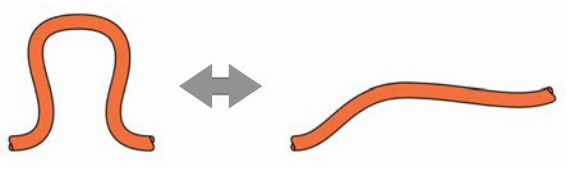
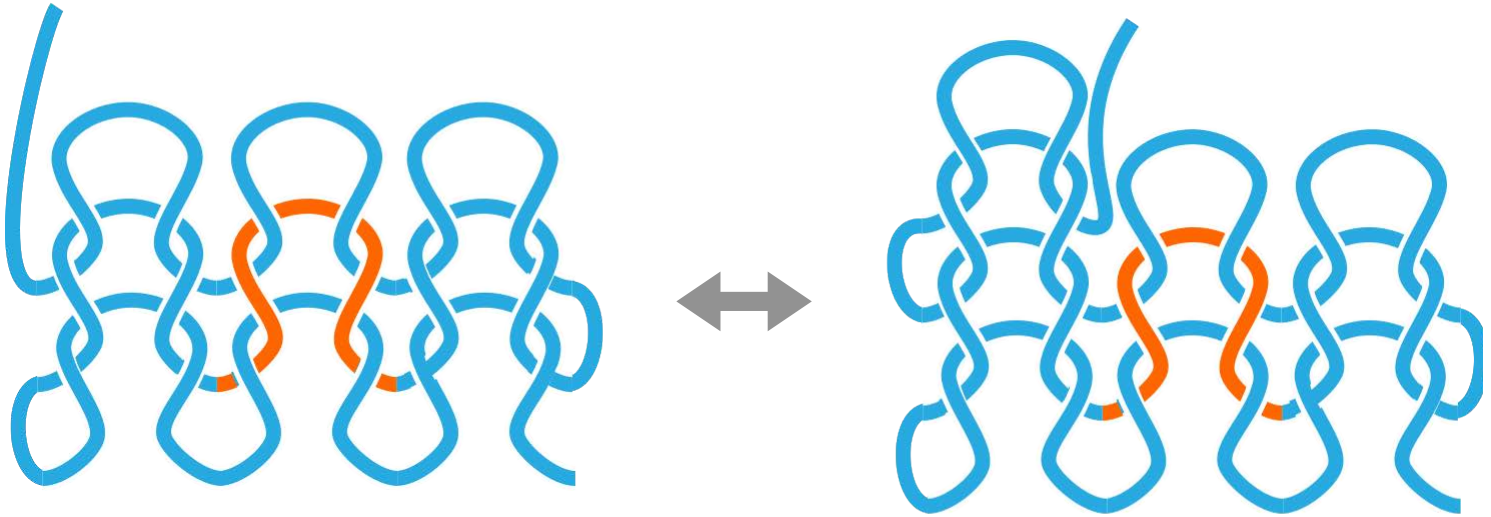
Knitout Programs and Colorwork

Recap: What is Knitting?



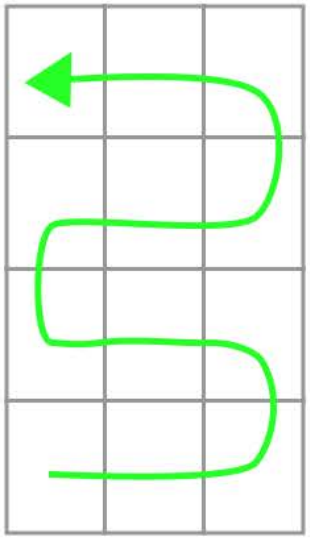
Yarn can be bent to make a loop.

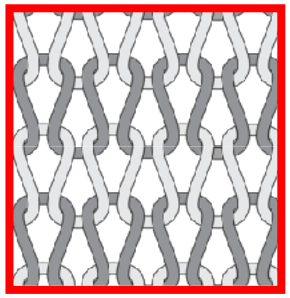


Loops can be pulled through loops to secure previous loops

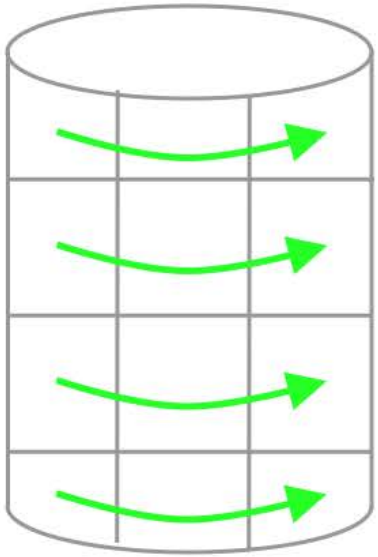
Source: Jenny Lin

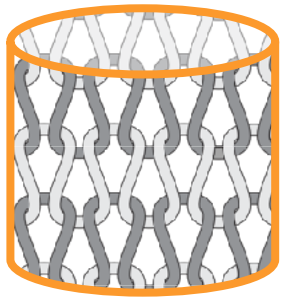
Recap: Basic Structures in Knitting





Sheets

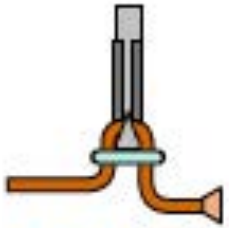




Tubes

Recap: Knit Stitch vs Purl Stitch on the Machine

Back Bed



Purl

(Yarn carrier goes between the two beds)

Front Bed

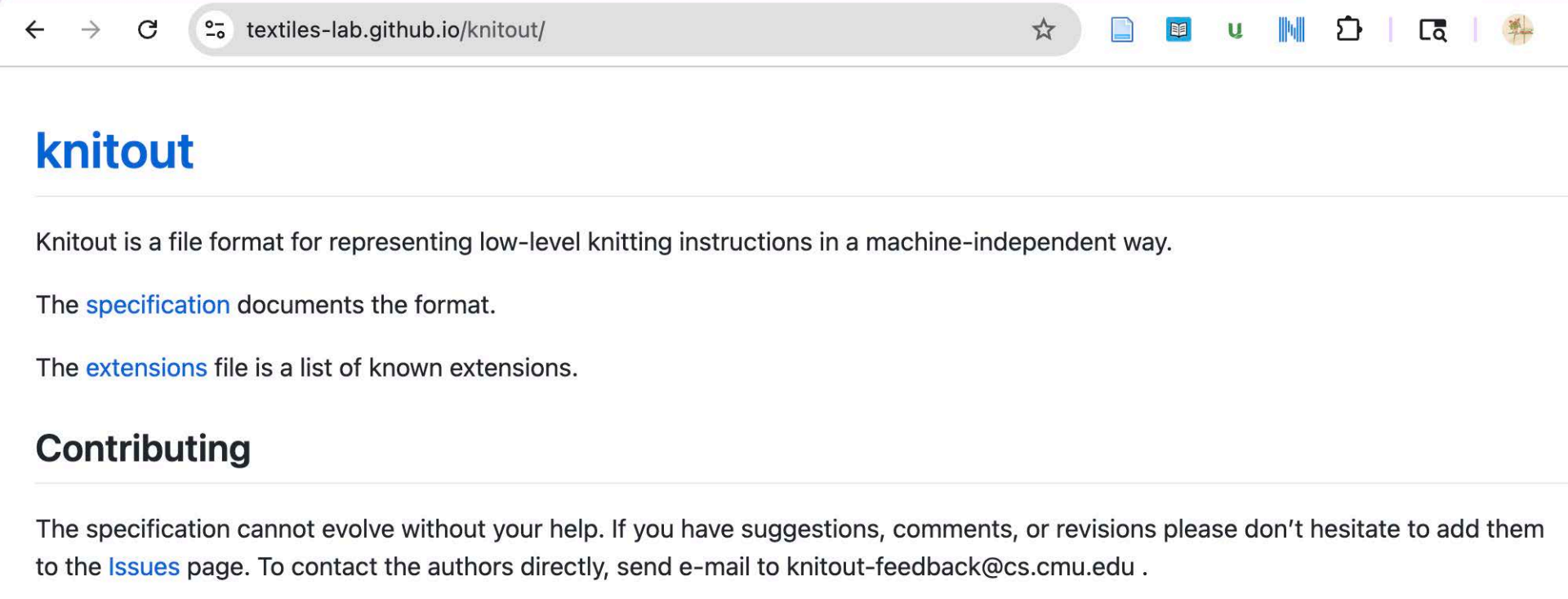
Knit

Agenda Today

- Components in a knitout program
- Colorwork techniques
  - stranded (fair isle)
  - intarsia
- Machine colorwork
  - plating
  - double-knit jacquard
    - tubular jacquard
    - birds-eye jacquard

What is knitout?

