COMP 110

Boolean Operators and Conditionals

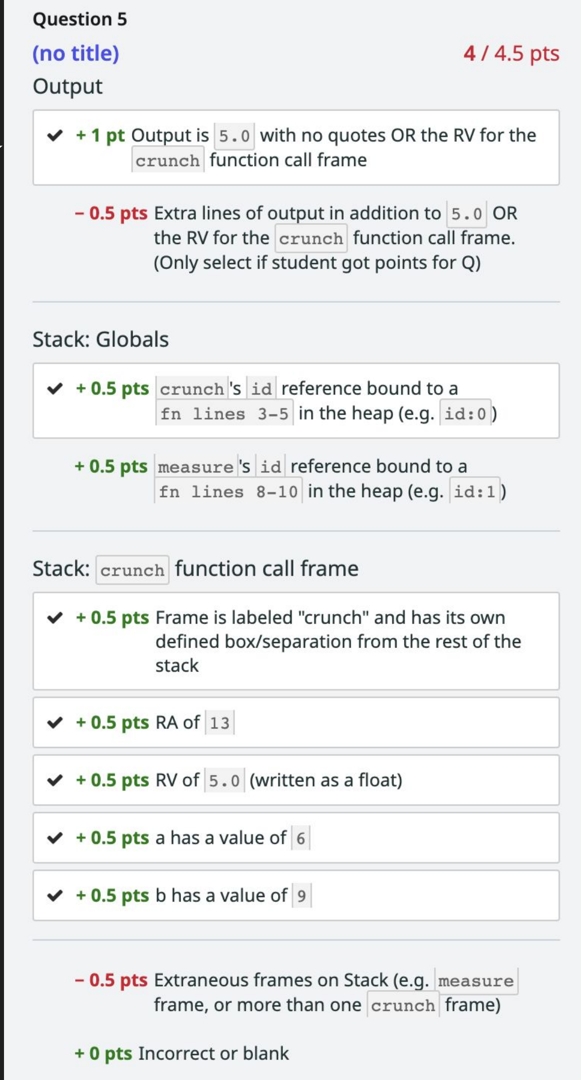
Announcements

Re: Quiz 00

- Graded quizzes will be available on Gradescope soon! Once they're released:
    - *Please review what you missed ASAP*; we will build on the topics covered in Quiz 00 throughout the course, and these foundational concepts are vital!
    - Don't understand a particular question/part of a memory diagram? Please visit us in Office Hours or Tutoring! Full list of hours on the site's support page
    - Please submit a regrade request *if you believe your quiz was not graded correctly according to the rubric*

LS05 and LS06 (multiple choice questions) – due tonight at 11:59pm

Note: You will be able to see the full rubric on Gradescope, but only boxed rubric items with check marks were applied to your quiz



Question 5

(no title) 4 / 4.5 pts

Output

✓ + 1 pt Output is `5.0` with no quotes OR the RV for the `crunch` function call frame

− 0.5 pts Extra lines of output in addition to `5.0` OR the RV for the `crunch` function call frame. (Only select if student got points for Q)

Stack: Globals

✓ + 0.5 pts `crunch`'s `id` reference bound to a `fn lines 3-5` in the heap (e.g. `id:0`)

+ 0.5 pts `measure`'s `id` reference bound to a `fn lines 8-10` in the heap (e.g. `id:1`)

Stack: `crunch` function call frame

✓ + 0.5 pts Frame is labeled "crunch" and has its own defined box/separation from the rest of the stack

✓ + 0.5 pts RA of `13`

✓ + 0.5 pts RV of `5.0` (written as a float)

✓ + 0.5 pts a has a value of `6`

✓ + 0.5 pts b has a value of `9`

− 0.5 pts Extraneous frames on Stack (e.g. `measure` frame, or more than one `crunch` frame)

+ 0 pts Incorrect or blank

Boolean

- Something that evaluates to True or False
- Typically shown with relational operator and/or boolean operator

Boolean

- Something that evaluates to True or False
- Typically shown with relational operator and/or boolean operator

```
“Hello” == “hello”

4 >= 2
```

Boolean

- Something that evaluates to True or False
- Typically shown with relational operator and/or boolean operator

Boolean Operators

- `not, and, or`
- Can be used to express more with booleans
  - It is not rainy: `weather != “rain”`

Boolean Operators

- `not, and, or`
- Can be used to express more with booleans
  - It is not rainy: `not (weather == "rain")`

Boolean Operators

- not, and, or
- Can be used to express more with booleans
  - It is not rainy: `not (weather == “rain”)`
  - It is rainy or it is snowy: `(weather == “rain”) or (weather == “snow”)`

Not

- not inverts the value of a boolean expression

Not

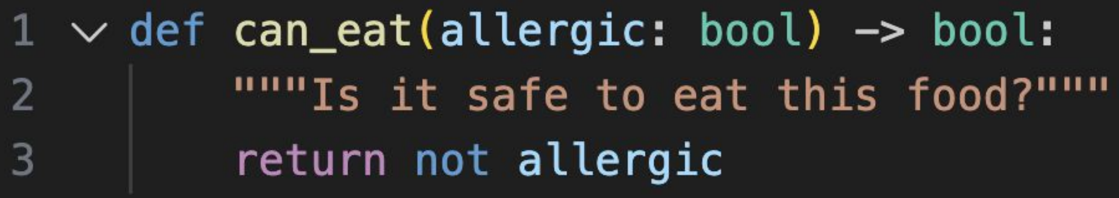
- not inverts the value of a boolean expression

