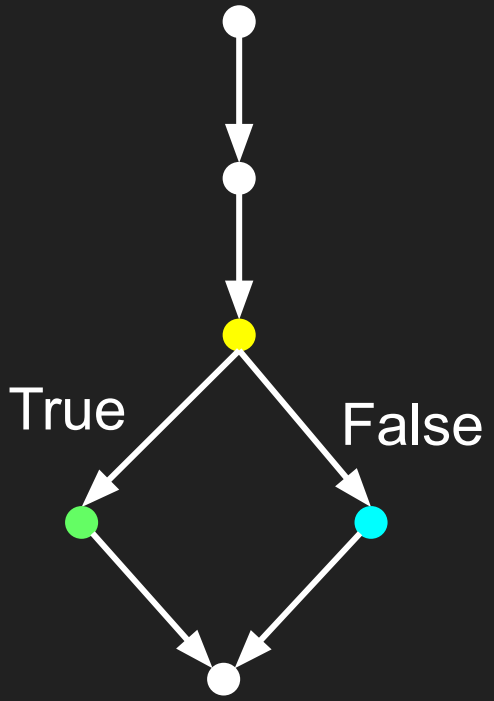
While Loops

First, Review

Conditionals:

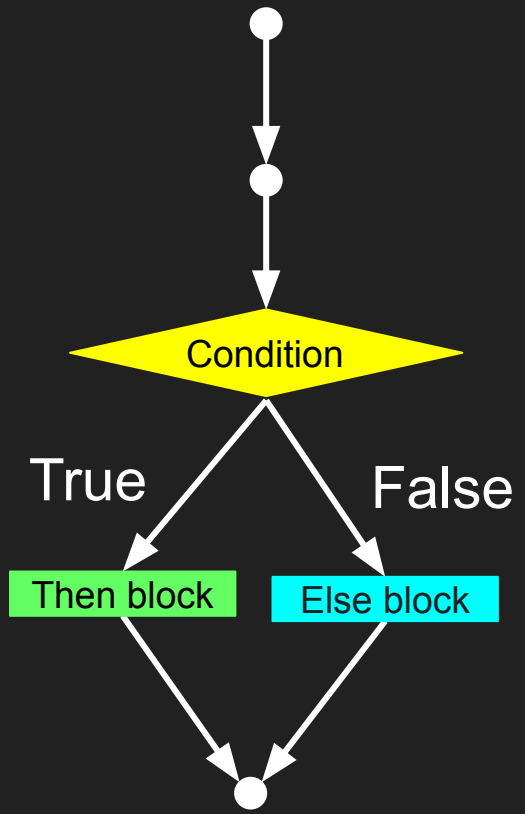
```
if <something>:
    <do something>
else:
    <do something else>
<continue program>
```



First, Review

Conditionals:

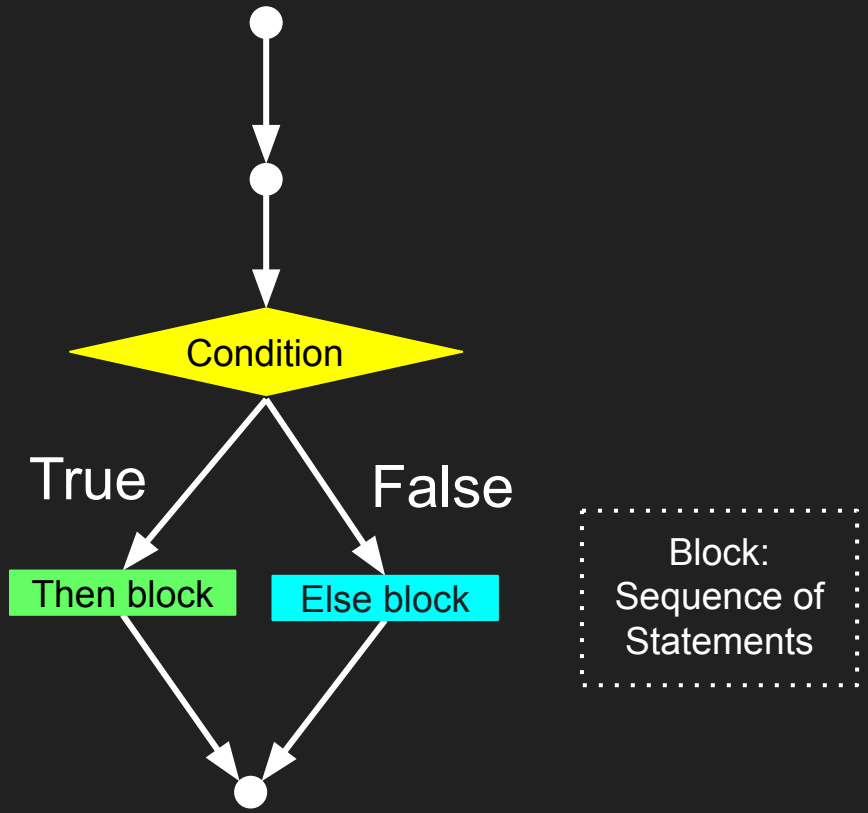
```
if <something>:
    <do something>
else:
    <do something else>
<continue program>
```