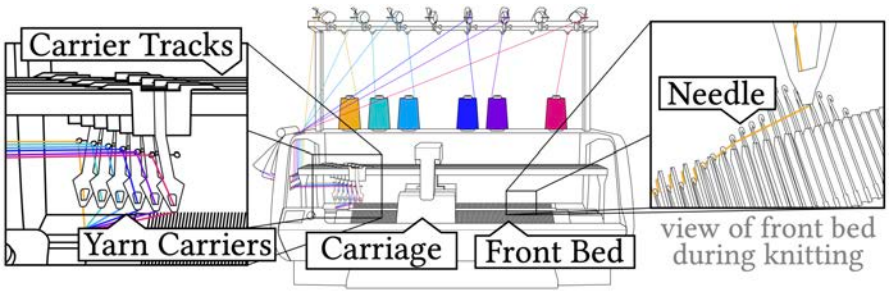
- You've already seen the .k files from HW2
- It's a language for expressing knitting instructions for knitting machines developed by the CMU Textiles Lab



What is knitout?

- It directly connects to how the machine works and each line represents a machine operation, which could include:
  - knit with a yarn
  - bring in a yarn carrier
  - drop a stitch
  - cut the yarn and bring back the yarn carrier
  - ...
- The language has constructs pointing to entities on the machine, including yarn carriers, needles, racking values, etc.

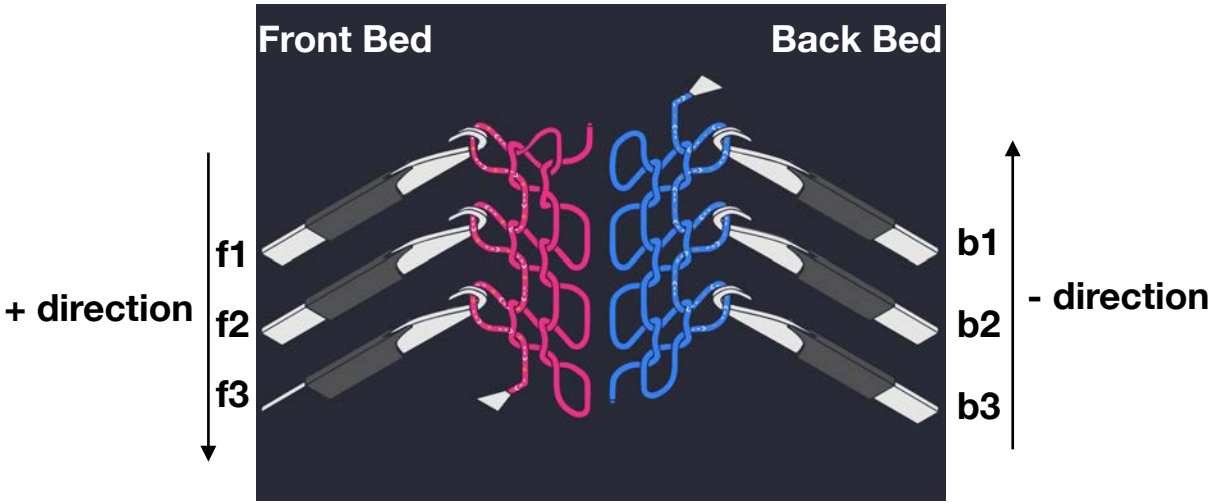
Knitting Machine Structure

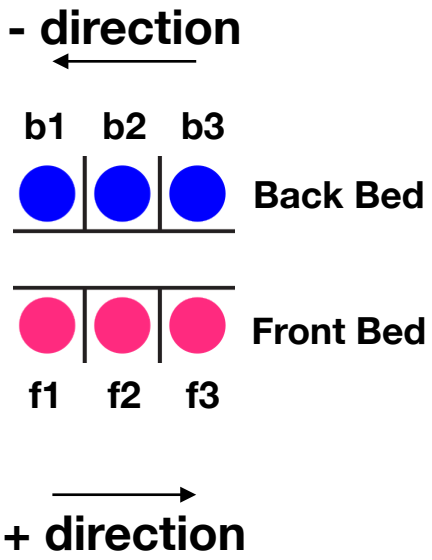


yarns carriers ∈ [1,2,3,4,5,6]

Source: Jenny Lin

Knitting Machine Structure





Source: Jenny Lin

Example from HW2: mode 1 with carrier 2

```
;!knitout-2
;;Carriers: 1 2 3 4 5 6 7 8 9 10
;;Position: Center
;;Width: 450
inhook 2
tuck - f20 2
tuck - f18 2
tuck - f16 2
tuck - f14 2
tuck - f12 2
tuck - f10 2
tuck - f8 2
tuck - f6 2
tuck - f4 2
tuck - f2 2
```

;! signifies the version of the knitout language

;; marks a comment line, just like in Python we use # for inline comment

The first few lines of ;; are file headers, which specifies basic information about the machine and where you want the piece to start on the bed, and the bed width in needles

Example from HW2: mode 1 with carrier 2

```
;!knitout-2
;;Carriers: 1 2 3 4 5 6 7 8 9 10
;;Position: Center
;;Width: 450
inhook 2
tuck - f20 2
tuck - f18 2
tuck - f16 2
tuck - f14 2
tuck - f12 2
tuck - f10 2
tuck - f8 2
tuck - f6 2
tuck - f4 2
tuck - f2 2
```

1~10

syntax: inhook <carrier number>

Example from HW2: mode 1 with carrier 2

```
;!knitout-2
;;Carriers: 1 2 3 4 5 6 7 8 9 10
;;Position: Center
;;Width: 450
inhook 2
tuck - f20 2
tuck - f18 2
tuck - f16 2
tuck - f14 2
tuck - f12 2
tuck - f10 2
tuck - f8 2
tuck - f6 2
tuck - f4 2
tuck - f2 2
```