`b`	`not b`
True	False
False	True

Not

- not inverts the value of a boolean expression

`b`	`not b`
True	False
False	True

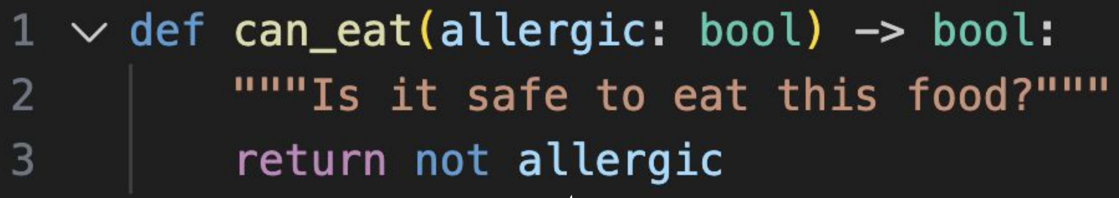


```python
def can_eat(allergic: bool) -> bool:
    """Is it safe to eat this food?"""
    return not allergic
```

Not

- not inverts the value of a boolean expression

`b`	`not b`
True	False
False	True



```python
def can_eat(allergic: bool) -> bool:
    """Is it safe to eat this food?"""
    return not allergic
```

Practice: try calling this function such that you get `False` to print in your terminal

and

- booleans combined with and evaluate to True if and only if
  both booleans are True

`a`	`b`	`a and b`
True	True	True
True	False	False
False	True	False
False	False	False

and

- booleans combined with and evaluate to True if and only if
  both booleans are True

```
def can_eat(allergic: bool, temp: float) -> bool:
    """Is it safe to eat this food?"""
    return not allergic and temp >= 165.0
```

`a`	`b`	`a and b`
True	True	True
True	False	False
False	True	False
False	False	False

and

- booleans combined with and evaluate to True if and only if both booleans are True

```python
def can_eat(allergic: bool, temp: float) -> bool:
    """Is it safe to eat this food?"""
    return not allergic and temp >= 165.0
```

Practice: try calling this function such that you get `False` to print in your terminal. (Challenge: set `allergic=False`)

`a`	`b`	`a and b`
True	True	True
True	False	False
False	True	False
False	False	False

or

- booleans combined with or evaluate to True if at least one is True

`a`	`b`	`a or b`
True	True	True
True	False	True
False	True	True
False	False	False

or

- booleans combined with or evaluate to True if at least one is True

```python
def can_order(got_paid: bool, cost: float) -> bool:
    """Can I afford to eat this?"""
    return got_paid or cost < 5.0
```

`a`	`b`	`a or b`
True	True	True
True	False	True
False	True	True
False	False	False

or

- booleans combined with or evaluate to True if at least one is True

```python
def can_order(got_paid: bool, cost: float) -> bool:
    """Can I afford to eat this?"""
    return got_paid or cost < 5.0
```

Practice: try calling this function such that you get `True` to print in your terminal. What variable values did you use?

`a`	`b`	`a or b`
True	True	True
True	False	True
False	True	True
False	False	False

or

- booleans combined with or evaluate to True if at least one is True

```python
def can_order(got_paid: bool, cost: float) -> bool:
    """Can I afford to eat this?"""
    return got_paid or cost < 5.0
```

Practice: try calling this function such that you get `True` to print in your terminal. What variable values did you use?