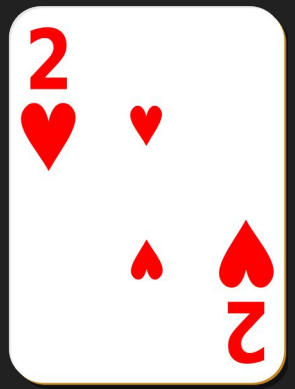
First, Review

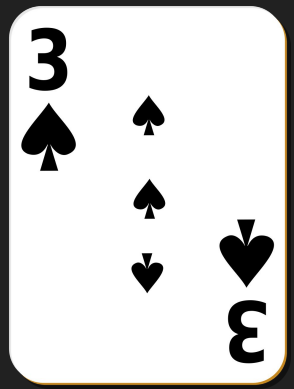
Conditionals:

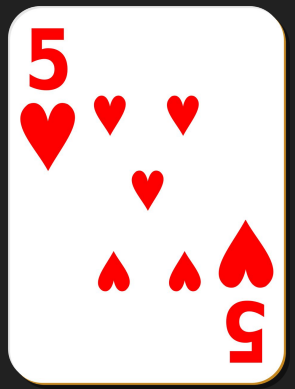
```
if <something>:
    <do something>
else:
    <do something else>
<continue program>
```



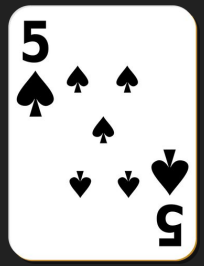
Finding the Lowest Card



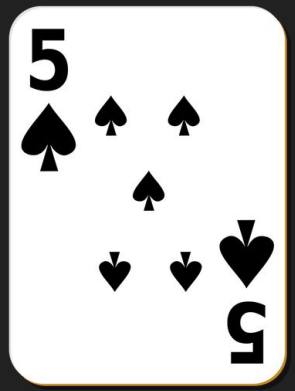


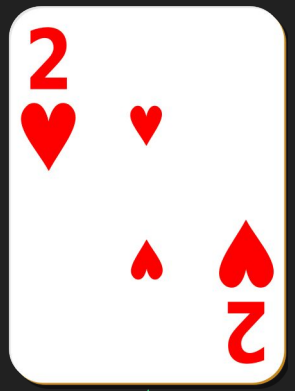


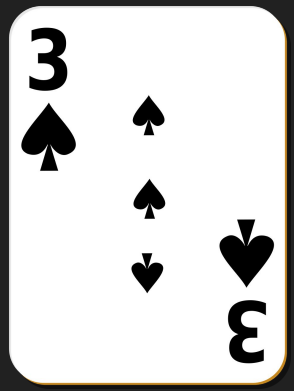
Low card:

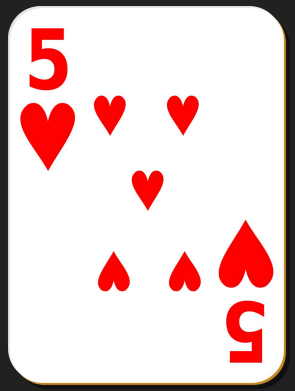


Finding the Lowest Card



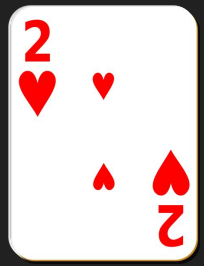




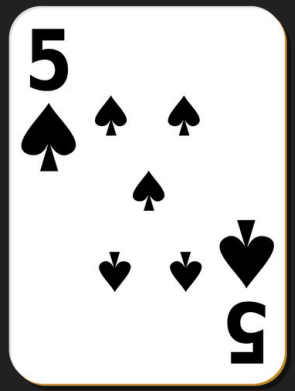


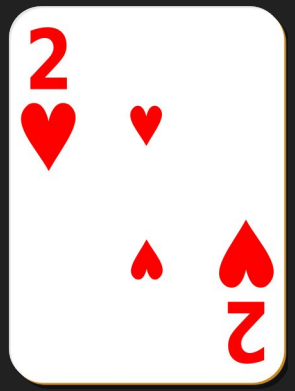
2 < 5? ✓

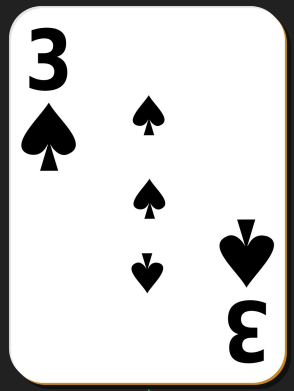
Low card:

