Placing Patterns

Computer Science - Week 6  
Jul 29, 2022 - Version 1.0.2

Please make sure that all members of the group place their UD **email** AND **name** below.

Choose roles following the [instructions here](https://blockpy.cis.udel.edu/assignments/reading/bakery_appendix_pogil).

You should work in groups of 3. If you cannot find 3 group members, then work in groups of 2.

| **Role** | **Name** | **Email** |
| --- | --- | --- |
| **Manager** |  |  |
| **Speaker** |  |  |
| **Recorder** |  |  |

# 1) Use Vocabulary

Fill in the Vocabulary Table on the right by using the best term from the Vocabulary Terms List below the image.

Not all the words will be used, but you should only use words from the list.

|  | **Vocabulary Table**   | **A** |  | | --- | --- | | **B** |  | | **C** |  | | **D** |  | | **E** |  | | **F** |  | | **G** |  | | **H** | Expected result | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Vocabulary Terms List**   | Parameter  For keyword  Def keyword  Return value | Parentheses  Loop body  ~~Expected result~~  String literal | Argument  Function body  Iteration variable  Iteration list | | --- | --- | --- | | |

# 2) Describe Patterns

Write the name of the most appropriate loop pattern based on the description given on the right hand. Each of the 9 loop patterns will be used only ONCE.

Given a list of values…

| … Produce a new list with each value changed in some way. |  |
| --- | --- |
| … Produce a single value that combines all the values into a single value using some operation. |  |
| … Produce a single element from the list matching some condition. |  |
| … Produce a new list with all the values from the original list until a value that matches some condition. |  |
| … Produce the largest value from the list. |  |
| … Produce a number representing all the values of the list added together. |  |
| … Produce a new list with the same or fewer elements as the original list, only including elements according to some condition. |  |
| … Produce a number indicating how many values were in the list. |  |
| … Produce the smallest value from the list. |  |

The loop patterns are:

| **Count** | **Sum** | **Accumulate** | **Map** | **Filter** |
| --- | --- | --- | --- | --- |
| **Find** | **Take** | **Minimum** | **Maximum** |  |

# 3) Label Patterns

In each of the boxes below, we are showing the INPUTs and OUTPUTs of some test cases. Choose the general Loop Pattern(s) that would cause the given change.

In some cases, there might be multiple valid answers. Choose the answer that is more appropriate and uses the fewest number of patterns that you can think of. Be prepared to justify your answer.

| **Given the input…** | **And the output…** | **You could use the loop patterns…** |
| --- | --- | --- |
| [1, 2, 3, 4, 5] | [2, 4, 6, 8, 10] | Map |
| [1, 2, 3] | ["♥", "♦", "♠"] |  |
| ["♥", "♦", "♠"] | [1, 2, 3] |  |
| ["♥", "♦", "♠"] | ["♥", "♠"] |  |
| ["♥", "♦", "♠", "♥"] | ["♥", "♥"] |  |
| ["♥", "♦", "♠"] | "♦" |  |
| [6, 1, 9, 3, 4] | 1 |  |
| [6, 1, 9, 3, 4] | 9 |  |
| [6, 1, 9, 3, 4] | 5 |  |
| [6, 1, 9, 3, 4] | 23 |  |
| [6, 1, 9, 3, 4] | 6 |  |
| ["♥", "♦", "♠"] | "♥♦♠" | Accumulate |
| ["♥", "♥", "♥", "♥"] | True |  |
| [4, 9, 8, -7, 4, 3] | [4, 9, 8] |  |
| [4, 9, 8, -7, 4, 3] | [4, 9, 8, 4, 3] |  |
| ["♥", "♦", "♠"] | 107 |  |
| [True, True, False, True] | False |  |
| ["♥", "♦", "♠"] | [True, False, True] |  |
| ["♥", "♦", "♦", "♥"] | [3, 3] |  |

# 4) Missing Bodies

In the 5 code snippets below, the loop’s body is missing. Fill in the boxes with the appropriate missing code to fulfill the test cases.

In some cases, you may need to write more than one line.

We will not grade on capitalization, but get the syntax correct otherwise!

| def count\_cards(cards: list[str]) -> int:  count = 0  for card in cards:   |  | | --- |   return count  assert\_equal(count\_cards([]), 0)  assert\_equal(count\_cards(["♠3", "♦4", "♠2"]), 3)  assert\_equal(count\_cards(["♥2", "♥5"]), 2) |
| --- | --- |
| def count\_hearts(cards: list[str]) -> int:  count = 0  for card in cards:   |  | | --- |   return count  assert\_equal(count\_hearts([]), 0)  assert\_equal(count\_hearts(["♥2", "♥5"]), 2)  assert\_equal(count\_hearts(["♠4", "♥2"]), 1)  assert\_equal(count\_hearts(["♠3", "♦4", "♠2"]), 0) |
| def filter\_hearts(cards: list[str]) -> list[str]:  kept\_cards = []  for card in cards:   |  | | --- |   return kept\_cards  assert\_equal(filter\_hearts(["♥3", "♦2", "♣9"]), ["♥3"])  assert\_equal(filter\_hearts(["♥3", "♣4", "♥2"]), ["♥3", "♥2"])  assert\_equal(filter\_hearts(["♣3", "♠4", "♦2"]), [])  assert\_equal(filter\_hearts([]), None) |
| def find\_heart(cards: list[str]) -> str:  found = None  for card in cards:   |  | | --- |   return found  assert\_equal(find\_heart(["♣3", "♦2", "♥9"]), "♥9")  assert\_equal(find\_heart(["♣3", "♥4", "♦2"]), "♥4")  assert\_equal(find\_heart(["♣3", "♠4", "♦2"]), None)  assert\_equal(find\_heart([]), None) |
| def is\_all\_hearts(cards: list[str]) -> bool:  result = True  for card in cards:   |  | | --- |   return result  assert\_equal(is\_all\_hearts(["♥3", "♦2", "♣9"]), False)  assert\_equal(is\_all\_hearts(["♥3", "♥4", "♥2", "♥9"]), True)  assert\_equal(is\_all\_hearts(["♣3", "♠4", "♦2"]), False)  assert\_equal(is\_all\_hearts([]), True) |

# 5) Reflect and Review

Discuss among yourselves: what did you learn from this activity? What was surprising or interesting? If you didn’t learn anything, what do you think we were trying to teach you? How could this activity be improved?

|  |
| --- |
|  |
|  |

# Final Submission

When your team is happy with your answers for all the questions, download this file as a Word Document (docx) and upload the file to the appropriate assignment on Canvas.

Only one member of your team needs to submit.