Ordering

P

E

MD

AS

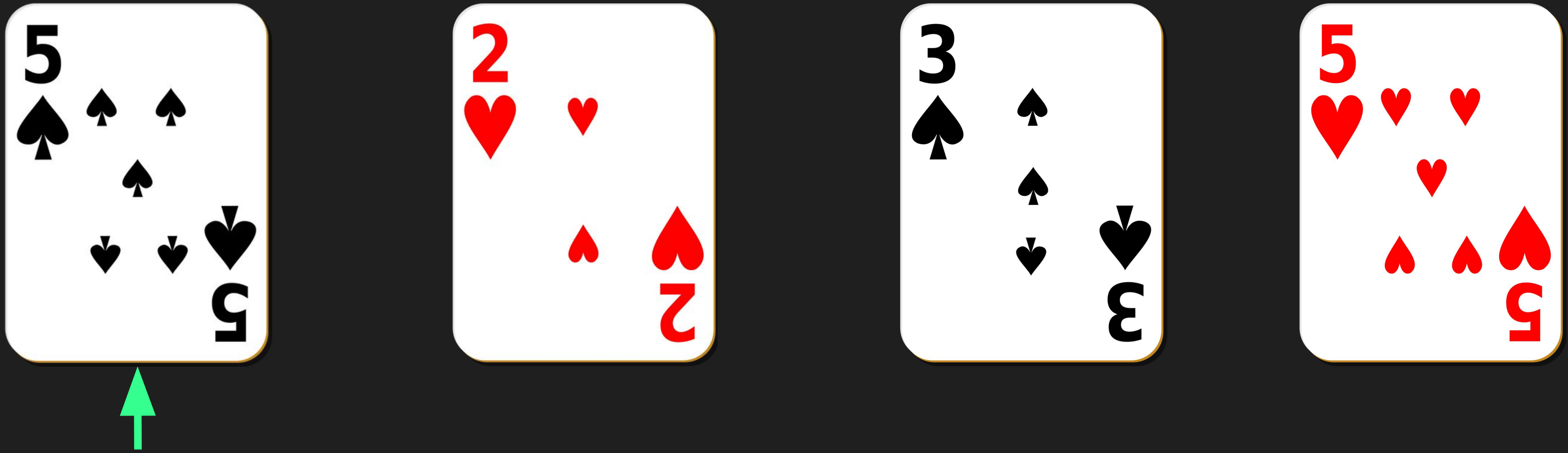
not

and

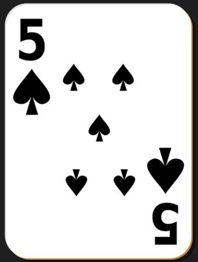
or

Conditionals

Recall: Finding the Lowest Card

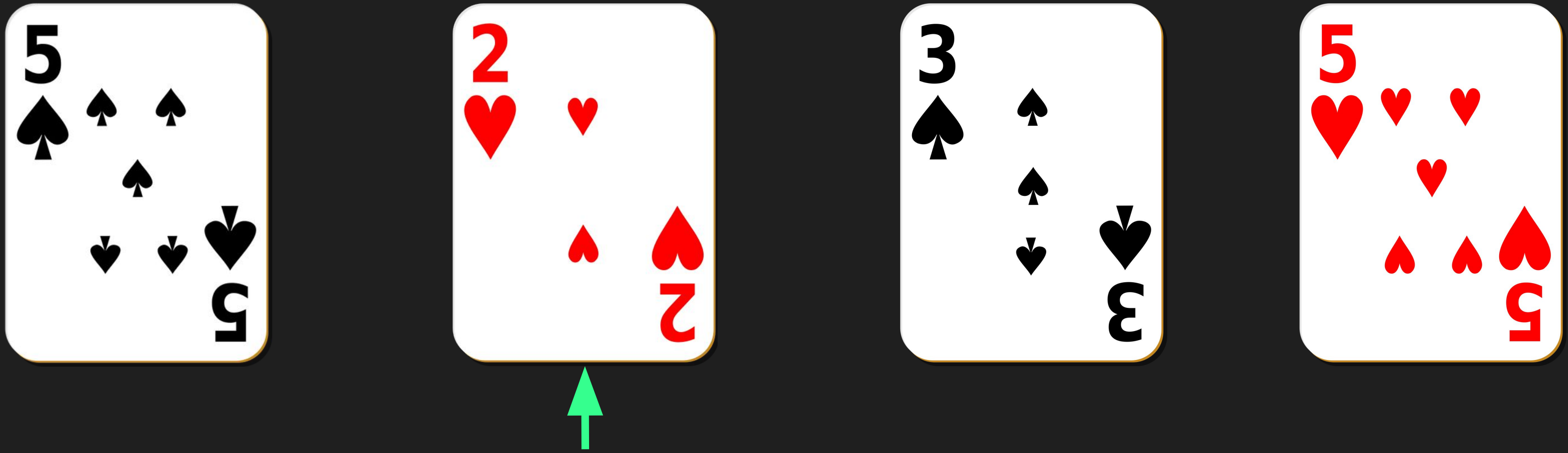


Low card:



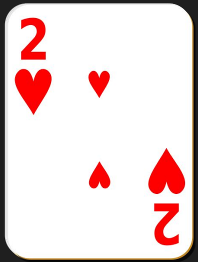
If current card < low card, make it the low card.

Recall: Finding the Lowest Card



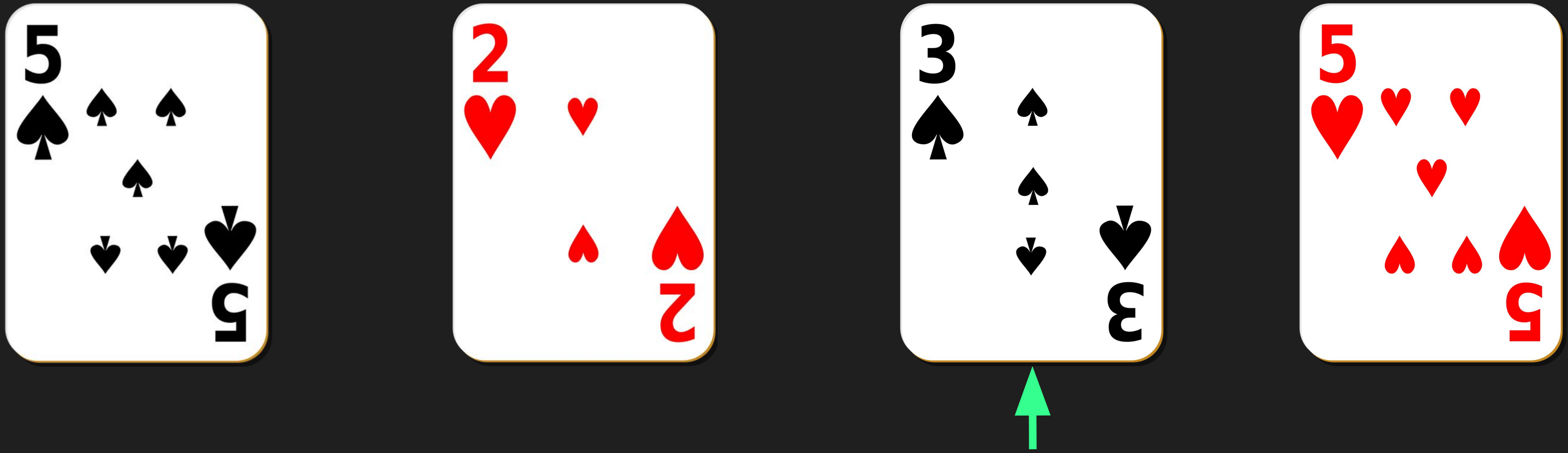
2 < 5? ✓

Low card:



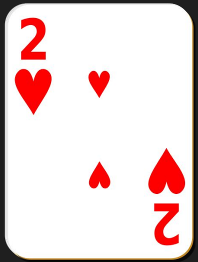
If current card < low card, make it the low card.