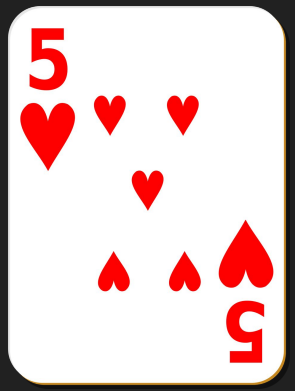


Finding the Lowest Card



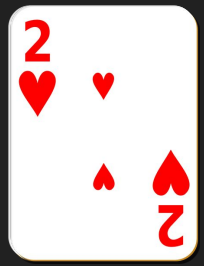




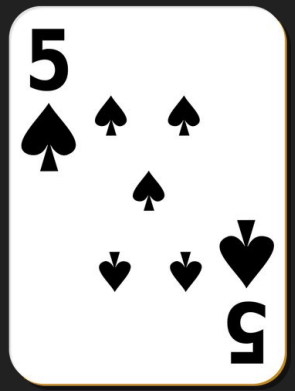


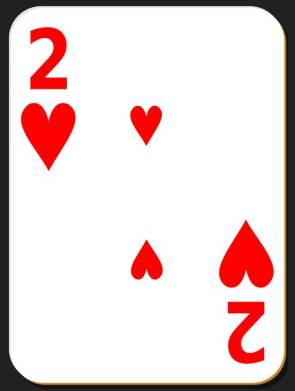
3 < 2?

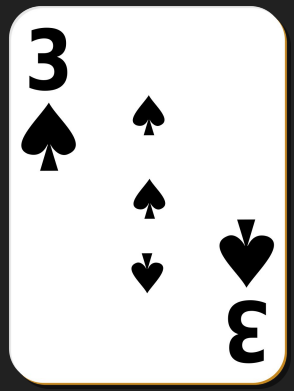
Low card:



Finding the Lowest Card

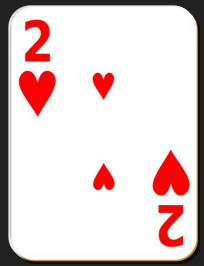






5 < 2? ✗

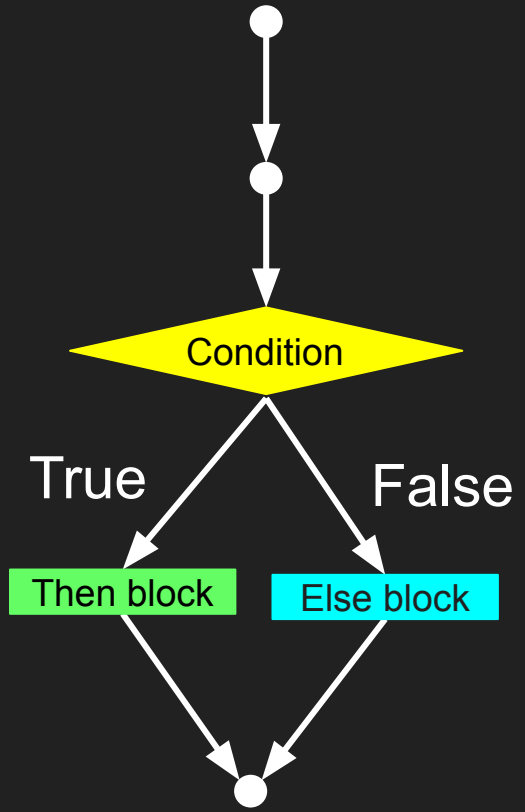
Low card:



Finding the Lowest Card

Finding the low card pseudocode:

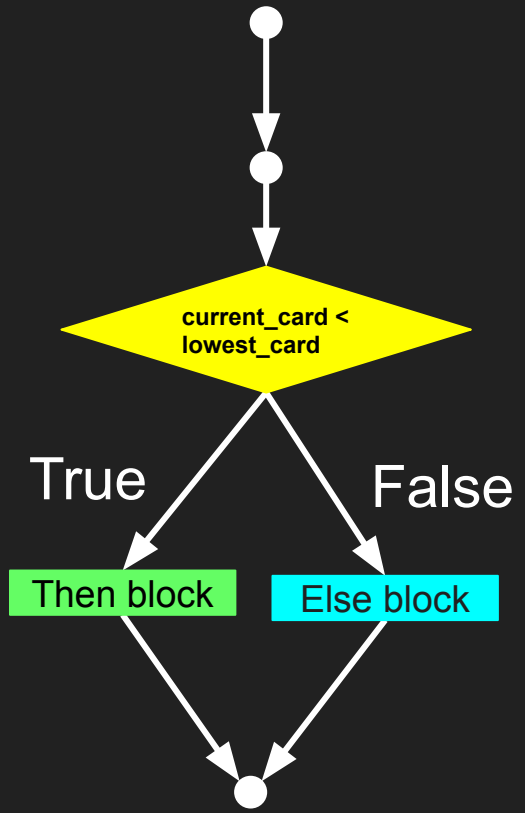
```
1 lowest_card = first card in deck
2 Repeatedly until end of deck:
3     if current_card < lowest_card:
4         lowest_card = current_card
```



