Practice with Conditional Control Flow

Announcements

Re: Quiz 00

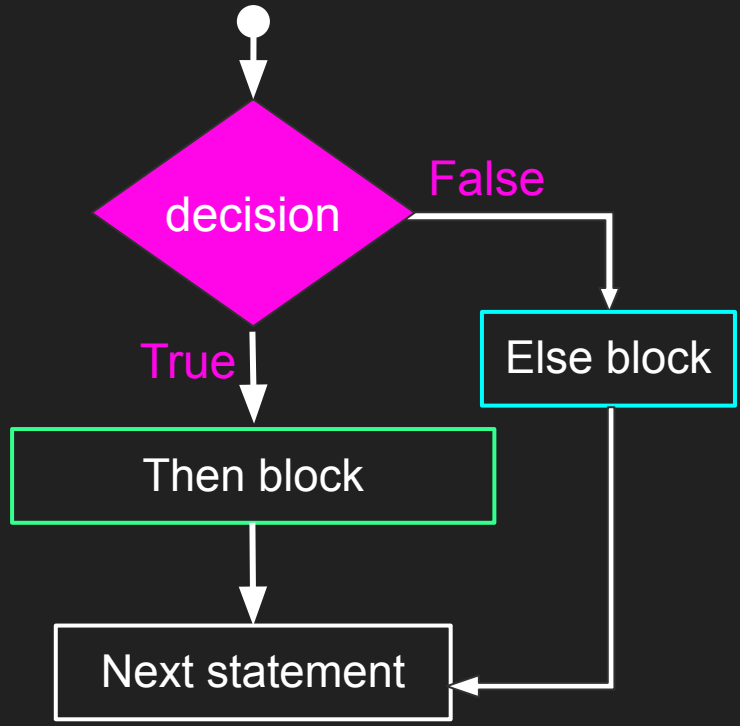
- Don’t understand a particular question/part of a memory diagram? Please come see us in Office Hours or Tutoring!
- *Regrade requests will be open till Wednesday at 11:59pm*. Please submit a regrade request *if you believe your quiz was not graded correctly according to the rubric*

Conditionals Review:
General syntax and semantics

```
if <condition>:
    <then block>
else:
    <else block>
<rest of program>
```

Semantics:

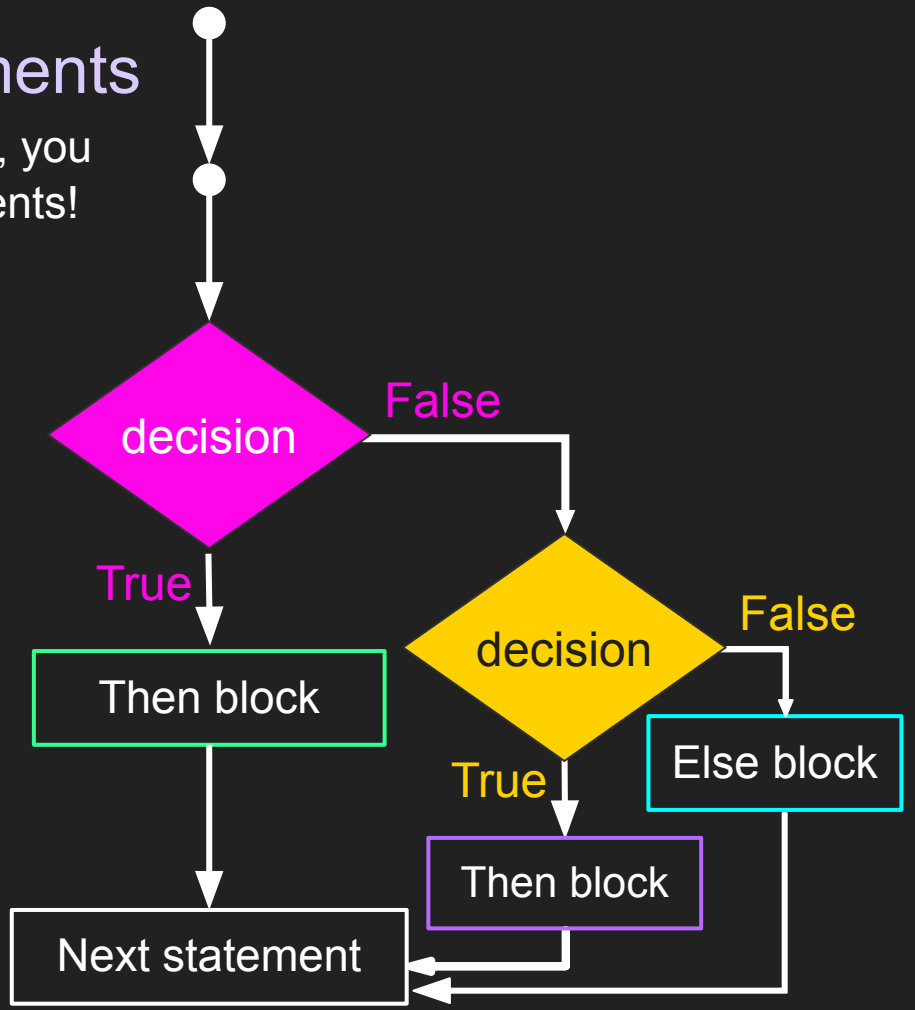
1. When evaluation reaches an **if statement**, the **boolean test expression** is evaluated.
2. If the expression evaluates to `True`, control continues into the **then statement block**. If the then statement block completes without a return, control continues by moving on to the next statement after the if statement.
3. Otherwise, if the test expression evaluates to `False`, control *jumps over the then block* and continues to the next line, whether it is an **else statement block** (if there is one) or the next statement in the program.



If-elif-else / *Conditional* Statements

If you want to test multiple different conditions, you can use one or more **"else if" (`elif`)** statements!

```
if <condition>:
    <then, execute these statements>
elif <different condition>:
    <then, execute these statements>
else:
    <execute these other statements>
<rest of program>
```



Memory diagram

```
"""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)
```

```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)
```

We could eliminate the need for a “nested” **`if-then-else`** statement (inside another conditional’s **`else`** statement) by adjusting this code to use an **`elif`** statement. How?

Practice

Write a function called `check_first_letter` that takes as input two `str`s, named `word` and `letter`

It should behave as follows:

- If `letter`'s value *doesn't* contain exactly one character, return `"letter's argument should be one character!"`
- If the first character of `word` is the same as `letter`, return `"match!"`
- Otherwise, 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!"`
- `check_first_letter(word="happy", letter="ha")` would return `"letter's argument should be one character!"`

```python
"""Calling to and fro..."""


def ping(i: int) -> int:
    print("ping: " + str(i))
    if i <= 0:
        return i
    else:
        return pong(i=i - 1)


def pong(i: int) -> int:
    print("pong: " + str(i))
    return ping(i=i - 1)


print(ping(i=2))
```