UNC TECHNOLOGY, ETHICS & CULTURE IN STOCKHOLM

UNC
COLLEGE OF ARTS AND SCIENCES
Study Abroad

COMP 380
Technology, Ethics, & Culture

May 21 - June 13, 2025

Cece McMahon
mcmahon@cs.unc.edu

More Info & Apply
go.unc.edu/tech-ethics-culture

Info Session at 5pm
on Wednesday, Jan 29
in Fred Brooks (FB) 009

- No prerequisites
- Any major can participate!
- Fulfills the following requirements:
  - Ethical and Civic Values Focus Capacity (FC-Values)
  - High Impact Experience

COMP110

CL06 - Boolean Operators and Conditional Control Flow

Announcements

Re: Quiz 00

- Median grade was 85% – great job!
- Will publish on Gradescope tomorrow
  - *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 come see us in Office Hours/Tutoring!
- *Regrade requests will be open for one week*. Please submit a regrade request if you believe your quiz was not graded correctly according to the rubric

LS06 and LS07 (multiple choice questions) – due tonight at 11:59pm

EX01 – Tea Party Planner – due Tuesday, Jan 28!

Warm-up Questions

Given these two function definitions, reason through the questions below with your neighbors!

```python
"""Warmup question"""


def is_21(age: int) -> bool:
    """Return whether age is at least 21."""
    print("in is_21's function body")
    return age == 21 or age > 21


def birthday(age: int) -> int:
    """Increases age by 1."""
    print("in birthday's function body")
    return age + 1
```

1. Which expression is valid, based on parameter and return type declarations?
   - **a.** `is_21(age=birthday(age=21))`
   - **b.** `birthday(age=is_21(age=21))`
2. For the selected expression above, which function call expression evaluates first?
   - a. Inner-most function call based on parentheses
   - b. Outer-most function call based on parentheses
   - c. First function call encountered, reading from left to right, ignoring parentheses

3. What is the *printed output* of evaluating the following? `is_21(age=21)`

4. What is the *returned value* of evaluating the following? `is_21(age=21)`

Relational Operators (Review)

These operators are placed between expressions of the same type* to compare them.
Relational operators evaluate to *boolean values*.

Operator Symbol	Verbalization	`True` Ex.	`False` Ex.
`==`	Is equal to?	`1 == 1`	`1 == 2`
`!=`	Is NOT equal to?	`1 != 2`	`1 != 1`
`>`	Is greater than?	`1 > 0`	`0 > 1`
`>=`	Is at least?	`1 >= 0` or `1 >= 1`	`0 >= 1`
`<`	Is less than?	`0 < 1`	`1 < 0`
`<=`	Is at most?	`0 <= 1` or `1 <= 1`	`1 <= 0`

*Comparisons between int and float values will automatically convert (“type coerce”) the ints to floats.

Relational Operator Practice

1. 1 + 2 < 3 + 4

Which operator must have higher precedence? < or +?

2. 110.0 != 110

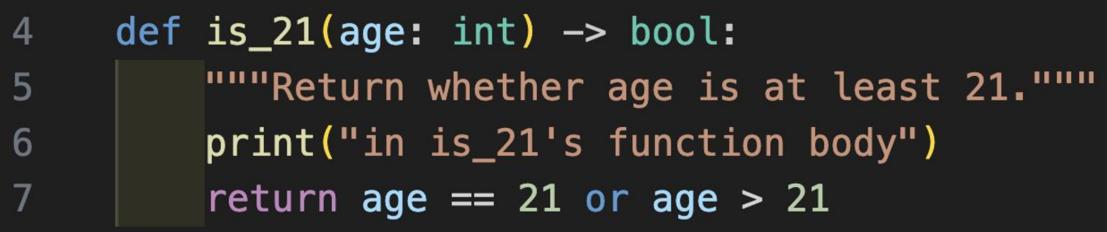
3. “UNC” == “Unc”

Beware of string comparisons! (Read an explanation here.)