Finding the Lowest Card

Finding the low card pseudocode:

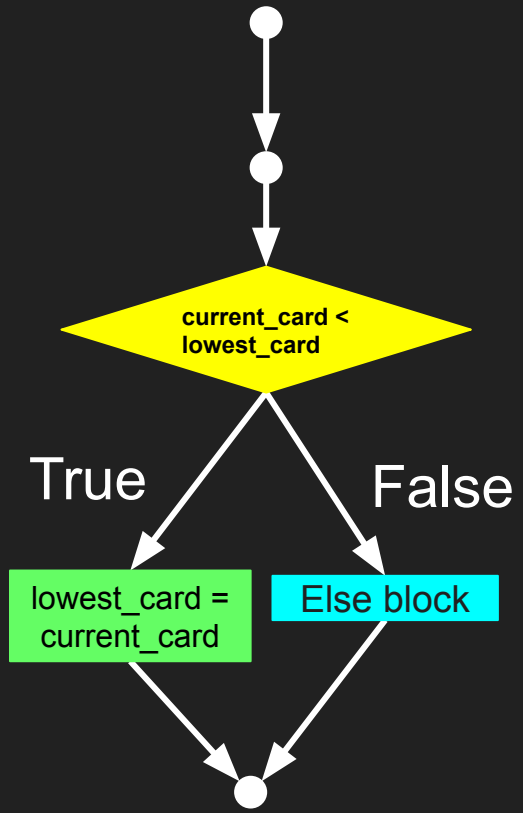
```
1 lowest_card = first card in deck
2 Repeatedly until end of deck:
3     if current_card < lowest_card:
4         lowest_card = current_card
```



Finding the Lowest Card

Finding the low card pseudocode:

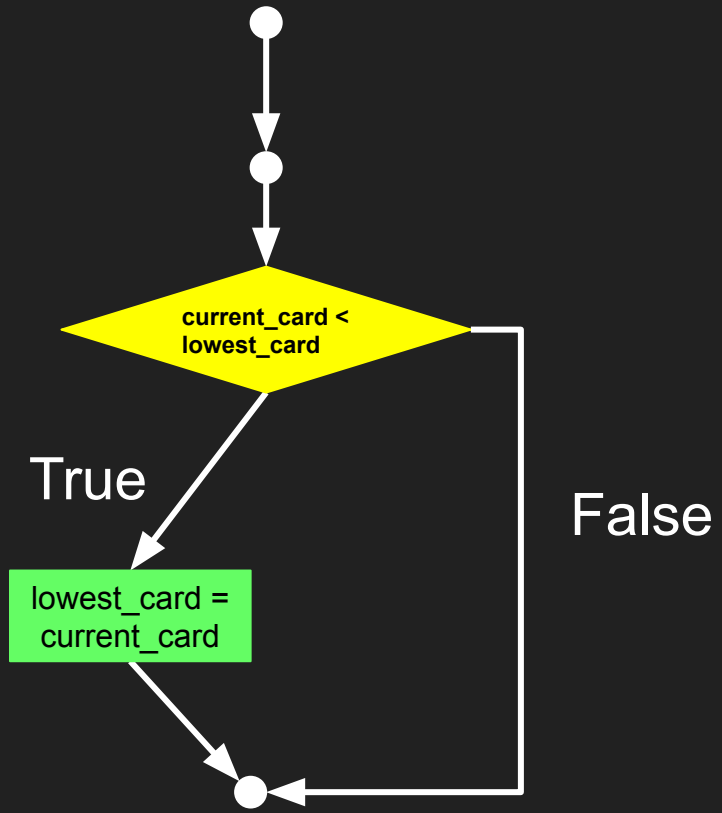
```
1 lowest_card = first card in deck
2 Repeatedly until end of deck:
3     if current_card < lowest_card:
4         lowest_card = current_card
```



Finding the Lowest Card

Finding the low card pseudocode:

