```
  public static int region_given_pvt(double p, double v, double t)

  //helpers to easily generate values "randomly", ensuring we're within the range we care about- NOT PHYSICALLY BASED
  //"percent" = "percent between min value and max value across given dimension"
  public static double p_given_percent(   double t) { return Lerpd(p_min,p_max,t); }
  public static double psat_given_percent(double t) { return Lerpd(psat_min,psat_max,t); }
  public static double v_given_percent(   double t) { return Lerpd(v_min,v_max,t); }
  public static double t_given_percent(   double t) { return Lerpd(t_min,t_max,t); }
  public static double u_given_percent(   double t) { return Lerpd(u_min,u_max,t); }
  public static double s_given_percent(   double t) { return Lerpd(s_min,s_max,t); }
  public static double h_given_percent(   double t) { return Lerpd(h_min,h_max,t); }
  public static double q_given_percent(   double t) { return t; } //q already is a percent

  public static double percent_given_p(   double p) { return (p-p_min)/(p_max-p_min); }
  public static double percent_given_psat(double psat) { return (psat-psat_min)/(psat_max-psat_min); }
  public static double percent_given_v(   double v) { return (v-v_min)/(v_max-v_min); }
  public static double percent_given_t(   double t) { return (t-t_min)/(t_max-t_min); }
  public static double percent_given_u(   double u) { return (u-u_min)/(u_max-u_min); }
  public static double percent_given_s(   double s) { return (s-s_min)/(s_max-s_min); }
  public static double percent_given_h(   double h) { return (h-h_min)/(h_max-h_min); }
  public static double percent_given_q(   double q) { return q; } //q already is a percent

  //rule of naming for consistency: prefer lexical ordering "p < v < t < u < s < h < q", ie "p_given_vt" rather than "p_given_tv"

  public static double p_given_vt(double v, double t)
  {
    return IAPWS95.IAPWS95_pressure(1.0/v,t)*1000.0; //expects:Kg/M^3,K returns KPa
  }

  public static double p_given_vu(double v, double u) //CONFIRMED NEEDED!
  {
    return p_given_percent(0.5); //TODO:
  }
```

Use 95: fix v, guess T2 —> u2, iterate. Use v, T2 with 95 —> P2

```
  public static double v_given_pt(double p, double t) //CONFIRMED NEEDED!
  {
    return 1.0/IF97.rhomass_Tp(t,p/1000000.0); //expects:K,MPa returns Kg/M^3
  }
```

This will only work outside the vapor dome

Use 97: rhomass_Tp. Then v = 1/rhomass_Tp

Inside the vapor dome T,P is not enough information to fix the state

```
  public static double t_given_pv(double p, double v) //CONFIRMED NEEDED!
  {
    return t_given_percent(0.5); //TODO:
  }
```

If in the vapor dome, use T=Tsat97(p)

If outside the vapor dome: 1) Use 95: guess T2 —>P2, iterate

2) use 97: guess T2 —> v2 from rhomass_Tp, iterate

```
  public static double t_given_pu(double p, double u) //CONFIRMED NEEDED!
  {
    return t_given_percent(0.5); //TODO:
  }
```

If in the vapor dome, use T=Tsat97(p)

If outside the vapor dome: Use 97: guess T2, umass_Tp —> u2, iterate

```
  public static double tsat_given_p(double p)
  {
    return IF97.Tsat97(p/1000000.0);
  }
```

```
public static double vliq_given_p(double p)
{
 return 1.0/IF97.rholiq_p(p/1000000.0); //expects:MPa returns Kg/M^3
}

public static double vvap_given_p(double p)
{
 return 1.0/IF97.rhovap_p(p/1000000.0); //expects:MPa returns Kg/M^3
}

public static double v_given_pu(double p, double u) //CONFIRMED NEEDED!
{
 return v_given_percent(0.5); //TODO:
}
```

If outside the vapor dome: Use 97: P2, guess T2 —> u2, iterate. Use rhomass(T2,p2) —> v2

If in vapor dome: use Q_pumass —>Q2, Use v_pQ

```
public static double u_given_pt(double p, double t) //CONFIRMED NEEDED!
{
 return u_given_percent(0.5); //TODO:
}
```

If outside vapor dome Use 97: umass(T,p)

If inside vapor dome, T,p not enough information to fix the state

```
public static double s_given_pt(double p, double t)
{
 return s_given_percent(0.5); //TODO:
}
```

Use 97: But this will only work outside the vapor dome

```
public static double s_given_pu(double p, double u) //CONFIRMED NEEDED!
{
 return s_given_percent(0.5); //TODO:
}
```

If inside vapor dome: use 97: Q_pumass —> Q2, then use smass_pQ —> s2

If outside vapor dome: use 97: umass(Tguess, p) —> u2 iterate then use T2, p2 with smass(T,p) —> s2

```
public static double h_given_pt(double p, double t)
{
 return h_given_percent(0.5); //TODO:
}
```

Use 97: But this will only work outside the vapor dome

```
public static double h_given_pu(double p, double u) //CONFIRMED NEEDED!
{
 return h_given_percent(0.5); //TODO:
}
```

If inside vapor dome: use 97: Q_pumass —> Q2, then use hmass_pQ —> h2

If outside vapor dome: use 97: umass(Tguess, p) —> u2 iterate then use T2, p2 with hmass(T,p) —> h2

```
public static double q_given_pt(double p, double t)
{
 return q_given_percent(0.5); //TODO:
}
```

This will not work - T, p are not enough information to set Q