Recall: Finding the Lowest Card



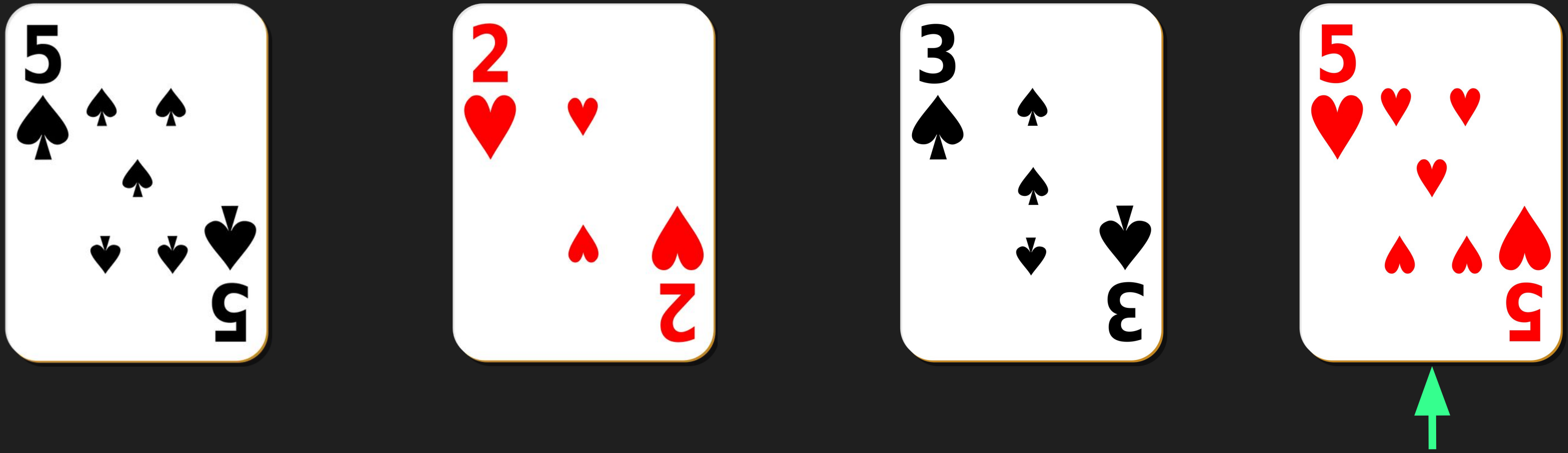
3 < 2? ✗

Low card:

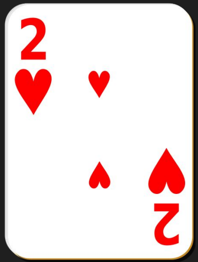


If current card < low card, make it the low card.

Recall: Finding the Lowest Card

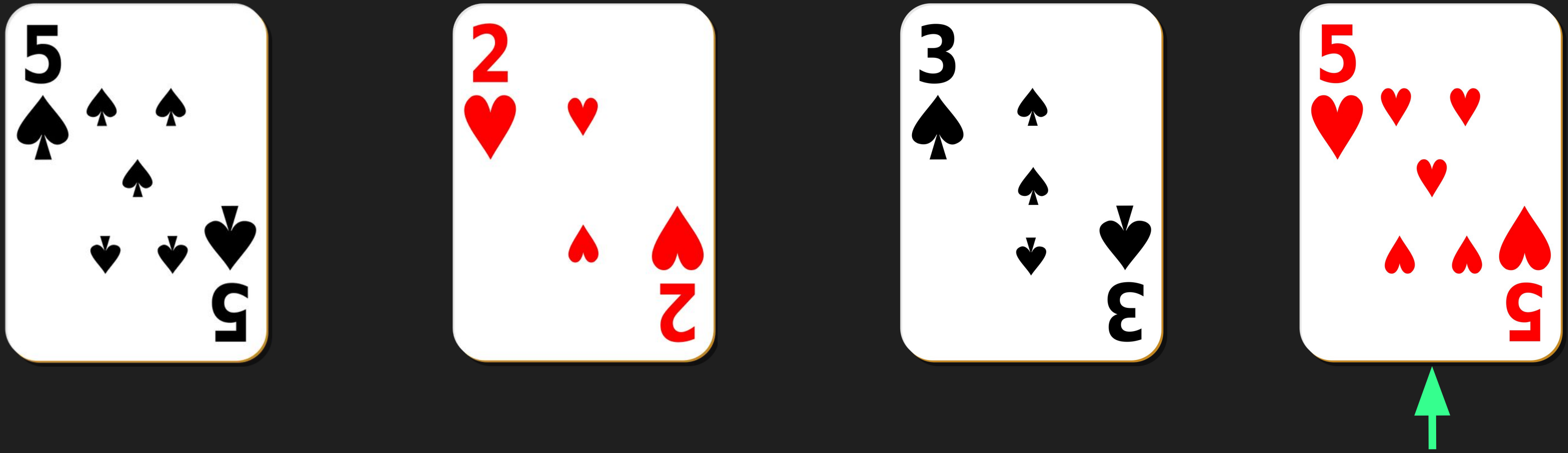


Low card:



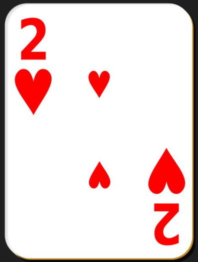
If current card < low card, make it the low card.