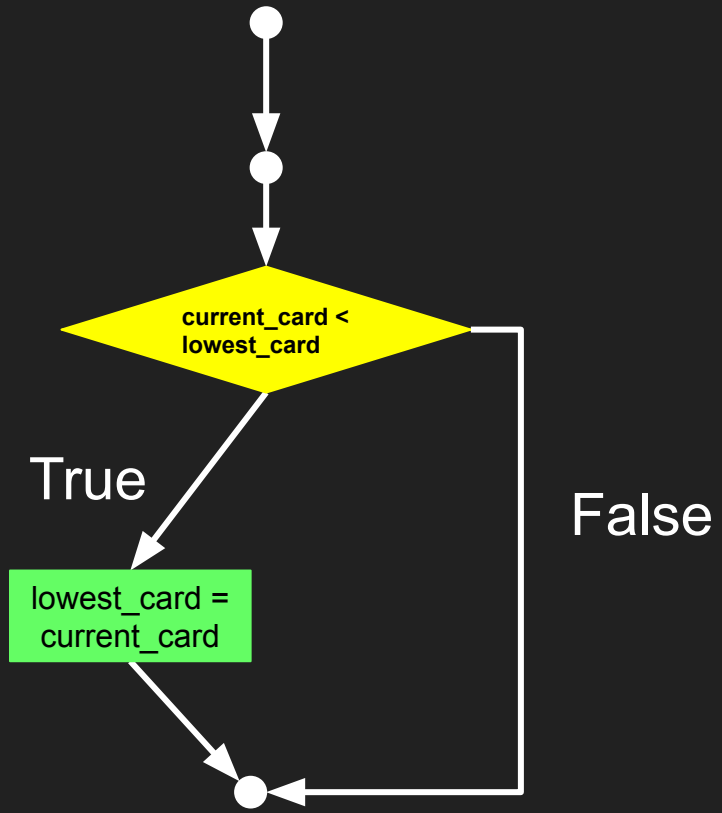
```
1 lowest_card = first card in deck
2 Repeatedly until end of deck:
3     if current_card < lowest_card:
4         lowest_card = current_card
```



Finding the Lowest Card

Finding the low card pseudocode:

```
1 lowest_card = first card in deck
2 Repeatedly until end of deck:
3       if current_card < lowest_card:
4               lowest_card = current_card
```



Loops

- Used to carry out statements in a program repeatedly an arbitrary number of times.

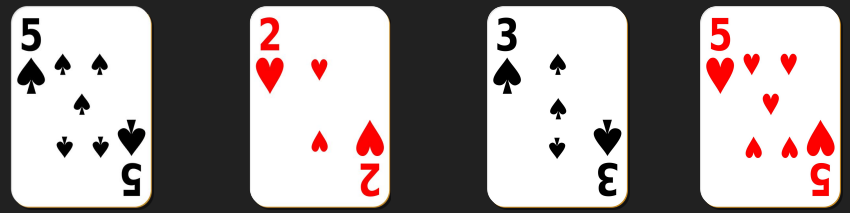
Loops

- Used to carry out statements in a program repeatedly an arbitrary number of times.

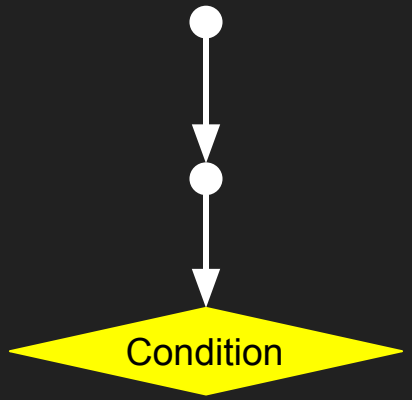
Finding the low card pseudocode:

Loop

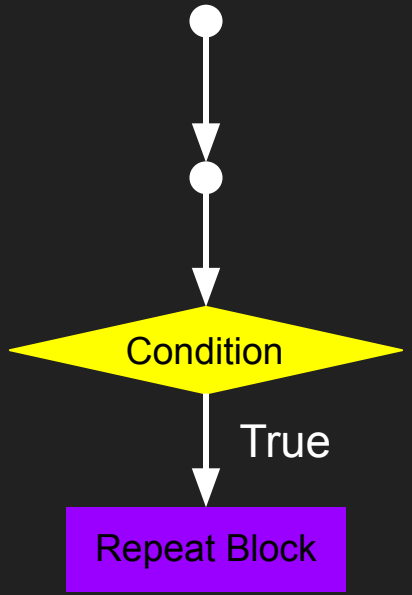
```
1 lowest_card = first card in deck
2 Repeatedly until end of deck:
3       if current_card < lowest_card:
4               lowest_card = current_card
```