f/b 1~450

+/- <bed><needle number>

syntax: tuck <direction> <needle> <carrier>

an operation code, there are also **knit**, **miss**, that share the same syntax

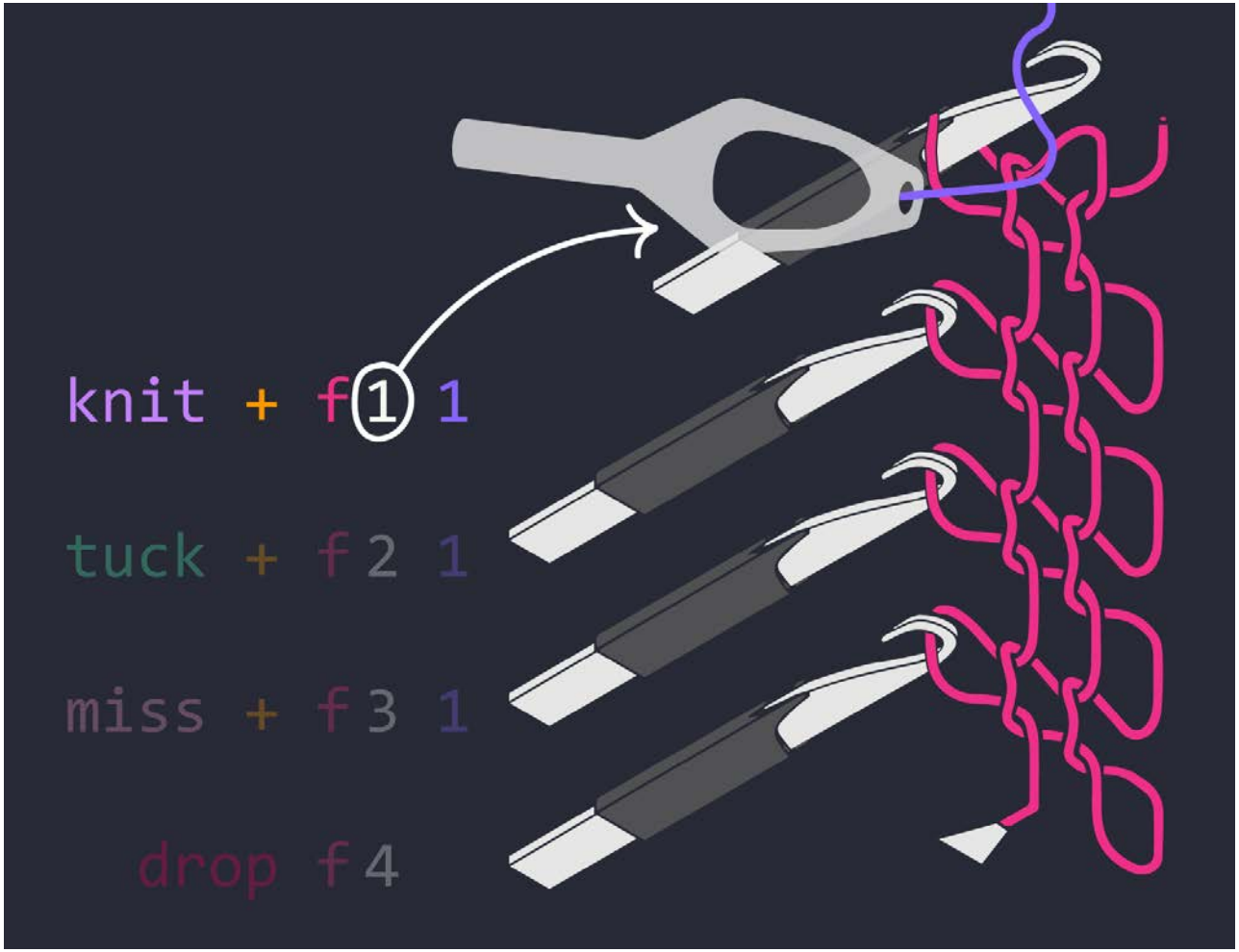
Op Codes in knitout (functions!)

textiles-lab.github.io/knitout/knitout.html

Opcodes

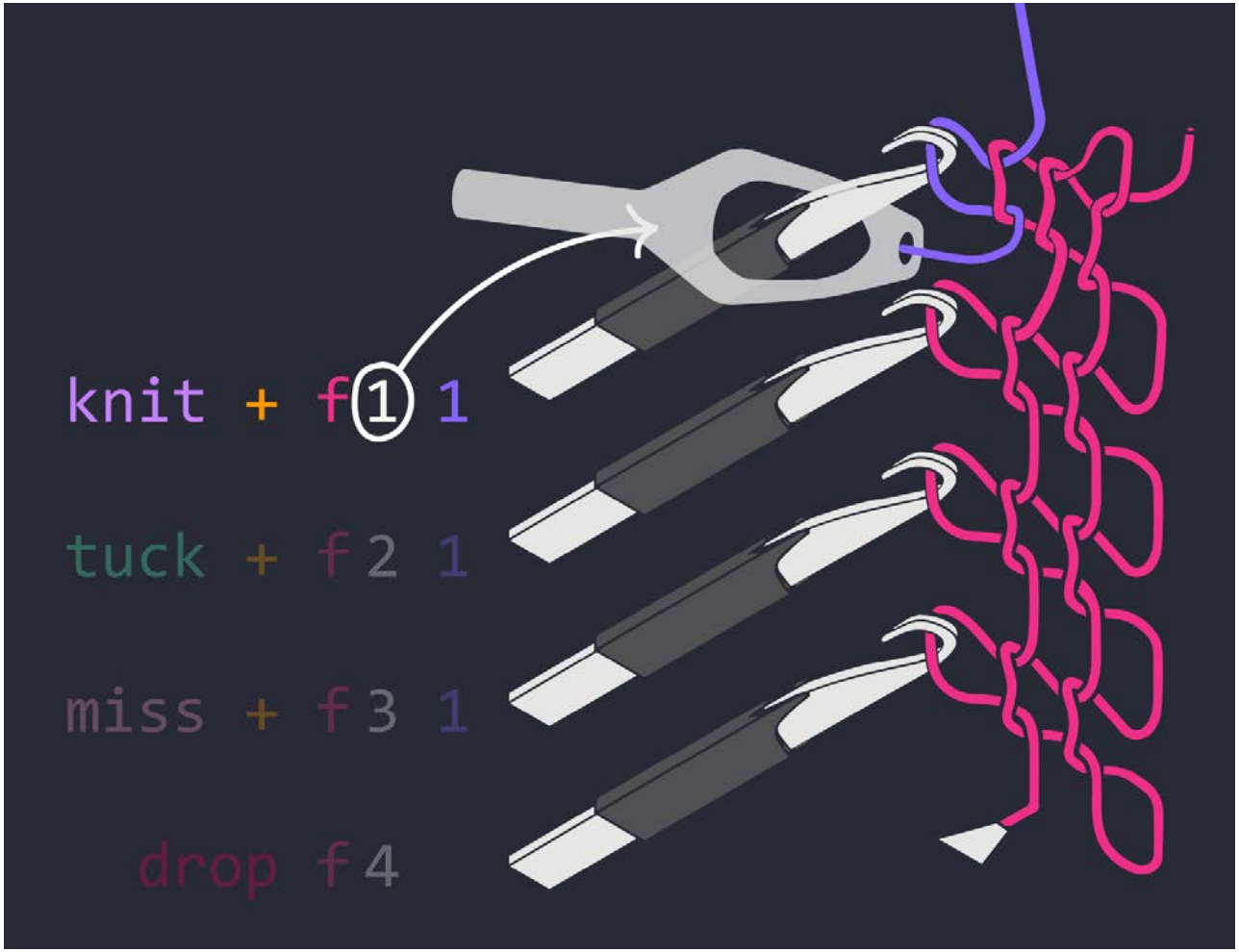
Opcode	Parameters	Description
in	CS	Indicate that the given carrier set should be brought into action from the yarn grippers when next used.
inhook	CS	Indicate that the given carrier set should be brought into action using the yarn inserting hook when next used. The inserting hook will be parked just before the first stitch made with the carriers.
releasehook	CS	Release the yarns currently held in the yarn inserting hook. Must be proceeded by a call to inhook with the same carrier set and at least one knitting operation.
out	CS	Bring a set of carriers out of action by directly moving into the grippers.
outhook	CS	Bring a set of carriers out of action, using a yarn inserting hook to bring their yarns to the grippers.
stitch	L T	Before forming a loop, pull needle by L machine units. After forming a loop, pull needle by T machine units.
rack	R	Set the offset of the back bed relative to the front bed.
knit	D N CS	Pull a loop formed in direction D by the yarns in carriers CS through the loops on needle N, dropping them in the process. Knitting with an empty carrier set will drop.
tuck	D N CS	Add a loop formed in direction D by the yarns held by carriers in CS to those already on needle N. Tucking with an empty carrier set will pull on the stitches without doing anything else (an "a-miss").
split	D N N2 CS	Pull a loop formed in direction D by the yarns in carriers CS through the loops on needle N, transferring the old loops to opposite-bed needle N2 in the process. Splitting with an empty carrier set will transfer.