Recall: Finding the Lowest Card



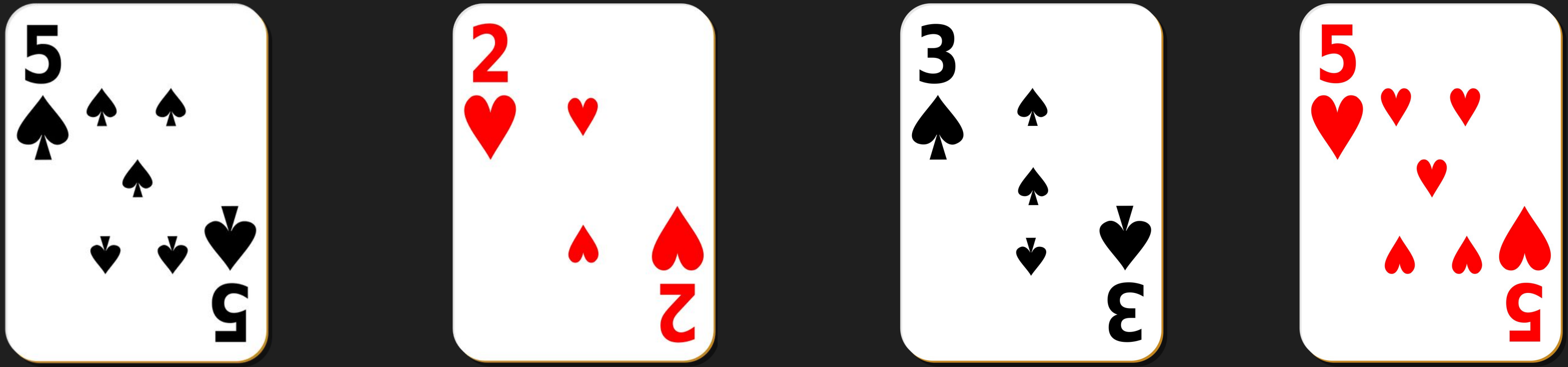
5 < 2? ✗

Low card:



If current card < low card, make it the low card.

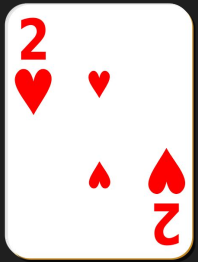
Recall: Finding the Lowest Card



Conditional Statement

If current card < low card, make it the low card.

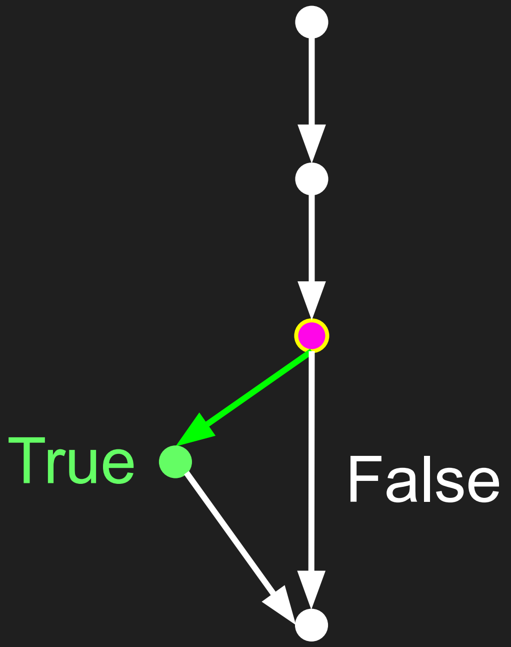
Low card:



Conditional Statements

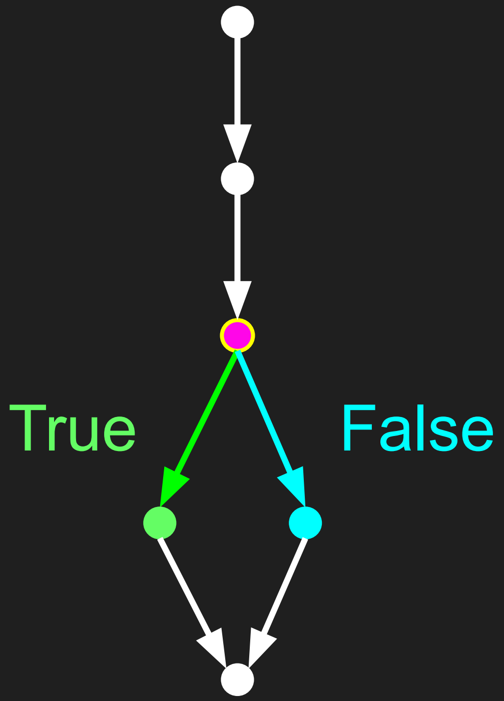
bool

```
if <something>:
    <do something>
<rest of program>
```



