Conditional execution

Presenter: Steve Baskauf
steve.baskauf@vanderbilt.edu

DISC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

Jean & Alexander Heard LIBRARIES

CodeGraf landing page

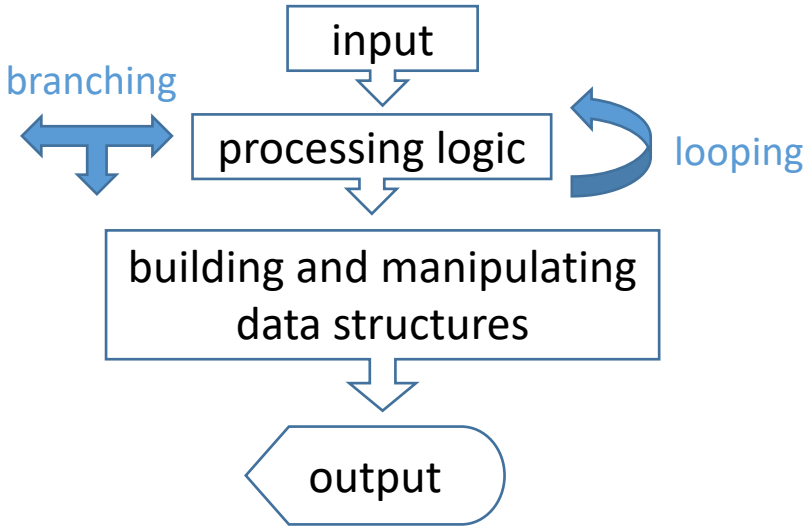
- vanderbi.lt/codegraf

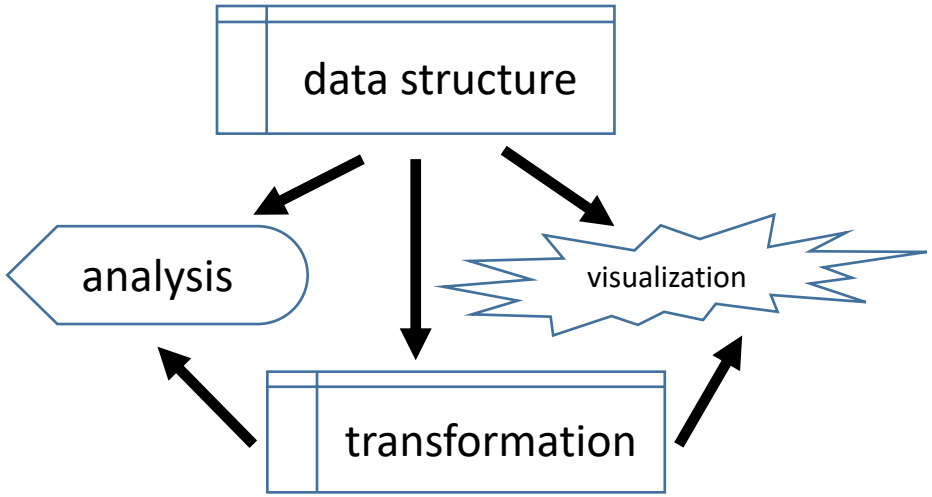
Procedural vs. vectorized paradigm

Jean & Alexander Heard
LIBRARIES

Procedural vs. vectorized programming

beginner **Python** advanced





advanced **R** beginner

`if` … `else` … statements

Jean & Alexander Heard
LIBRARIES

Conditional execution

```
is_friday = False

if is_friday:
    print('Woopie! ')
    print('TGIF !')
```

- The **`if`** statement evaluates a **boolean**
- If **`True`**, the following **indented code block** is executed. (Don't forget colon!).
- Notice how I named the boolean variable to make the code readable.

Conditional execution

```
name = input('What is the name of the character? ')
is_micky = name == 'Mickey Mouse'
print(name)
print(is_micky)

if is_micky:
    print('You are a Disney character')
print('That is all!')
```

- The comparison operator (==) is different from the assignment operator (=) and produces a boolean.
- The indented **`print`** statement is only executed if the condition is **`True`**.
- The non-indented **`print`** statement is always executed.
- Indentation is **super important** in Python!

`else` and `elif`

- **`else`** defines the default code block if no condition is satisfied.
- **`elif`** combines **`else`** and **`if`**; use to check additional conditions.
- Python does NOT have the **`switch-case`** structure common in other languages.

Note:

- Notice how indentation is used to control which code blocks are conditionally executed and which ones are always executed.
- Notice that the program is really dumb. It only does what you say and doesn't really have any idea what a Disney character is.

Flags

DISC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

Jean & Alexander Heard LIBRARIES

What are flags for?

- A **flag** is a variable that we use to keep track of the state of some condition in our code.
- Flags often contain boolean **`True`** or **`False`** values.
- Clever naming allows us to write readable code when testing a condition:

```
if door_open:
    result = close(door)
```

Error trapping

Jean & Alexander Heard
LIBRARIES

`try...except...` for error trapping

```
try:
    code that might throw an error goes here
except:
    code to be executed if there's an error goes here
here's where the code execution continues
```

- Error trapping handles problems gracefully instead of having the script crash.
- An error is called an exception.
- Code blocks are identified by indentation (as usual)
- Colons required after **`try`** and **`except`**

`try…except…` for error trapping

- Example:

```
from math import pi
typed_in = input('What is the diameter of your circle? ')

try:
    diameter = float(typed_in)
    print('The circumference is:', diameter * pi)
except:
    print("Sorry, you didn't enter a number.")
```

- It's a good idea to error trap any error that can be predicted to happen sometime (e.g. file not found)