A Little Knitting Program



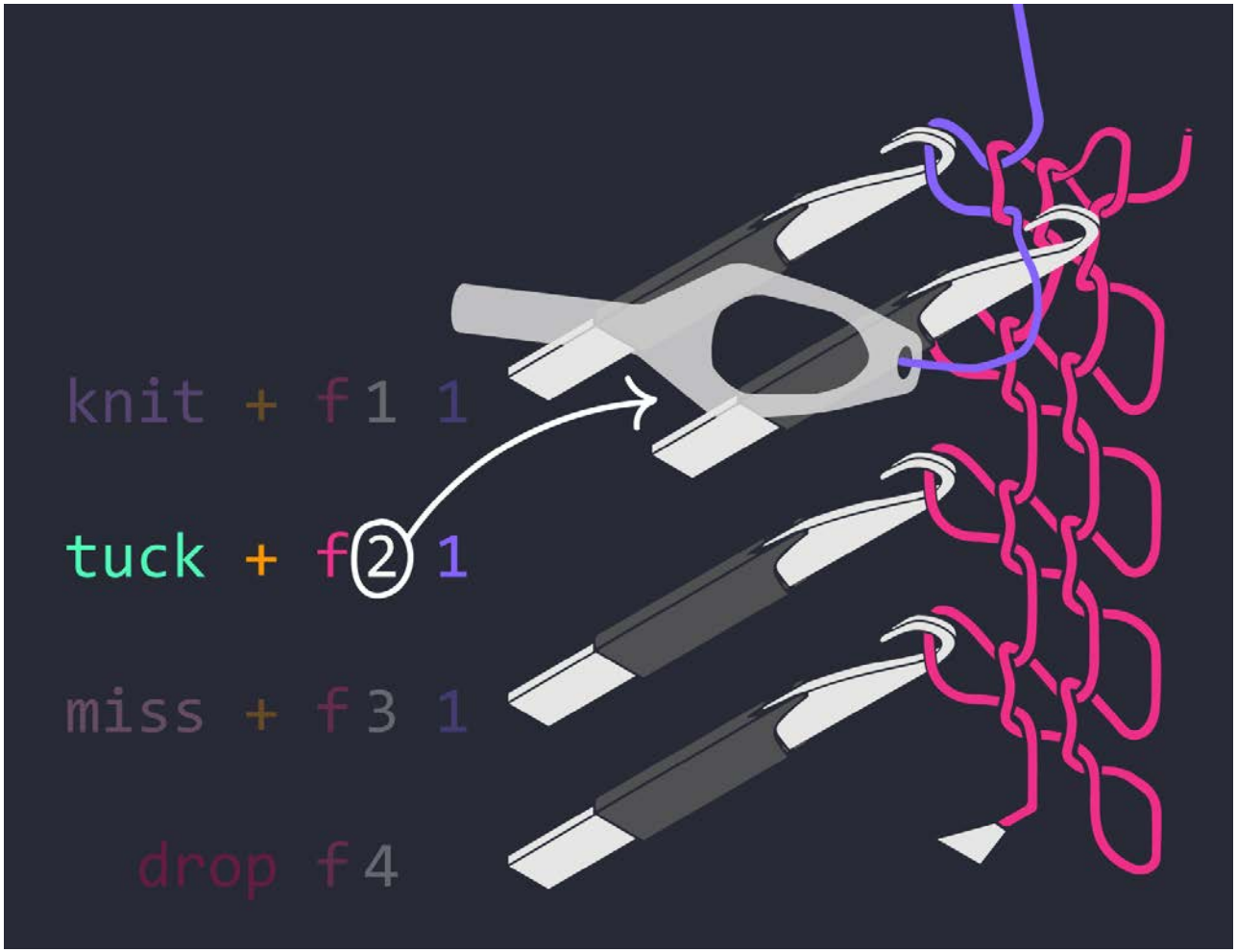
Source: Jenny Lin

A Little Knitting Program



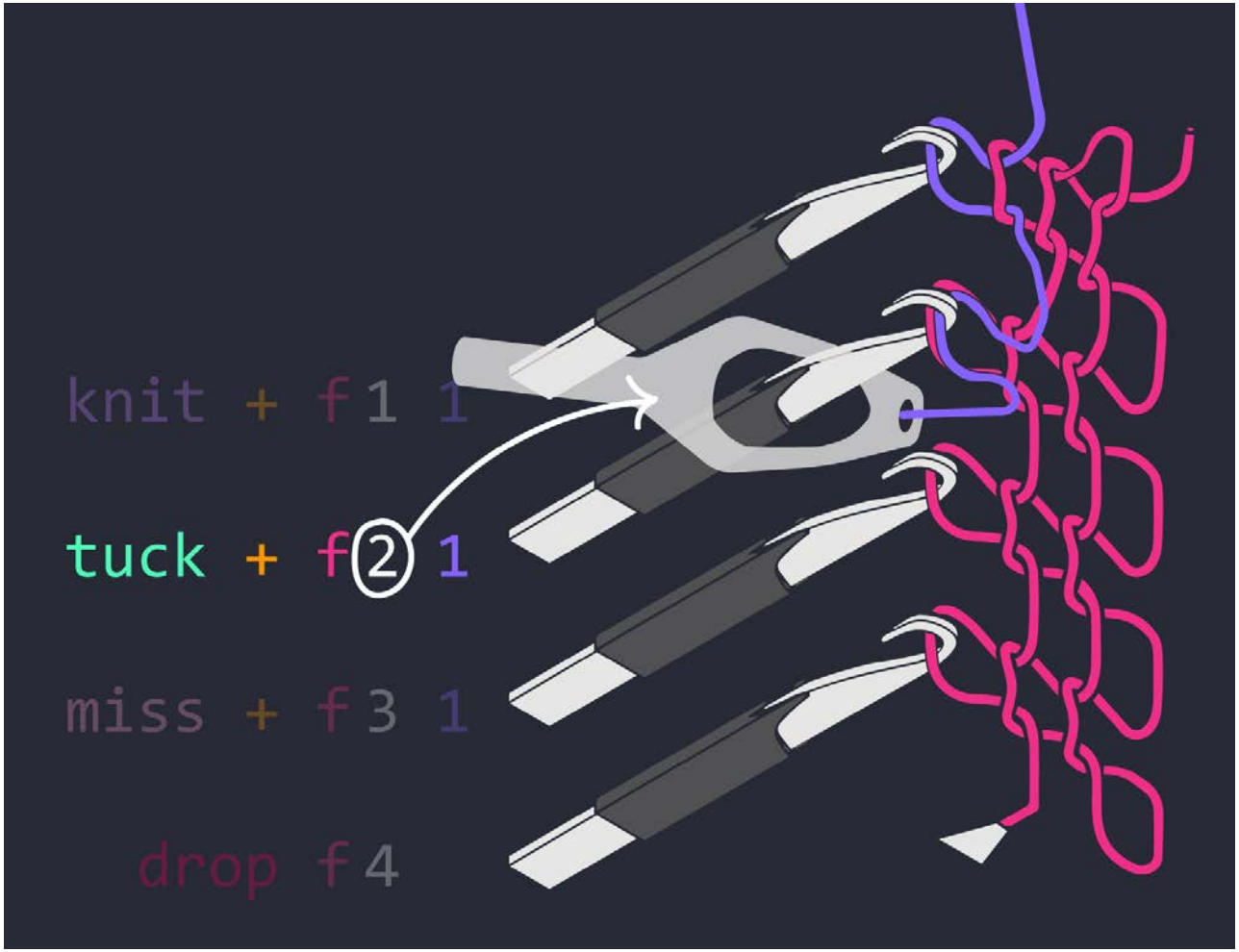
Source: Jenny Lin

A Little Knitting Program



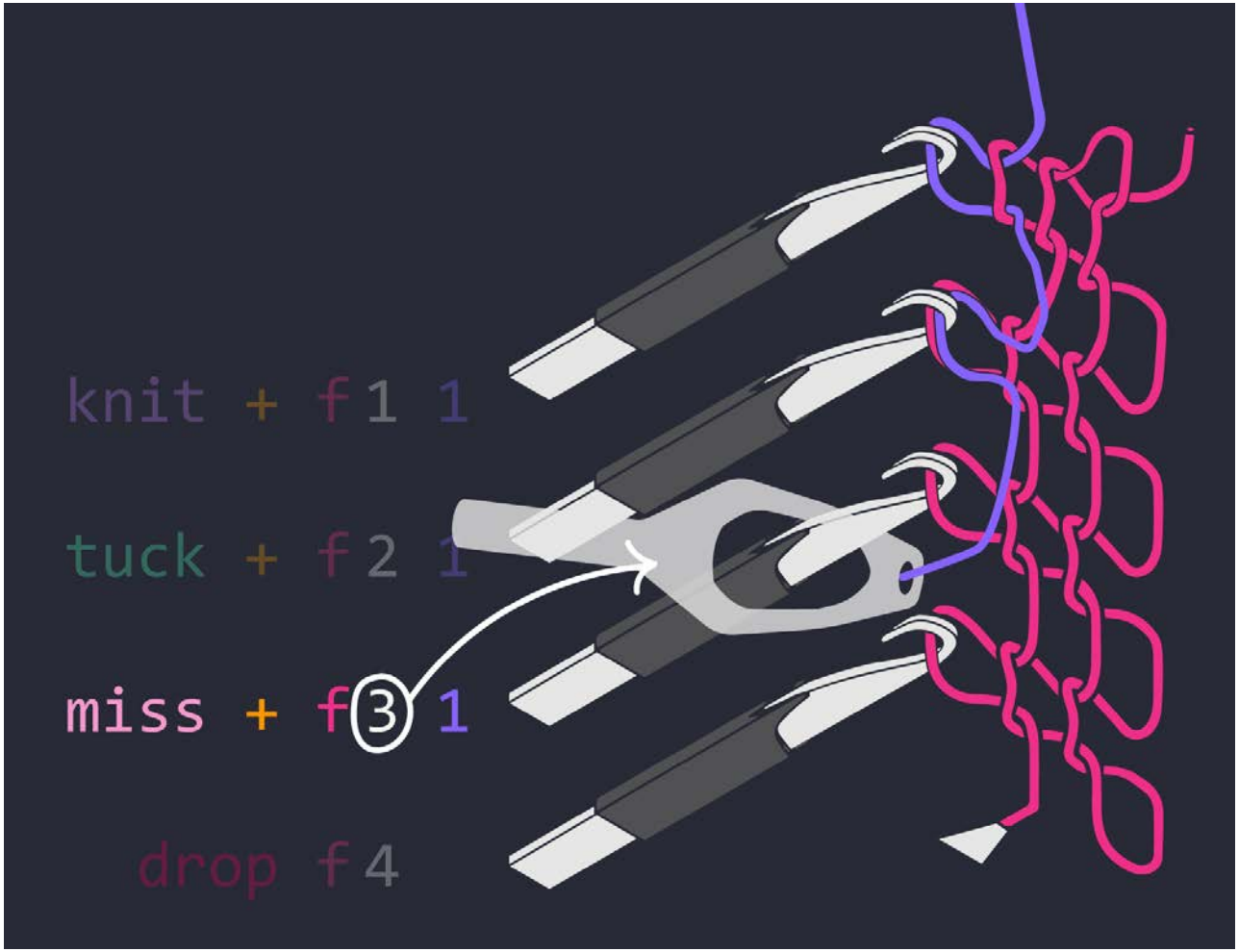
Source: Jenny Lin

A Little Knitting Program



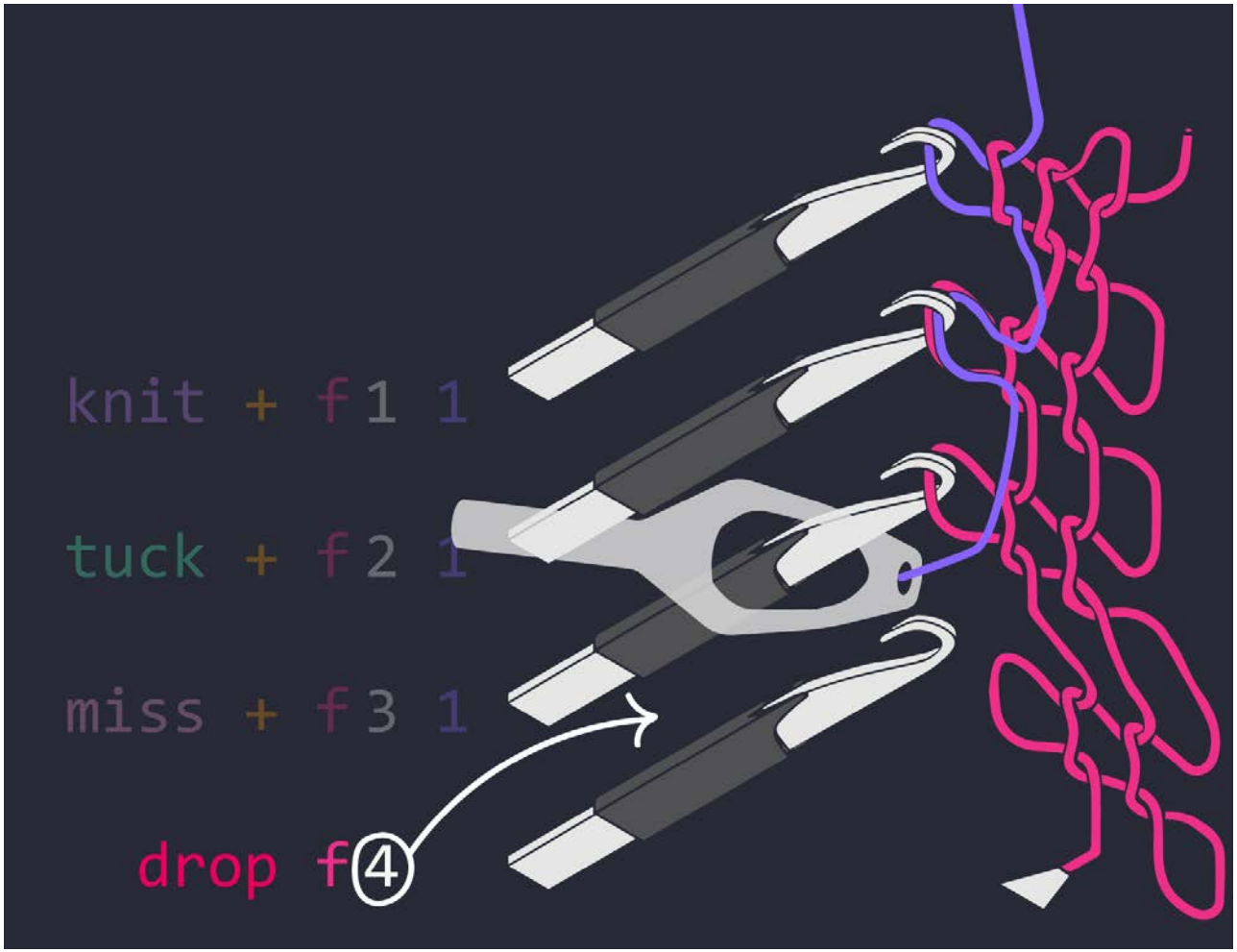
Source: Jenny Lin

A Little Knitting Program



Source: Jenny Lin

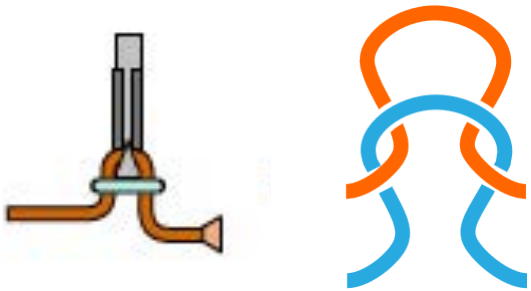
A Little Knitting Program



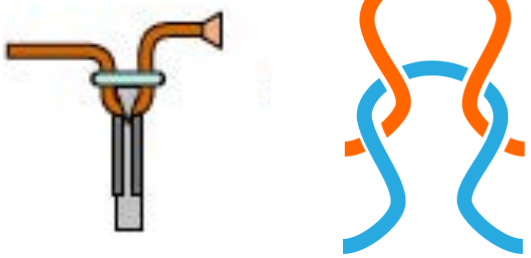
Source: Jenny Lin

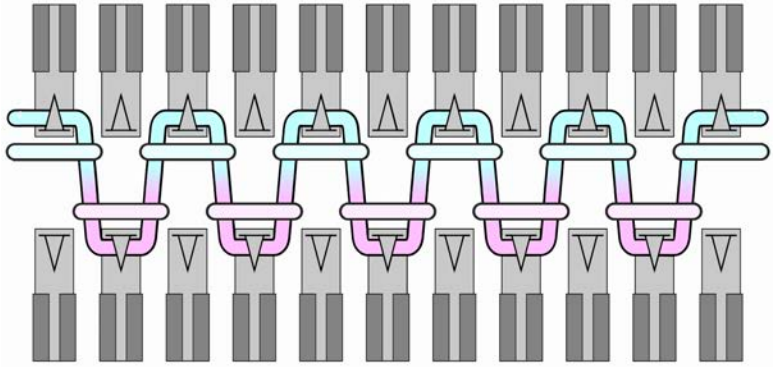
Front and Back of Knitting

Back Bed



Front Bed





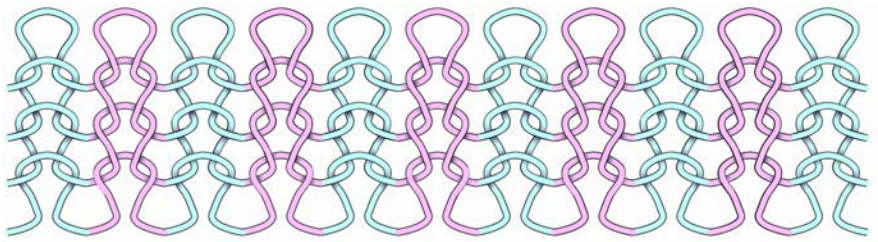
knit + b1 1
knit + f2 1
knit + b3 1
knit + f4 1
knit + b5 1
knit + f6 1
knit + b7 1
knit + f8 1