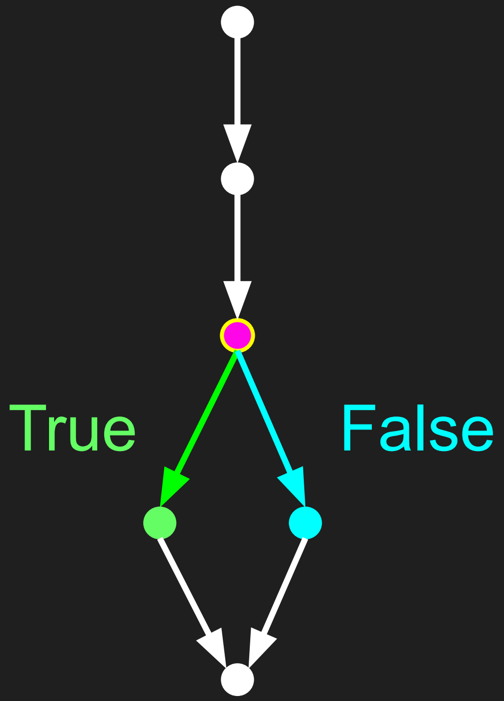
Conditional Statements

```
if <something>:
    <do something>
else:
    <do something else>
<rest of program>
```



Conditional Statements

```
if <something>:
    <do something>
else:
    <do something else>
<rest of program>
```



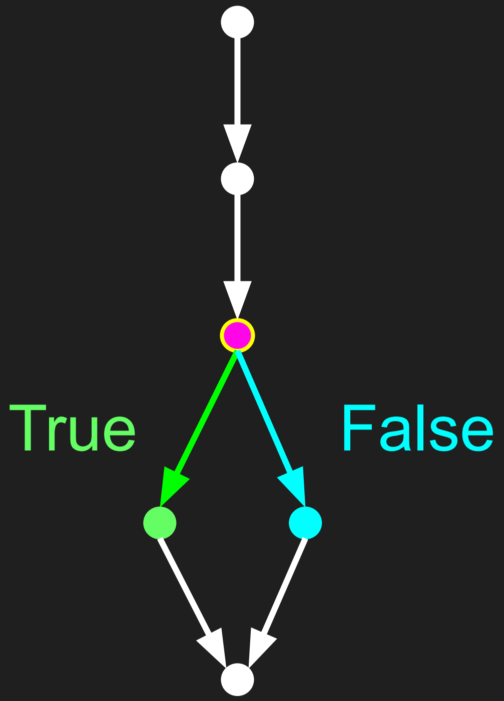
Discussion

What is a decision you make in your day-to-day that you can express as an conditional (if-else) statement?

E.g. If I my assignment is due tomorrow, I start working on it. Else (it’s not due tomorrow), I procrastinate another day.
(This is bad behavior and I don’t condone it!)

Conditional Statements

```
if <something>          :
    <do something>
else:
    <do something else>
<more stuff outside of conditional>
```



Practice

Write a function called check_first_letter that takes a input two strs: word and letter

It should return “match!” if the first character of word is letter

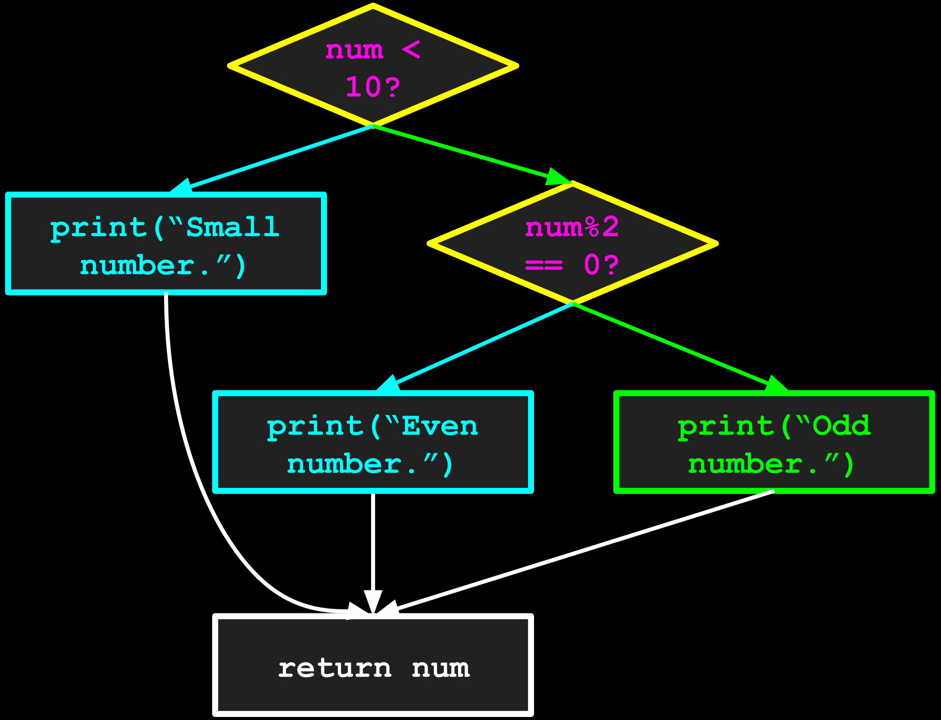
Otherwise, it should return “no match!”