4. “UNC” > “DUKE”

Reasoning through the logical `or` operator

Recall the warm-up question…



```
4    def is_21(age: int) -> bool:
5        """Return whether age is at least 21."""
6        print("in is_21's function body")
7        return age == 21 or age > 21
```

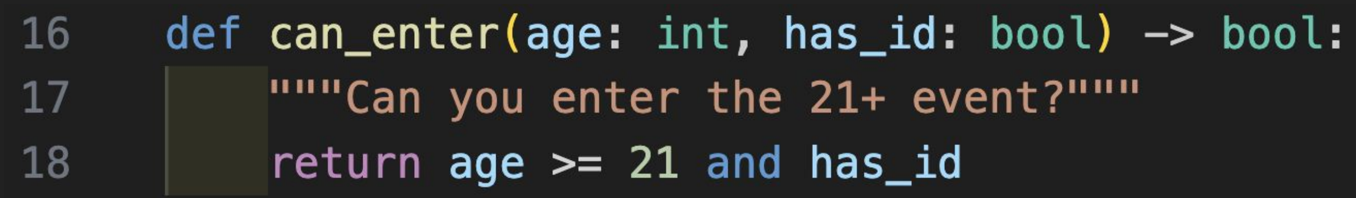
`is_21` returns `True` if age is at least 21, and `False` otherwise. How must the `or` operator work?

How could we rewrite line 7 to simplify it using a different relational operator?

Expression	Evaluated Result
False **or** False	
True **or** False	
False **or** True	
True **or** True	

Reasoning through the logical **and** operator

Consider the function…



```python
def can_enter(age: int, has_id: bool) -> bool:
    """Can you enter the 21+ event?"""
    return age >= 21 and has_id
```

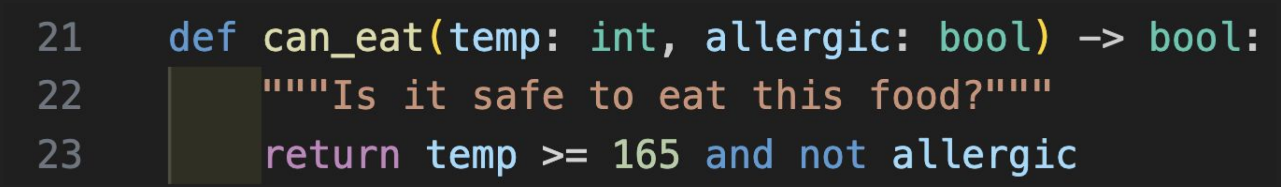
can_enter returns **True** if **age** is at least 21 and **has_id** is **True**, and **False** otherwise. How does the **and** operator work?

Expression	Evaluated Result
False **and** False	
True **and** False	
False **and** True	
True **and** True	

What must have higher precedence:
>= (relational operator), or
and (logical/boolean operator)?

Reasoning through the logical `not` operator

Consider the function…



```python
def can_eat(temp: int, allergic: bool) -> bool:
    """Is it safe to eat this food?"""
    return temp >= 165 and not allergic
```

`can_eat` returns `True` if `temp` is at least 165 and `allergic` is `False`, and `False` otherwise. How does the `not` operator work?

Expression	Evaluated Result
not False	
not True	

For this to be sensible, what must be the precedence of `not`, `and`, and `or`?

Logical / Boolean Operators

Expression	Evaluation
False **or** False	False
True **or** False	True
False **or** True	True
True **or** True	True

Expression	Evaluation
False **and** False	False
True **and** False	False
False **and** True	False
True **and** True	True

Expression	Evaluation
not False	True
not True	False

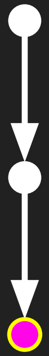
Precedence (highest to lowest):

