```
1
2  # Family name: xxxx
3  # Student number:  123456
4  # Course: IT1 1120E
5  # Assignment Number 1
6
7  import turtle
8  import math
9
10 ##############################################################
   #########
11 # Question 1
12 ##############################################################
   ##########
13
14 def pythagorean_pair(a,b):
15     ''' (int,int)->bool
16     Returns True if a and b are pythagorean pair and False otherwise'''
17
18     c=round((a**2+b**2)**0.5) # accept int() too. do not worry about numerical errors
19     return a**2+b**2 == c**2
20
21 ##############################################################
   ##########
22 # Question 2
23 ##############################################################
   ##########
24
25 def mh2kh(s):
26     ''' (number)->number
27     Returns the speed s expressed in km/h'''
28
29     return s*1.609344
30
31
32 ##############################################################
   ##########
33 # Question 3
34 ##############################################################
   ##########
35
36 def in_out(xs,ys,side):
37     ''' (number,number,number)->None
38     Preconditions: side is non-negative
39     Prints True if the user inputted query point is inside of the input square,
40           and False otherwise'''
41
42     x=float(input("Enter a number for the x coordinate of a query point: "))
43     y=float(input("Enter a number for the y coordinate of a query point: "))
44
45     print( (x>=xs) and (x<=xs+side) and (y >= ys) and (y<=ys+side) )
46
47 ##############################################################
   ##########
```

```
48  # Question 4
49  ##############################################################
    ##########
50
51  def safe(n):
52      ''' (int)->bool
53      Precondition: n is a non-negative integer
54      Returns True if n is safe and False otherwise.'''
55
56      tens_digit = n // 10
57      ones_digit = n % 10
58      return not(tens_digit == 9 or ones_digit == 9 or n % 9 == 0)
59
60
61  ##############################################################
    #########
62  # Question 5
63  ##############################################################
    ##########
64  def quote_maker(quote, name, year):
65      '''(str, str,int)->str'''
66      return "In "+str(year)+", a person called "+name+" said: "+"\""+quote+"\""
67
68  ##############################################################
    #########
69  # Question 6
70  ##############################################################
    ##########
71  def quote_displayer():
72      '''(None)->None
73      Displays quote nicely'''
74      quote=input("Give me a quote: ")
75      name=input("Who said that? ")
76      year=int(input("What year did she/he say that? "))
77      print(quote_maker(quote, name, year))
78
79
80
81  ##############################################################
    ##########
82  # Question 7
83  ##############################################################
    ##########
84
85  def rps_winner():
86      ''' (None)->None
87      Prints results for Player 1 in RPS game'''
88      choice1=input('What choice did player 1 make? \nType one of the following options:
    rock, paper, scissors: ')
89      choice2=input('What choice did player 2 make? \nType one of the following options:
    rock, paper, scissors: ')
90      print("Player 1 wins. That is", choice1=='rock' and choice2=='scissors' or
    choice1=='scissors' and choice2=='paper' or choice1=='paper' and  choice2=='rock')
91      print("It is a tie. That is not", choice1!=choice2)
```

```
92
93
94  ##############################################################
    ##########
95  # Question 8
96  ##############################################################
    ##########
97
98  def fun(x):
99      ''' (number)->number
100     Precondition: x is positive
101     Returns solution for y of the equation 10**(4y)=x+3'''
102
103     return math.log10(x+3) / 4
104
105 ##############################################################
    ##########
106 # Question 9
107 ##############################################################
    ##########
108
109 def ascii_name_plaque(name):
110     ''' (str)->None
111     Draws/Prints name plaque'''
112     print(5*"*"+len(name)*"*"+5*"*")
113     print("*"+4*" "+len(name)*" "+4*" "+"*")
114     print("*  "+2*"_"+name+2*"_"+"  *")
115     print("*"+4*" "+len(name)*" "+4*" "+"*")
116     print(5*"*"+len(name)*"*"+5*"*")
117
118
119 ##############################################################
    ##########
120 # Question 10
121 ##############################################################
    ##########
122
123 def draw_bike():
124     pass
125
126 ##############################################################
    ##########
127 # Question 11
128 ##############################################################
    ##########
129
130 def alogical(n):