Examples:

- check_first_letter(word=”happy”, letter=”h”) would return “match!”
- check_first_letter(word=”happy”, letter=”s”) would return “no match!”

Drawing the control flow…

```python
def number_info(num: int) -> None:
    if num < 10:
        print("Small number.")
    else:
        if num % 2 == 0:
            print("Even number.")
        else:
            print("Odd number.")
    return num
```

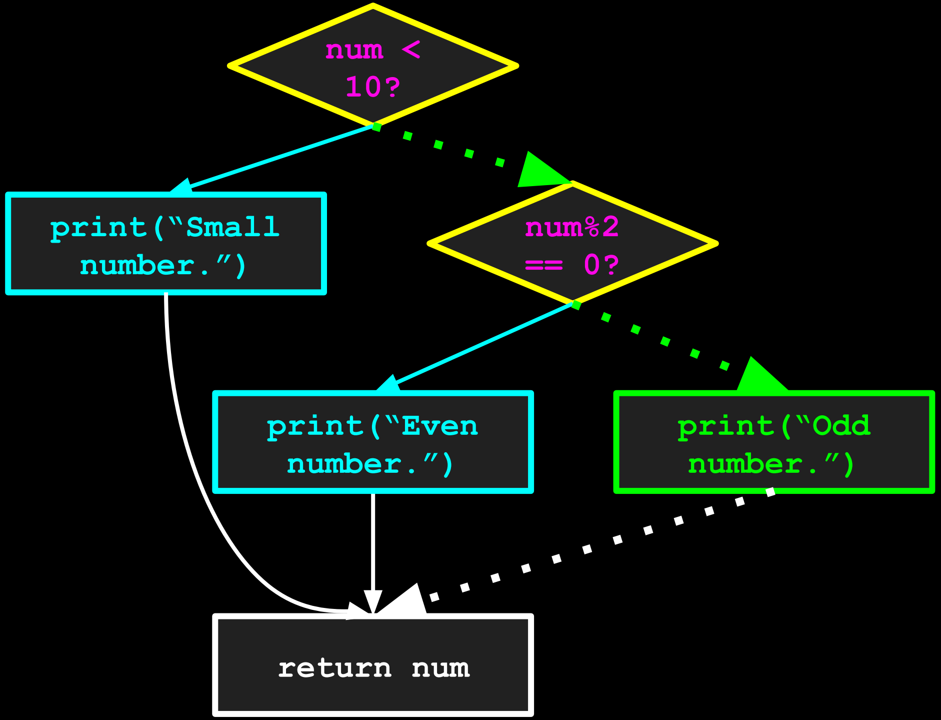


Diagram

```python
def number_info(num: int) -> None:
    if num < 10:
        print("Small number.")
    else:
        if num % 2 == 0:
            print("Even number.")
        else:
            print("Odd number.")
    return num

number_info(num=11)
print(number_info(num=4))
```

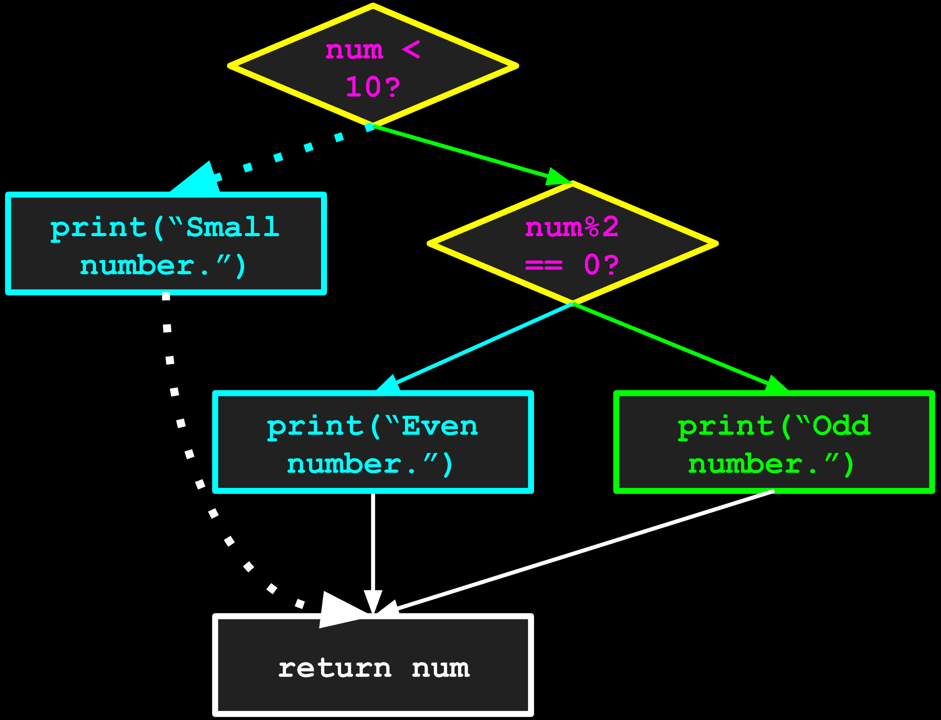
Drawing the control flow…

```
def number_info(num: int) -> None:
    if num < 10:
        print("Small number.")
    else:
        if num % 2 == 0:
            print("Even number.")
        else:
            print("Odd number.")
    return num

number_info(num=11)
print(number_info(num=4))
```