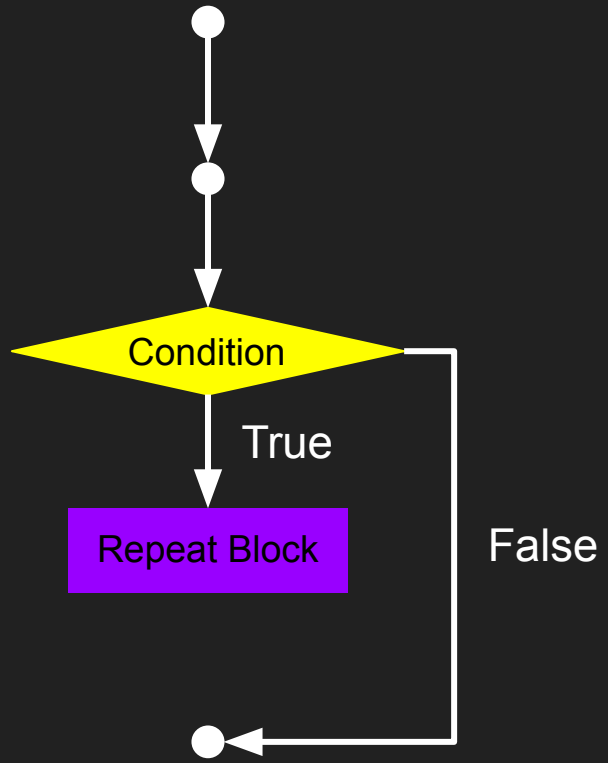
Loops



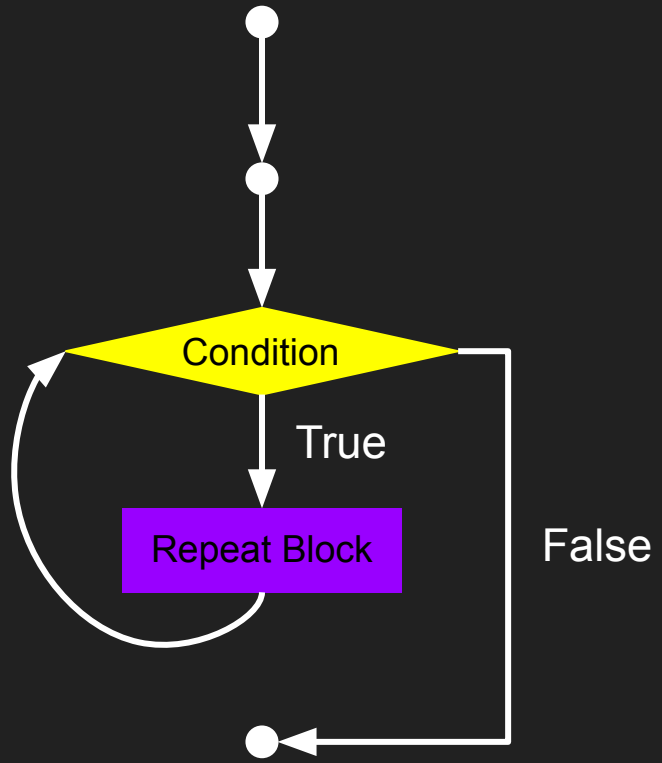
Loops



Loops

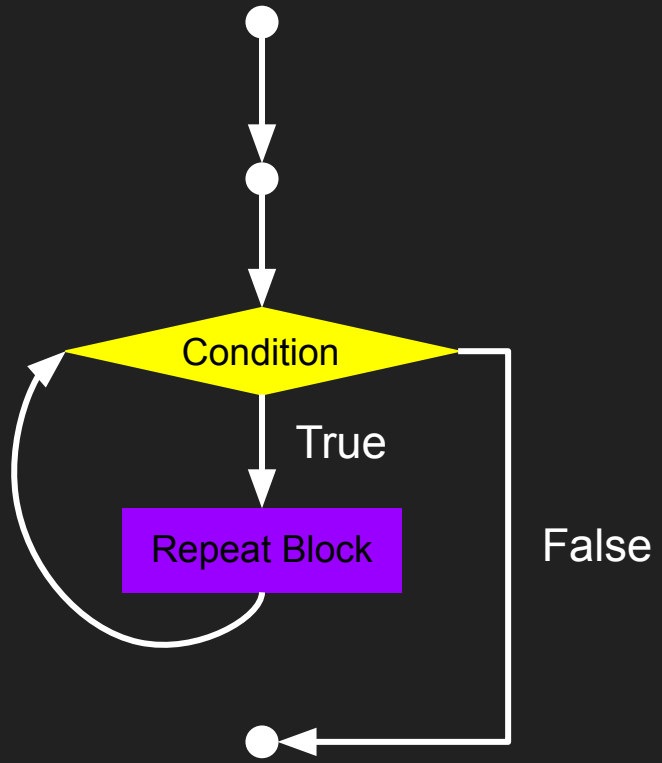


Loops



“While” Loops

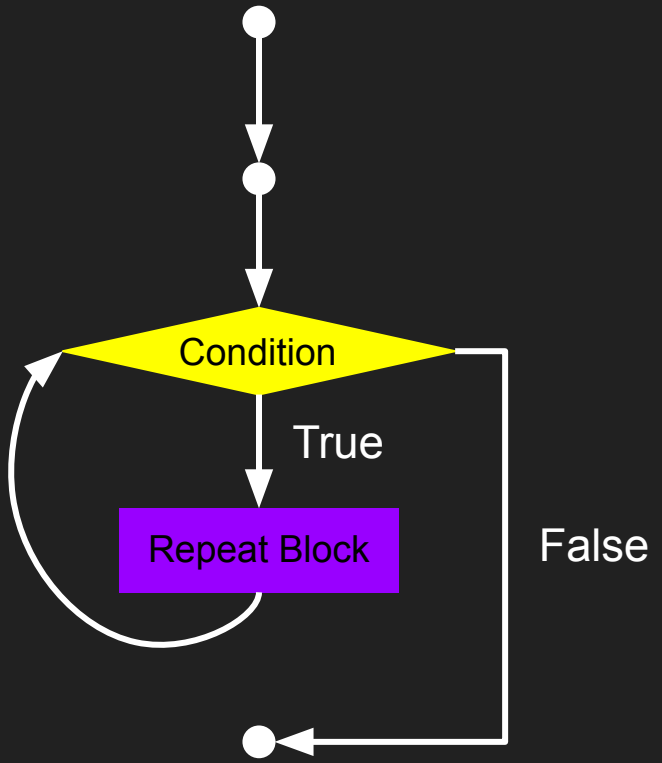
Repeat while condition is true



Loops

Finding the low card pseudocode:

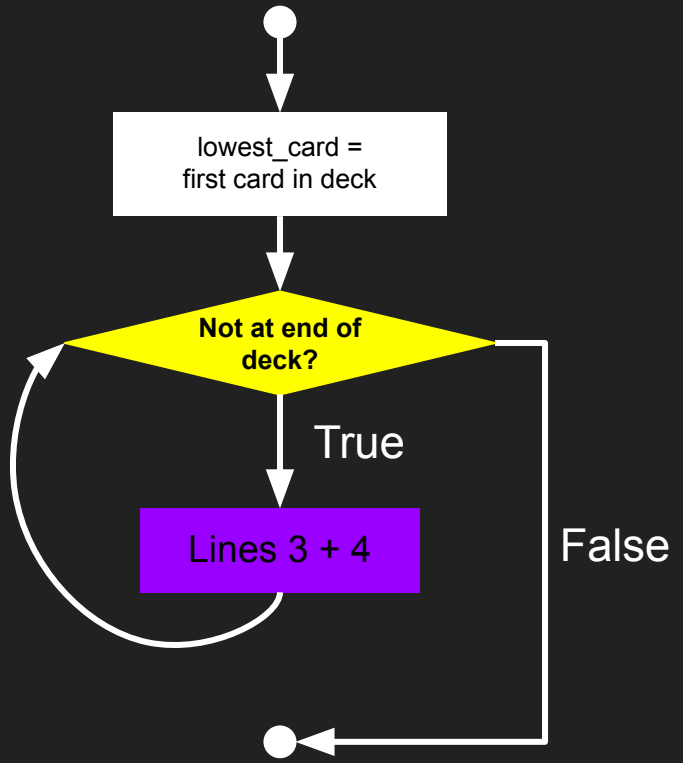
```
1 lowest_card = first card in deck
