```

Out[26]:
```
79 billion USD per year
```

Conclusion (I)

The sharp eyed reader will note that I then felt duty bound to double check my result, since this is adrift of the 22 million USD per year in Randall's comic.

In [52]:
```
# To review the calcs, I thought an order of magnitude approach would help:

sub order-of-magnitude($number is copy) {
    return Nil if $number == 0;  # Order of magnitude undefined for 0
    return floor(log10($number).Int);
}
```

Out[52]:
```
&order-of-magnitude
```

In [53]:
```
my $Lsol-oom = order-of-magnitude( +$solar-luminosity );
```

Out[53]:
```
26
```

In [54]:
```
my $Asol-oom = order-of-magnitude( +( $solar-area.in: <m2> ) );
```

Out[54]:
```
18
```

In [55]:
```
my $ppa-sun-oom = order-of-magnitude( +$power-per-area-on-sun );
```

Out[55]:
```
7
```

In [59]:
```
# check if oom calc is close enough

( $ppa-sun-oom - ( $Lsol-oom - $Asol-oom ) ) <= 1;
```

Out[59]:
```
True
```

- since many factors are invariant $cost-per-kwh, 1 m2', 20 %', 1 year' we can set them aside
- the main driver of the result is then the power-per-area

In [58]:
```
# get oom for ppa sun vs ppa roof

my $ppa-roof-oom = order-of-magnitude( +$power-per-area-on-roof );
```

Out[58]:
```
-4
```

In [68]:
```
my $sun-roof-power-ratio = $power-per-area-on-sun / $power-per-area-on-roof;
```

Out[68]:
```
1357714285700①
```

In [69]:
```
my $sun-roof-power-ratio-oom = order-of-magnitude( +$sun-roof-power-ratio );
```

Out[69]:
```
12
```

In [71]:
```
# get oom for epy sun vs ppa roof

my $sun-roof-epy-ratio = $earnings-per-year-on-sun / $$earnings-per-year-on-roof;
```

Out[71]:
```
1357714285700①
```

In [72]:
```
my $sun-roof-epy-ratio-oom = order-of-magnitude( +$sun-roof-epy-ratio );
```

Out[72]:
```
12
```

In [73]:
```
# check if oom calc is close enough

( $sun-roof-epy-ratio-oom - $sun-roof-power-ratio-oom ) <= 1;
```

Out[73]:
```
True
```

- can also visually check "under the hood" , and extract the ratio of value by hand since the units are the same:

In [74]:
```
dd $power-per-area-on-sun;
```

```
Irradiance $power-per-area-on-sun = Physics::Measure::Irradiance.new(value => 62857142.85714286e0, units =>   Unit.new( factor => 1, offset => 0, defn => 'W / m^2', type => Irradiance,
  dims => [0,1,-3,0,0,0,0,0], dmix => ("W"=>1,"m"=>-2).MixHash, names => ['W/m^2'] ));
, error => Error)
```

In [75]:
```
dd $power-per-area-on-roof;
```

```
Irradiance $power-per-area-on-roof = Physics::Measure::Irradiance.new(value => <1/21600>, units =>   Unit.new( factor => 1, offset => 0, defn => 'W / m^2', type => Irradiance,
  dims => [0,1,-3,0,0,0,0,0], dmix => ("W"=>1,"m"=>-2).MixHash, names => ['W/m^2'] ));
, error => Error)
```

In [76]:
```
62857142.85714286e0 / <1/21600>
```

Out[76]:
```
1357714285714.286
```

Conclusion (II)

So, I would say, that the right answer is 79 billion USD per year and that my math beats Randall's.

BUT - I am too conscious that I may have made an error - either in these calcs or in the Physics::Measure code.

In the spirit of check your results and show your workings I think I am partly protected by the grammar school code of Physics marking ... perhaps this only warrants 6/10 if the result is wrong!

So, PLEASE do feel free to check my results and let me know what you think!