A sequence of knit operations with the same carrier results in connected loops even if the needles aren't adjacent

Source: Jenny Lin

Knit Textures that Knit on both Front and Back Beds

Rib stitch
Alternating columns of front and back knits

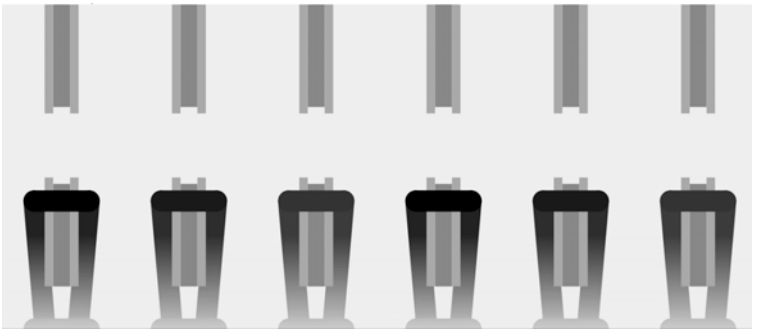


Source: Jenny Lin

Knit Textures that Requires Alternating on Front and Back Beds

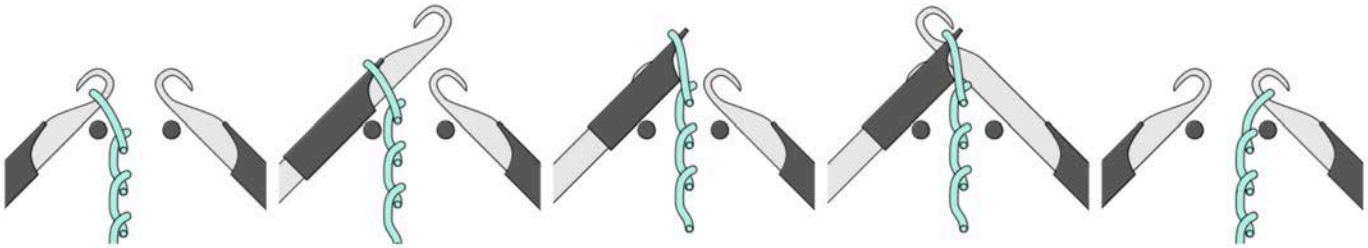
Seed stitch
A checkerboard of front and back knits

We need transfers!

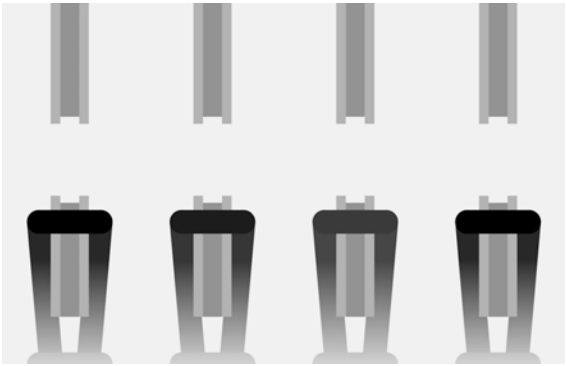


Source: Jenny Lin

Transfers



Back Bed



Front Bed

Transfers move loops from a needle to the needle across from it

xfer f3 b3
xfer b3 f3

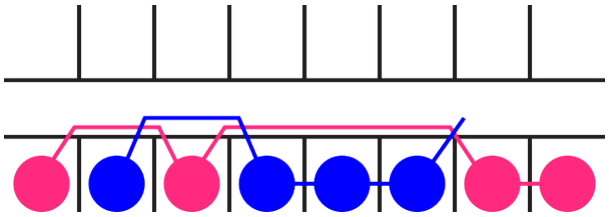
Source: Jenny Lin

Colorwork in Knitting: Stranded