2 Repeatedly until end of deck:
3       if current_card < lowest_card:
4               lowest_card = current_card
```



Loops

Finding the low card pseudocode:

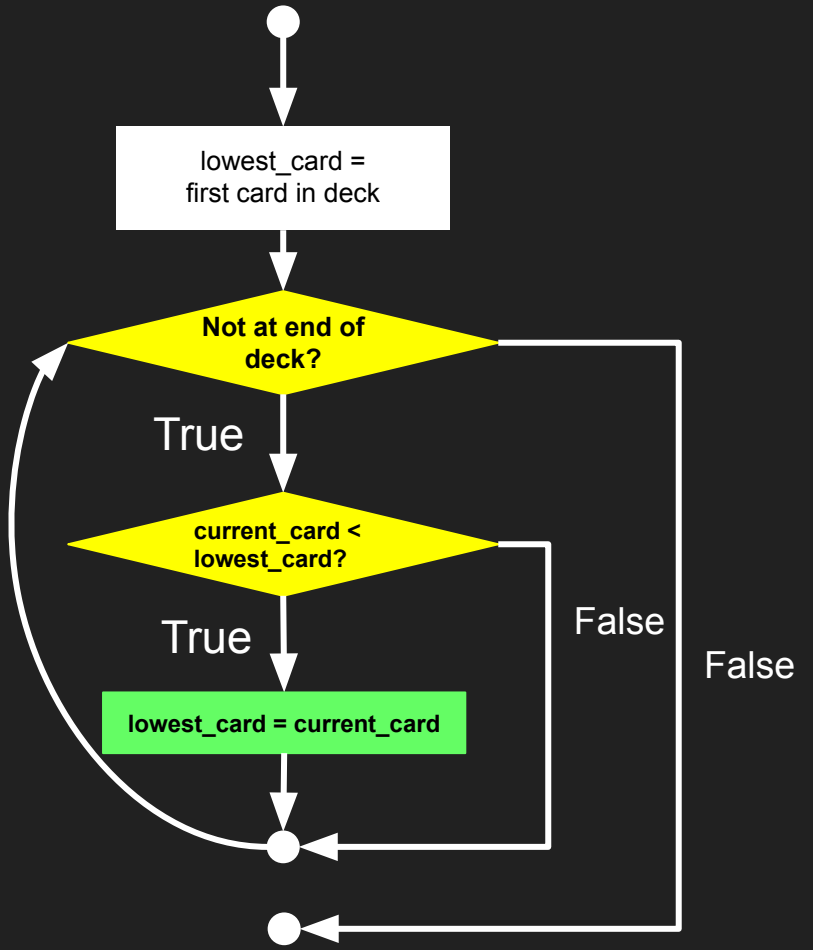
```
1 lowest_card = first card in deck
2 Repeatedly until end of deck:
3       if current_card < lowest_card:
4               lowest_card = current_card
```