0. Arithmetic operators (PEMDAS)
1. Relational Operators
2. Not
3. And
4. Or

Conditionals

Control flow is *linear*

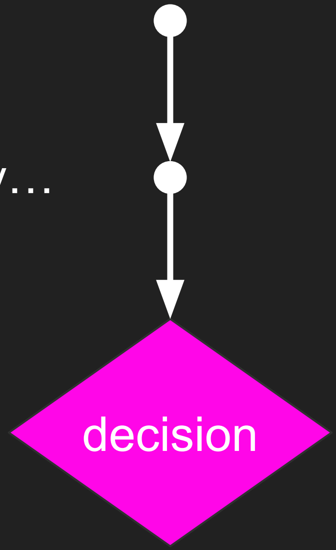
Going about your day…



Control flow is *linear*

Going about your day…

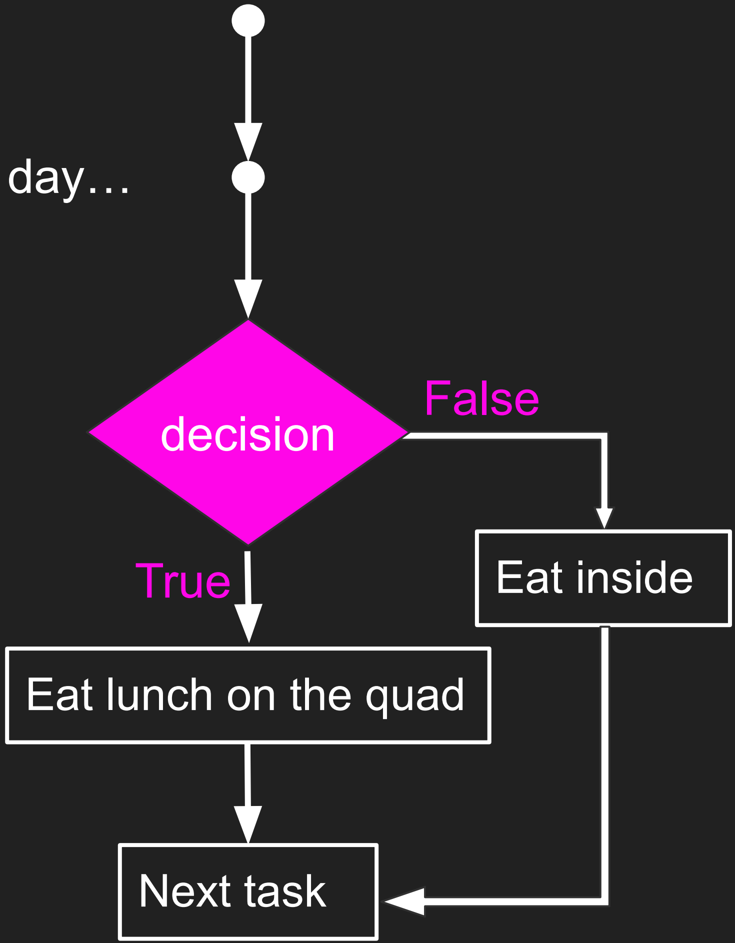
Is the weather nice?



Control flow is *linear*

Going about your day…

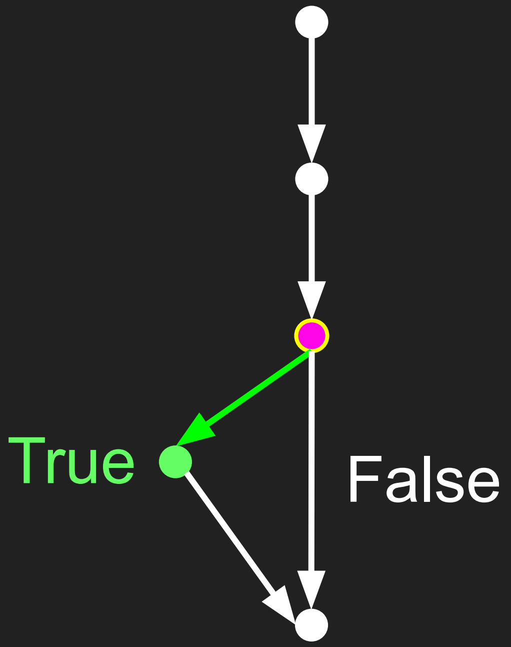
Is the weather nice?