source: https://knittingit.com/colorwork-knitting-stranded-and-fair-isle-explained/

Stranded on Knitting Machine

Easy-ish to write code for
Slightly less bleed through
Floats on the wrong side of fabric
Best for short, regular patterns



knit + f1 1
knit + f3 1
knit + f7 1
knit + f8 1
knit + f2 2
knit + f4 2
knit + f5 2
knit + f6 2

This is what we'll do in HW3!

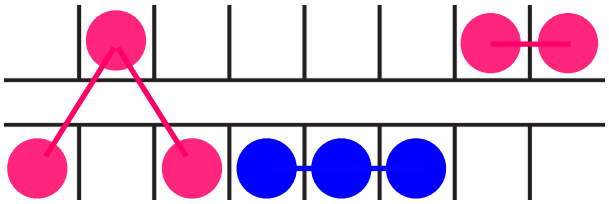
Source: Jenny Lin

Colorwork in Knitting: Intarsia

source: https://knittingit.com/colorwork-knitting-stranded-and-fair-isle-explained/

Intarsia on Knitting Machine

Hard to write code for
(especially on tubes!)
Thin fabric
Good for large blocks of color

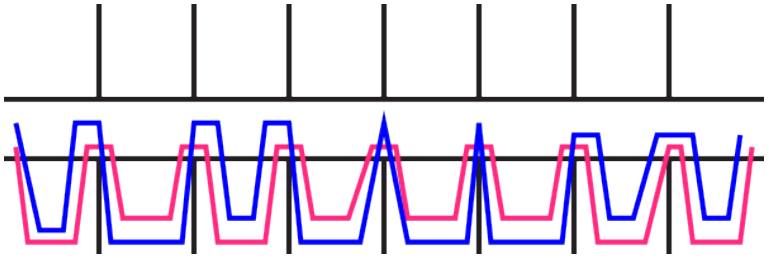


knit + f1 1
knit + b2 1
knit + f3 1
knit + f4 2
knit + f5 2
knit + f6 2
knit + b7 3
knit + b8 3

