



1. \_\_info\_\_ =\
2. """
3. Henry Post,
4. 1/15/2018,
5. Lab 01: Appliance Cost Calculator,
7. ITMD413,
8. IIT Spring 2018
10. """
12. **def** ensure(v, f, e):
13. **if**(f(v)):
14. **return** v
15. **else**:
16. **raise** e
18. **def** ensurePositive(v):
19. **return** ensure(v, **lambda** x: (True **if** x >= 0 **else** False), ValueError)

22. **class** appliance:
24. name:str = "generic fridge appliance"
25. costkWh:float = 0.15
26. annualUsage:float = 200.0
27. meta:object = None
29. **def** \_\_init\_\_(self, n=name, c=costkWh, au=annualUsage, m=None) -> None:
30. self.name = n
31. self.costkWh = float(c)
32. self.annualUsage = float(au)
33. self.meta = m
35. **def** totalCost(self) -> float:
36. **return** (self.costkWh \* self.annualUsage)
38. **def** prompt(self) -> None:
39. n =  input("Name:     > ")
41. **while** True:
42. **try**:
43. c = ensurePositive(float(input("$/kWh:    > $")))
44. **except** ValueError:
45. **print**("Please input a valid, non-negative number for $/kWh.")
46. **continue**
47. **else**:
48. **break**
50. **while** True:
51. **try**:
52. au = ensurePositive(float(input("kWh/year: > ")))
53. **except** ValueError:
54. **print**("Please input a valid, non-negative number for kWh/year.")
55. **continue**
56. **else**:
57. **break**
59. self.\_\_init\_\_(n, c, au)
61. **def** \_\_str\_\_(self) -> str:
62. **return** '"[{:<15}]" ' \
63. 'uses [{:<7.2f}]kWh/year ' \
64. 'at [{:<7.2f}]¢/KW, ' \
65. 'for a total of $[{:<7.2f}]/year.'.format(self.name,self.annualUsage,self.costkWh,self.totalCost())
67. **def** \_\_repr\_\_(self) -> str:
68. **return** str(self)

71. **class** appliances:
73. list: appliance = []
75. **def** \_\_init\_\_(self, l=list) -> None:
76. self.list = l
78. **def** peek(self) -> appliance:
79. **return** self.list[-1]
81. **def** append(self, a) -> None:
82. self.list.append(a)
84. **def** \_\_len\_\_(self) -> int:
85. **return** len(self.list)
87. **def** \_\_iter\_\_(self):
88. **return** self.list.\_\_iter\_\_()
90. **def** prompt(self) -> None:
91. a = appliance()
92. a.prompt()
94. self.append(a)
96. **def** costkWhs(self) -> float:
97. totCost = 0.0
99. **for** appliance **in** self.list:
100. totCost += appliance.costkWh
102. **return** totCost
104. **def** annualUsages(self) -> float:
105. totAU = 0.0
107. **for** appliance **in** self.list:
108. totAU += appliance.annualUsage
110. **return** totAU
112. **def** totalCosts(self) -> float:
113. totCost = 0.0
115. **for** appliance **in** self.list:
116. totCost += appliance.totalCost()
118. **return** totCost
120. **def** \_\_str\_\_(self):
121. ret = ""
122. ret += "{} appliances.\n\n".format(len(self))
123. ret += "{:.2f} total cost/kWh annually.\n".format(self.costkWhs())
124. ret += "{:.2f} total kWh annually.\n".format(self.annualUsages())
125. ret += "${:.2f} total cost annually.\n\n".format(self.totalCosts())
127. ret += "Appliance List:\n"
129. **for** appliance **in** self.list:
130. ret += str(appliance) + "\n"
132. **return** ret
134. inp = "y"
136. **print**("Hello!")
138. aps = appliances()
140. **while**(len(inp) > 0 **and** inp[0].upper() == 'Y'):
142. **print**(f"{len(aps)} appliances so far.")
144. aps.prompt()
145. **print**("That \"{}\" costs ${:.2f} annually.".format(aps.peek().name,aps.peek().totalCost()))

148. **print**("Enter anything other than 'y' to quit and display statistics.")
149. **print**("Continue? (y/n)")
150. inp = input("> ")
152. **print**("Summary: \n")
153. **print**(aps)
155. **print**("Bye!")
157. **print**(\_\_info\_\_)