Drawing the control flow…

```
def number_info(num: int) -> None:
    if num < 10:
        print("Small number.")
    else:
        if num % 2 == 0:
            print("Even number.")
        else:
            print("Odd number.")
    return num

number_info(num=11)
print(number_info(num=4))
```



What if…

```python
def number_info(num: int) -> None:
    if num < 10:
        print("Small number.")
    else:
        if num % 2 == 0:
            print("Even number.")
        else:
            print("Odd number.")
    return num
```

What if…

```python
def number_info(num: int) -> None:
    if num < 10:
        print("Small number.")
    else:
        if num % 2 == 0:
            print("Even number.")
        else:
            print("Odd number.")
    return num
```

```python
def number_info(num: int) -> None:
    if num < 10:
        print("Small number.")
    elif num % 2 == 0:
        print("Even number.")
    else:
        print("Odd number.")
    return num
```

What if…

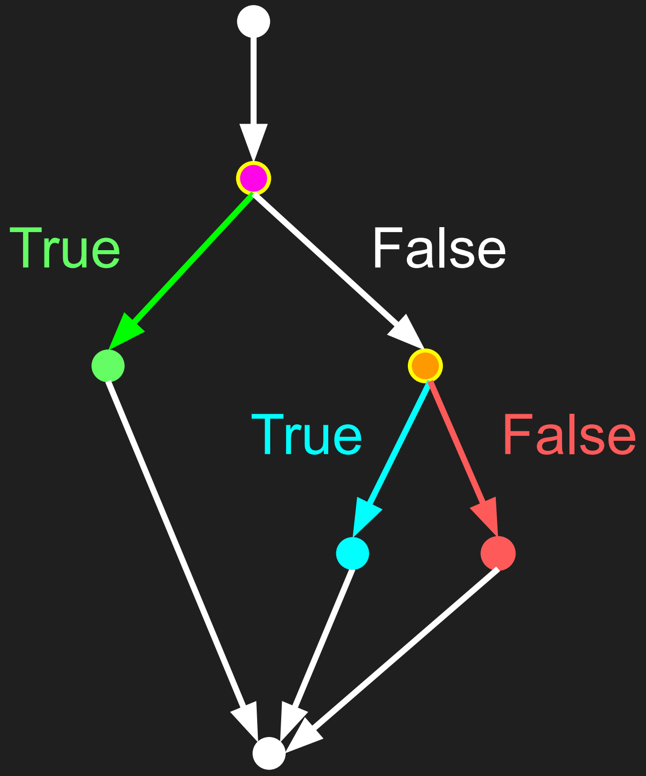
```python
def number_info(num: int) -> None:
    if num < 10:
        print("Small number.")
    else:
        if num % 2 == 0:
            print("Even number.")
        else:
            print("Odd number.")
    return num
```

elif

```python
def number_info(num: int) -> None:
    if num < 10:
        print("Small number.")
    elif num % 2 == 0:
        print("Even number.")
    else:
        print("Odd number.")
    return num
```

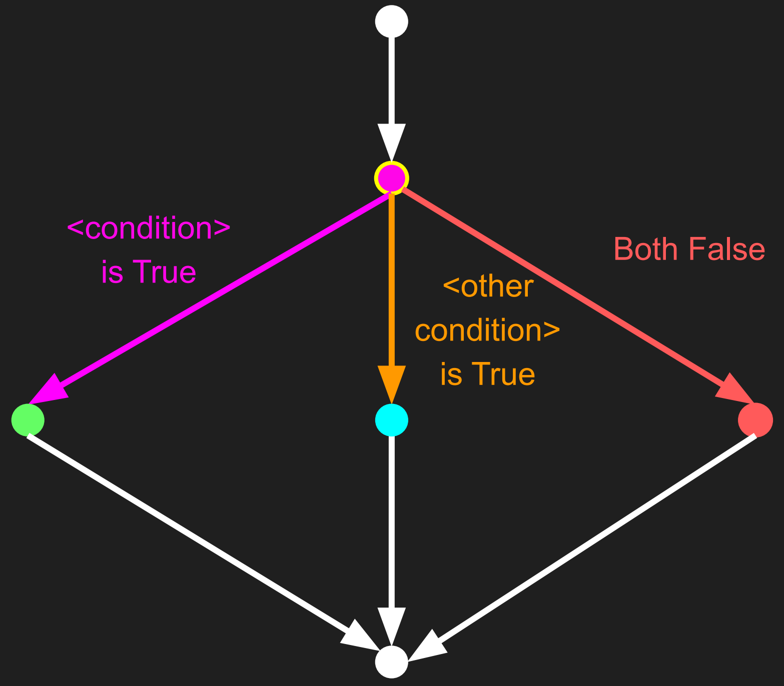
Previous Control Flow

```
if <condition>:
    <do something>
else:
    if <other condition>:
        <do something else>
    else:
        <do third thing>
<rest of program>
```



New Control Flow

```
if <condition>:
    <do something>
elif <other condition>:
    <do something else>
else:
    <do third thing>
<rest of program>
```



Practice

- Write a function called get_weather_report that takes weather: str as input and returns a str
- If weather is "rainy" or "cold", it should print "Bring a jacket!"
- If weather is "hot", it should print "Keep cool out there!"
- Otherwise, it should print "I don't recognize this weather."
- return the weather variable
- Call it with the input “cold” to see what you get!
- Now, use the input function to ask the user “What is the weather?” and pass that as the argument to get_weather_report