Source: Jenny Lin

Machine Colorwork: Plating

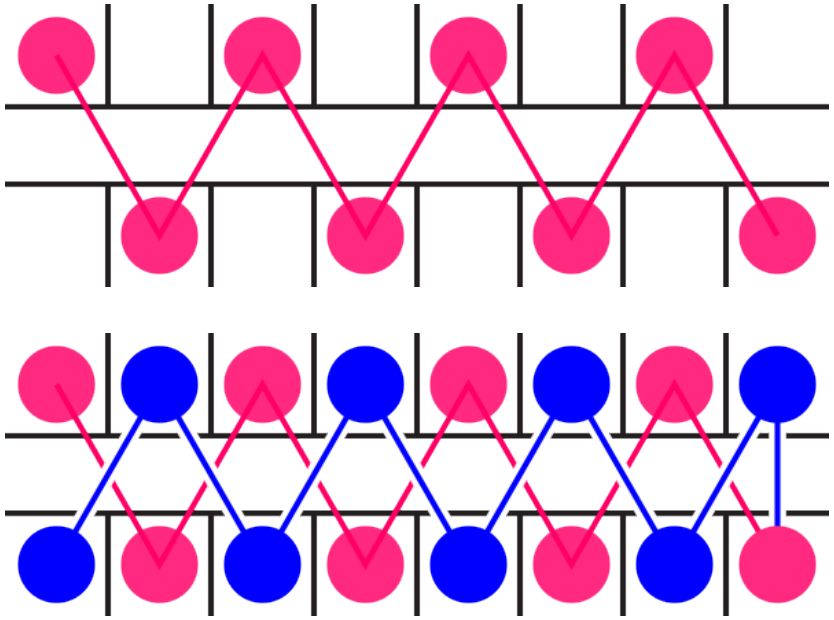
Easy to write code for
Multiple yarns make a thick fabric
Prone to bleed through and color errors



```
knit + f1 1 2
knit + f2 2 1
knit + f3 1 2
knit + f4 2 1
knit + f5 2 1
knit + f6 2 1
knit + f7 1 2
knit + f8 1 2
```

