In [5]:

```
import numpy
import scipy
import matplotlib
import pandas
import statsmodels
import seaborn
import sklearn

print("numpy:", numpy.__version__)
print("scipy:", scipy.__version__)
print("matplotlib:", matplotlib.__version__)
print("statsmodels:", statsmodels.__version__)
print("pandas:", pandas.__version__)
print("seaborn:", seaborn.__version__)
print("sklearn:", sklearn.__version__)
```

```
numpy: 1.23.5
scipy: 1.10.0
matplotlib: 3.7.0
statsmodels: 0.13.5
pandas: 1.5.3
seaborn: 0.12.2
sklearn: 1.2.1
```

In [11]:

```
5*6
2+2
```

Out[11]:

```
4
```

In [9]:

```
print(2+2)
2+3
6*7
```

```
4
```

Out[9]:

```
42
```

In [13]:

```
annual = 50000
```

In [14]:

```
annual/(40*52)
```

Out[14]:

```
24.03846153846154
```

In [16]:

```
def hourly(annualwage):
    hourlywage = annualwage/(40*52)


    return hourlywage
```

In [17]:

```
hourly(60000)
```

Out[17]:

```
28.846153846153847
```

In [18]:

```
my_hourly_wage = hourly(17000)
```

In [19]:

```
my_hourly_wage
```

Out[19]:
```
8.173076923076923
```

In [20]:
```
list(range(10))
```

Out[20]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [21]:
```
my_annual_salaries = [0,1,2,3,4,5,6,7,8,9]
```

In [22]:
```
hourly_wages = []
for salary in my_annual_salaries:
    hourly_wages.append( hourly(salary) )
hourly_wages
```

Out[22]:
```
[0.0,
 0.0004807692307692308,
 0.0009615384615384616,
 0.0014423076923076924,
 0.0019230769230769232,
 0.002403846153846154,
 0.0028846153846153848,
 0.0033653846153846156,
 0.0038461538461538464,
 0.004326923076923077]
```

In [23]:
```
annual/(40*52)
```

Out[23]:
```
24.03846153846154
```

In [24]:
```
def is_roth_eligible(annual_wage):
    return annual_wage < 130000
```

In [25]:
```
def is_roth_eligible(annual_wage):
    if annual_wage < 130000:
        return True
    else:
        return False
```

In [26]:
```
200000 < 130000
```

Out[26]:
```
False
```

In [27]:
```
is_roth_eligible(100000)
```

Out[27]:
```
True
```

In [28]:
```
is_roth_eligible(200000)
```

Out[28]:
```
False
```

In [29]:
```
hourly(my_annual_salaries)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[29], line 1
----> 1 hourly(my_annual_salaries)

Cell In[16], line 2, in hourly(annualwage)
      1 def hourly(annualwage):
----> 2     hourlywage = annualwage/(40*52)
      5     return hourlywage

TypeError: unsupported operand type(s) for /: 'list' and 'int'
```

In [30]:
```
hourly_wages = numpy.array([1,2,3])
```

In [31]:
```
hourly_wages/60
```

Out[31]:
```
array([0.01666667, 0.03333333, 0.05      ])
```

In [32]:
```
numpy.abs([-1,2,3])
```

Out[32]:
```
array([1, 2, 3])
```

In [33]:
```
numpy.abs(numpy.array([-1,2,3]))
```

Out[33]:
```
array([1, 2, 3])
```

In [34]:
```
numpy.sum([-1,2,3])
```

Out[34]:
```
4
```

In [35]:
```
for i in range(1000):
    print(i)
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
```

```
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
```

```
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
```

```
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
```

```
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
```

```
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
```

```
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
```

```
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
```

```
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

```
In [ ]:
```