131     ''' (number)->int
132     Precondition: n > 1
133     Returns the number of times n can be divided by 2 before we get something <=1
134     Thus you need to solve n/(2*2*2 ...)=n/(2**x) <= 1 equation for x.
135     That is a definition of log base 2 of n, log_2 n '''
136
137     return math.ceil( math.log2(n) )
```

```
138
139 ##############################################################
    #########
140 # Question 12
141 ##############################################################
    ##########
142
143 def time_format(h,m):
144     ''' (int,int)->str
145     Precondition: 0<=h<=23 and 0<=m<=59
146     Returns time as descriptive string'''
147
148     m_rounded= int(5*round(m/5))
149     if m_rounded==0:
150         return str(h)+" o'clock"
151     elif m_rounded==60:
152         return str((h+1)%24)+" o'clock"
153     elif m_rounded==30:
154         return "half past "+str(h)+" o'clock"
155     elif m_rounded<30:
156         return str(m_rounded)+" minutes past "+str(h)+" o'clock"
157     else:
158         return str(60-m_rounded)+" minutes to "+str((h+1)%24)+" o'clock"
159
160
161
162 ##############################################################
    ##########
163 # Question 13
164 ##############################################################
    ##########
165
166 def cad_cashier(price,payment):
167     ''' (number,number)->number
168
169     Preconditions: price and payment are non-negative floats
170     with 2 decimal places; payment>=price and the last decimal
171     in payment is 0 or 5.
172
173     Returns the change in CAD'''
174
175     change=round(payment-price,2)
176     change=(5/100)*round(change*100/5)
177     change=round(change,2)
178     return change
179
180
181
182 ##############################################################
    ##########
183 # Question 14
184 ##############################################################
    ##########
185 def min_CAD_coins(price,payment):
```

```
186     '''
187       (number,number)->tuple or (int, int, int, int, int)
188
189       Preconditions: price and payment are non-negative floats
190       with 2 decimal places; payment>=price and the last decimal in payment is 0 or 5.
191
192       Returns the minimum number of toonies, loonies, quarters,
193       dimes, and nickels that sum up to required change
194       Precondtion: amount is a positive number
195     '''
196
197     change=cad_cashier(price,payment)
198     dollars=int(change)
199     toonies=dollars//2
200     loonies=dollars%2
201
202     change=change - toonies*2 - loonies
203     cents=round(change*100)
204
205     quarters=cents//25
206     cents=cents%25     #this computes remaining number of cents after quarters are removed
207
208     dimes=cents//10
209     cents=cents%10
210
211     nickels=cents//5
212
213     return (toonies, loonies, quarters, dimes, nickels)
214
215
216   ##############################################################
      ##########
217   # Question 14
218   ##############################################################
      ##########
219   def min_CAD_coins_v2(price,payment):
220     '''
221       (number,number)->tuple or (int, int, int, int, int)
222
223       Preconditions: price and payment are non-negative floats
224       with 2 decimal places; payment>=price and the last decimal in payment is 0 or 5.
225
226       Returns the minimum number of toonies, loonies, quarters,
227       dimes, and nickels that sum up to required change
228       Precondtion: amount is a positive number
229     '''
230
231     change=cad_cashier(price,payment)
232     cents=round(change*100)
233
234     toonies=cents//200
235     cents=cents%200
236     loonies=cents//100
237
```

```
238     cents=cents%100
239
240     quarters=cents//25
241     cents=cents%25     #this computes remaining number of cents after quarters are removed
242
243     dimes=cents//10
244     cents=cents%10
245
246     nickels=cents//5
247
248     return (toonies, loonies, quarters, dimes, nickels)
249
```