Source: Jenny Lin

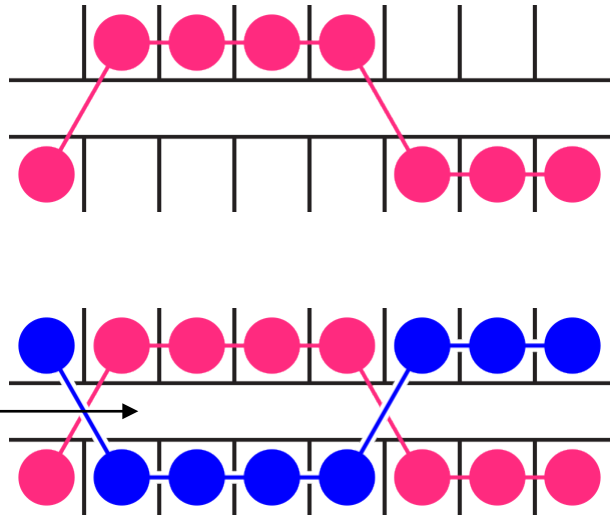
Interlock



Yarn zigzags between both beds in a way that locks both sheets together

Source: Jenny Lin

Machine Colorwork: Tubular Jacquard

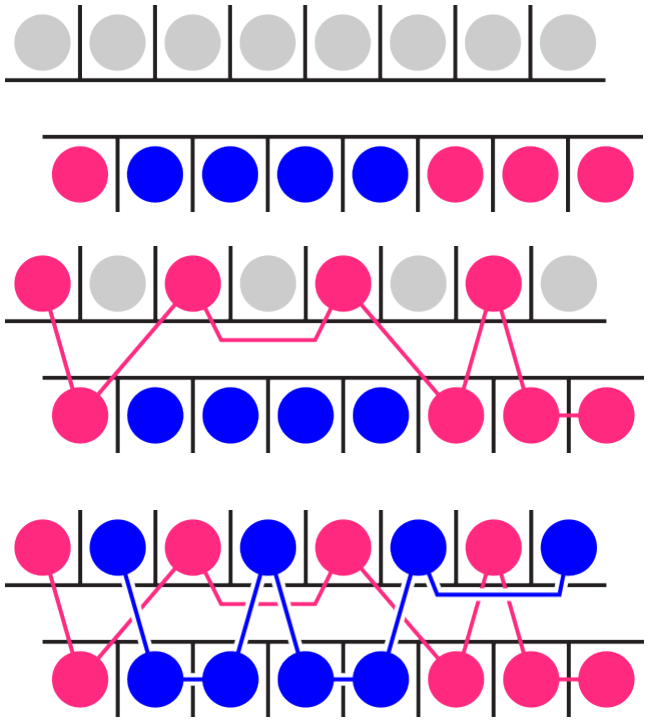


Large regions of one color will make a pocket

Source: Jenny Lin

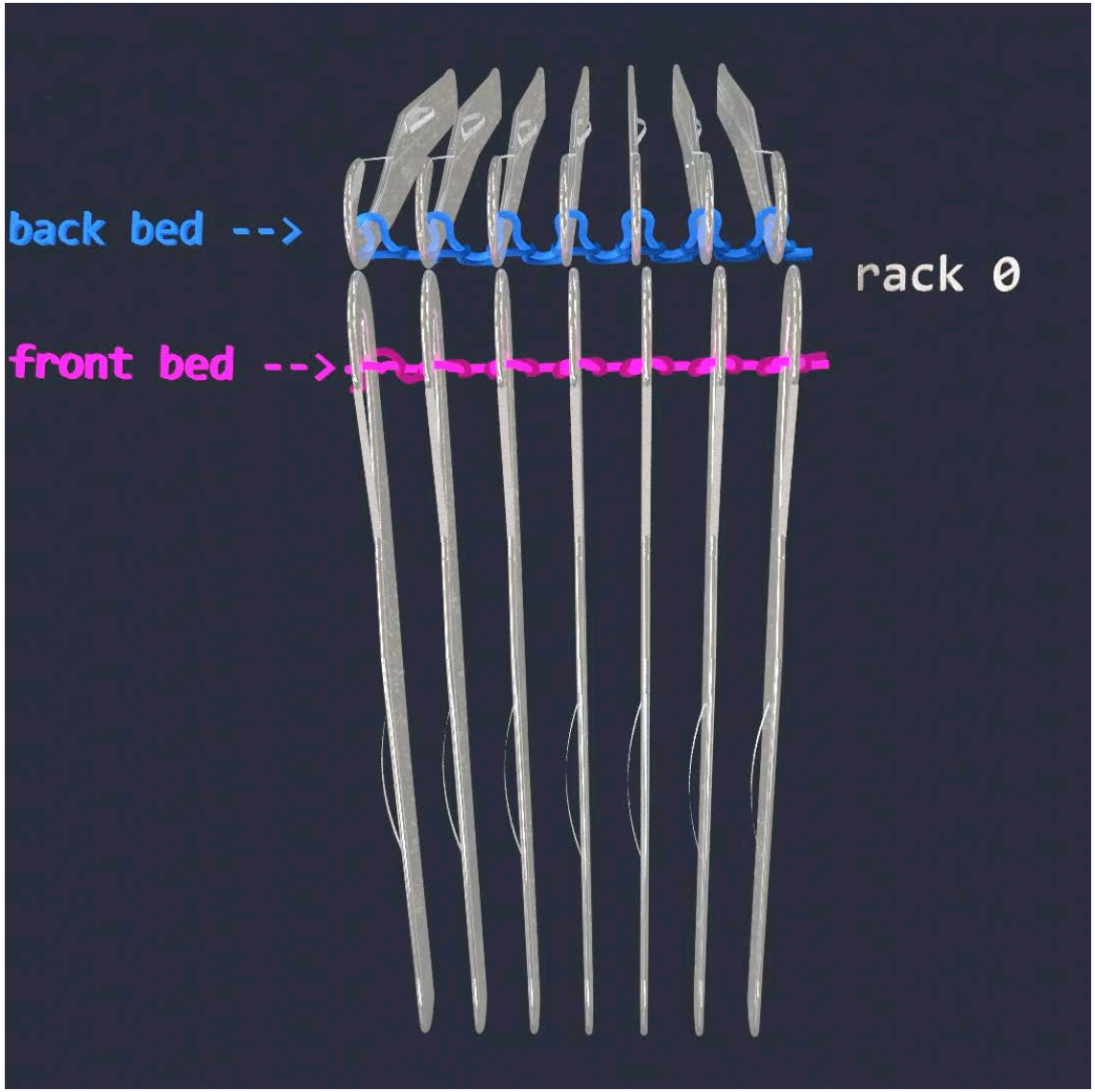
Machine Colorwork: Birds-eye Jacquard

Quarter pitch means needles are all offset



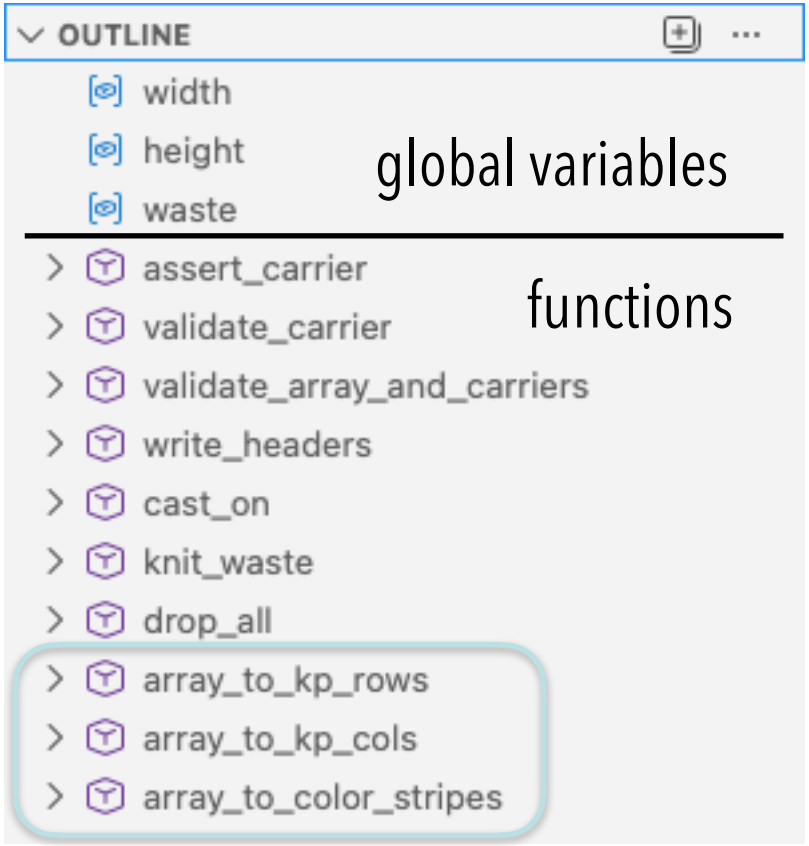
Source: Jenny Lin

Racking



Source: Jenny Lin

Let's read knitout_helpers.py in HW2



These correspond to the three modes used in HW2 pattern_generator.py

assignments > hw2 > knitout_helpers.py > array_to_color_stripes

```
1  """
2  COMP116 HW2 Part 2 - Pattern Generator
3
4  Helper functions for taking a binary array and converting
5  to a knitout file under different pattern variations
6
7  DO NOT MODIFY THIS FILE
8  """
9
10 import sys
11
12 # Pattern size constants.
13 width = 20
14 height = 20
15 waste = 4
16
17 ################ Carrier-related Assertions #################
18 def assert_carrier(carrier, name="Color"):
19     """Validate carrier number."""
20     assert carrier in [1, 2, 3, 4, 5], f"{name} must be one of 1, 2, 3, 4, or 5"
21
22 def validate_carrier(carrier, name="Color"):
23     """Validate carrier numbers."""
24     if isinstance(carrier, int):
25         assert_carrier(carrier)
26     elif isinstance(carrier, str):
27         try:
28             carrier = int(carrier)
29         except:
30             print(f"\n{name} {carrier} is not a number! Exiting...")
31             sys.exit(2)
32         assert_carrier(carrier)
33     return carrier
34
35 def validate_array_and_carriers(rand_array, carriers):
```

Poll Time!

- Read the functions that correspond to each mode (`array_to_kp_rows`, `array_to_kp_cols`, `array_to_color_stripes`) and select all that they have in common.

Let's read the four helper functions

- `write_headers`
- `cast_on`
- `knit_waste`
- `drop_all`

write_headers

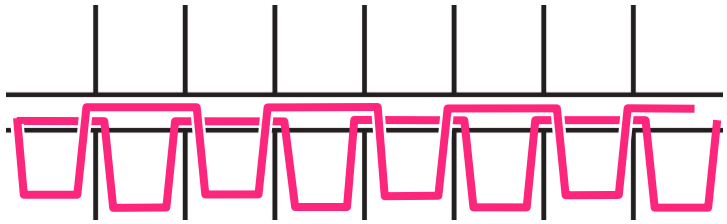
```
def write_headers(knitout_lines):
    """Write standard knitout file headers."""
    knitout_lines.append(";!knitout-2")
    knitout_lines.append(";;Carriers: 1 2 3 4 5 6 7 8 9 10")
    knitout_lines.append(";;Position: Center")
    knitout_lines.append(";;Width: 450")
```

The len(knitout_lines) increase by

4 after write_headers.

cast_on

```
def cast_on(knitout_lines, carrier):
    """Perform alternating tuck cast-on on front bed."""
    for s in range(width, 0, -1):
        if (width - s) % 2 == 0:
            knitout_lines.append(f"tuck - f{s} {carrier}")
    for s in range(1, width + 1):
        if (width - s) % 2 == 1:
            knitout_lines.append(f"tuck + f{s} {carrier}")
```



The len(knitout_lines) increase by

`width` after cast_on.

knit_waste

```
def knit_waste(knitout_lines, carrier):
    """Knit the waste rows."""
    for r in range(waste):
        for s in range(width, 0, -1):
            knitout_lines.append(f"knit - f{s} {carrier}")
        for s in range(1, width + 1):
            knitout_lines.append(f"knit + f{s} {carrier}")
```

The len(knitout_lines) increase by

`waste * width * 2` after cast_on.

drop_all

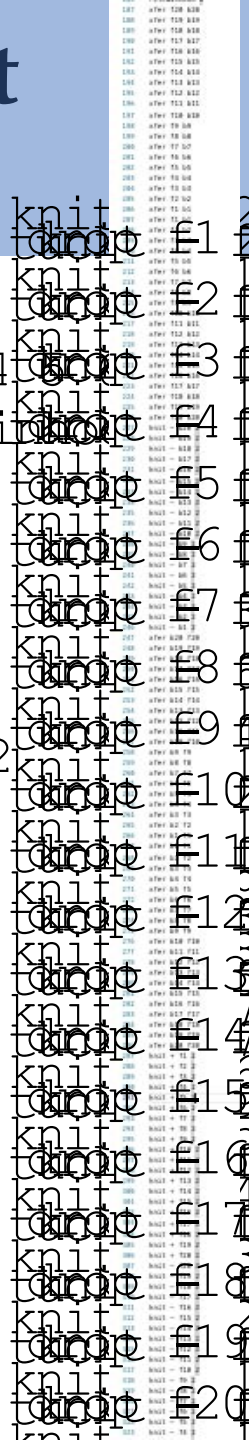
```
def drop_all(knitout_lines):
    """Drop all stitches on front bed."""
    for s in range(1, width + 1):
        knitout_lines.append(f"drop f{s}")
```

The len(knitout_lines) increase by `width` after cast_on.

The structure of a typical knitout program

- headers
- inhook the yarn carrier we need to use
- alternate tuck cast-on with the yarn carrier
- knit some waste rows
- releasehook that yarn carrier
- inhook/releasehook more if needed
- main pattern instructions
- knit some waste rows to end
- outhook the yarn carrier(s) used
- drop all stitches or bind off

```
;!knitout-2
;;Carriers: 1 2 3 4
;;Position: Center
;;Width: 450
releasehook 2
outhook 2
```



To write your own knitout code...

- Reference the functions in knitout_helpers.py
- See `array_to_kp_rows` for transferring entire rows of stitches
- See `array_to_kp_cols` for transferring specific stitches in a row
- See `array_to_color_stripes` for working with more than 1 yarn carrier
- You can write your own functions based on these helpers to produce your own patterns!