Loops

Finding the low card pseudocode:

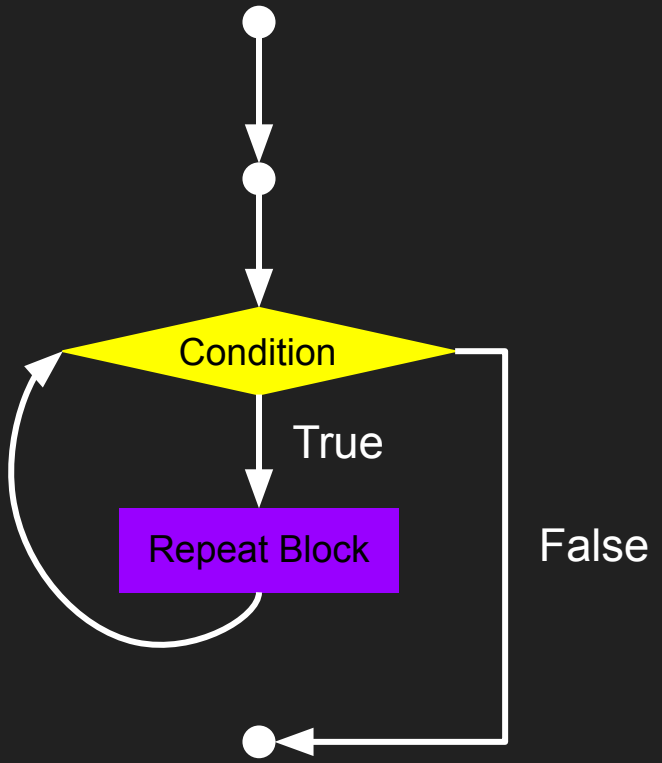
```
1 lowest_card = first card in deck
2 Repeatedly until end of deck:
3       if current_card < lowest_card:
4               lowest_card = current_card
```



Syntax

while <condition>:

    <repeat action>



Practice Memory Diagram

```
1   def loop(stop: int) -> None:
2       condition: bool = True
3       num_loops: int = 0
4       while condition:
5           print(num_loops)
6           num_loops = num_loops + 1
7           if num_loops >= stop:
8               condition = False
9
10  loop(stop=2)
```

Practice Memory Diagram

```
1  def characters(msg: str) -> None:
2      index: int = 0
3      while index < len(msg):
4          print(msg[index])
5          index = index + 1
6
7  characters(msg="Howdy")
```

Practice

Create a function called find_small_card. It should have parameter cards: str and return an int

For example,
find_small_card(cards=“8675309”) should return “0”

Bonus Lesson: Relative Reassignment Operators

Reassigning a variable relative to its current value: i = i+ 1

Addition reassignment operator shorthand has the same effect: i += 1

Since you will use meaningfully descriptive variable names, this is a big improvement!

total_dollars= total_dollars+ next_donation vs total_dollars+= next_donation

```
1  def characters(msg: str) -> None:
2      index: int = 0
3      while index < len(msg):
4          print(msg[index])
5          index = index + 1
6
7  characters(msg="Howdy")
```

```
1  def characters(msg: str) -> None:
2      index: int = 0
3      while index < len(msg):
4          print(msg[index])
5          index += 1
6
7  characters(msg="Howdy")
```

Before	After
`i = i + expr`	`i += expr`
`i = i - expr`	`i -= expr`
`i = i * expr`	`i *= expr`
`i = i / expr`	`i /= expr`
`i = i % expr`	`i %= expr`
`i = i // expr`	`i //= expr`
`i = i ** expr`	`i **= expr`