Conditional Statements

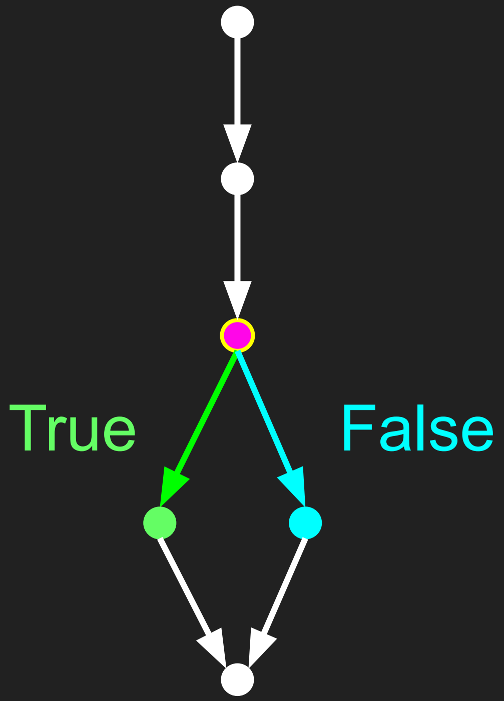
bool

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



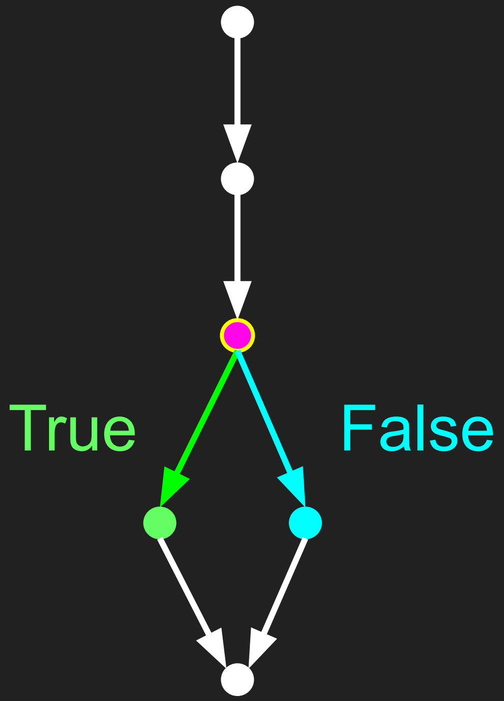
Conditional Statements

```
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?

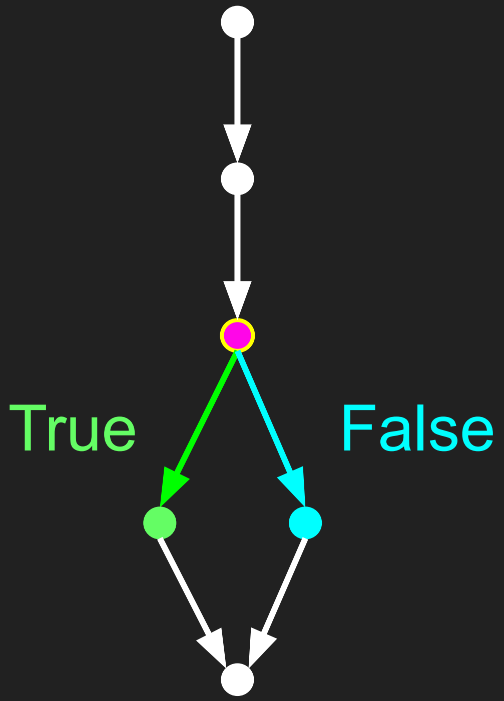
E.g. If I my assignment is due tomorrow, I start working on it. Else (it’s not due tomorrow), I procrastinate another day.
(This is bad behavior and I don’t condone it!)

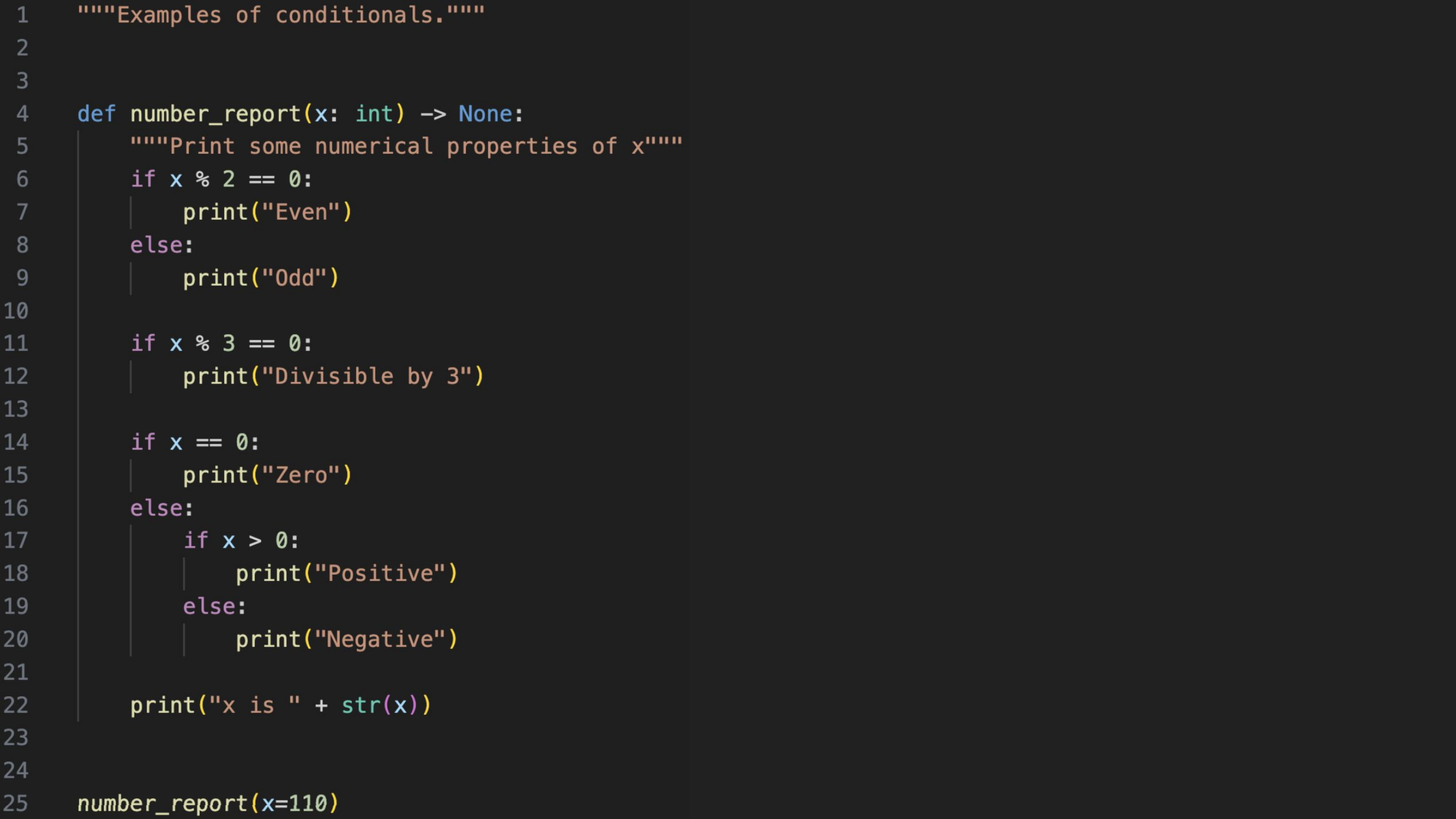
Conditional Statements

```
if                              :

else:

```





```python
"""Examples of conditionals."""


def number_report(x: int) -> None:
    """Print some numerical properties of x"""
    if x % 2 == 0:
        print("Even")
    else:
        print("Odd")

    if x % 3 == 0:
        print("Divisible by 3")

    if x == 0:
        print("Zero")
    else:
        if x > 0:
            print("Positive")
        else:
            print("Negative")

    print("x is " + str(x))


number_report(x=110)
```

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!”