Errors & exceptions

Divide by zero

Read a file that is not present

Accessing an invalid index/key in list/dictionary

Often useful to "trap" error in the program and take corrective action

Exception handling

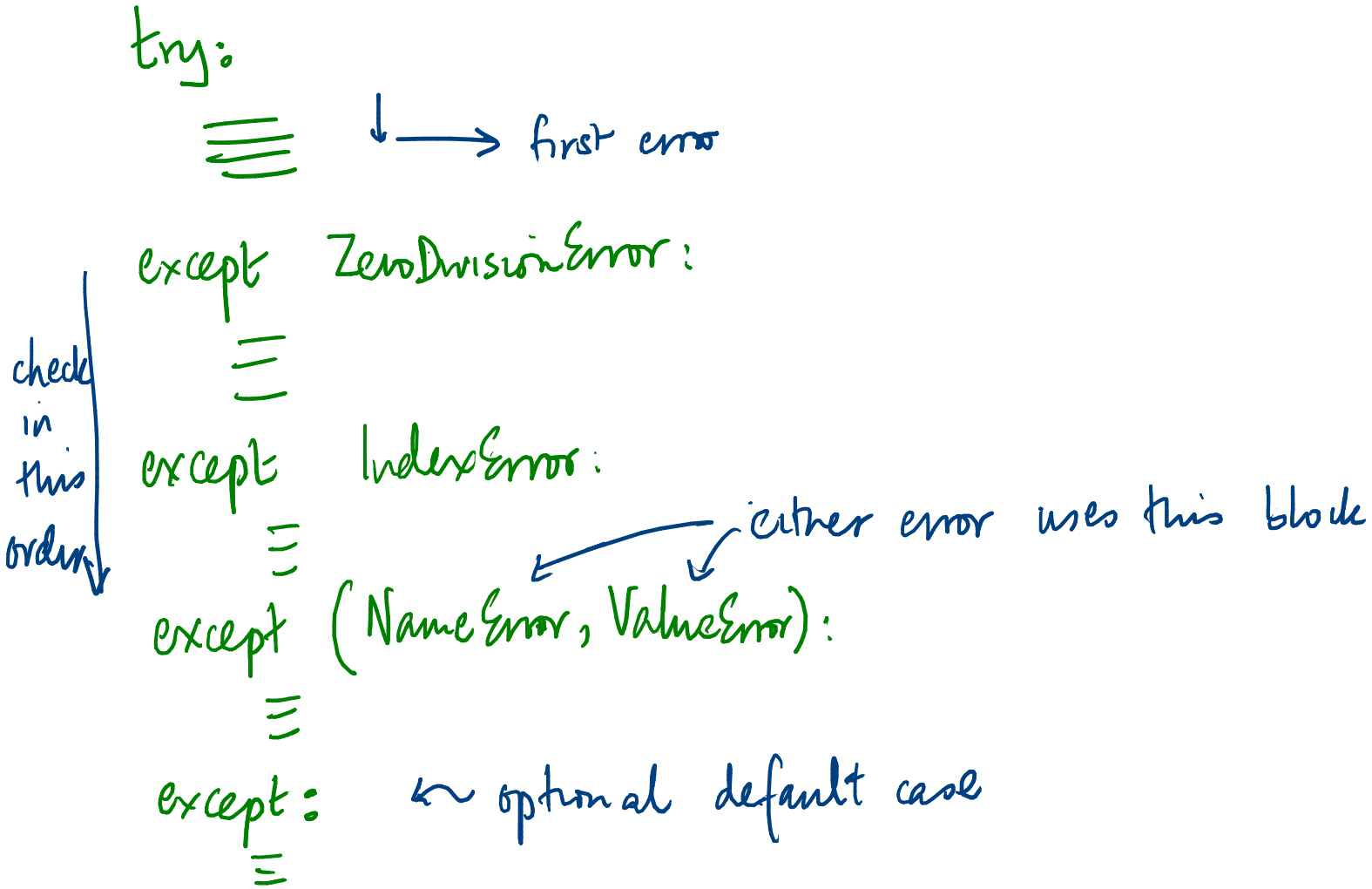
```
try:
    ===   } code that might generate error
except:
    ===   } recovery code
```

Can name the errors you want to deal with

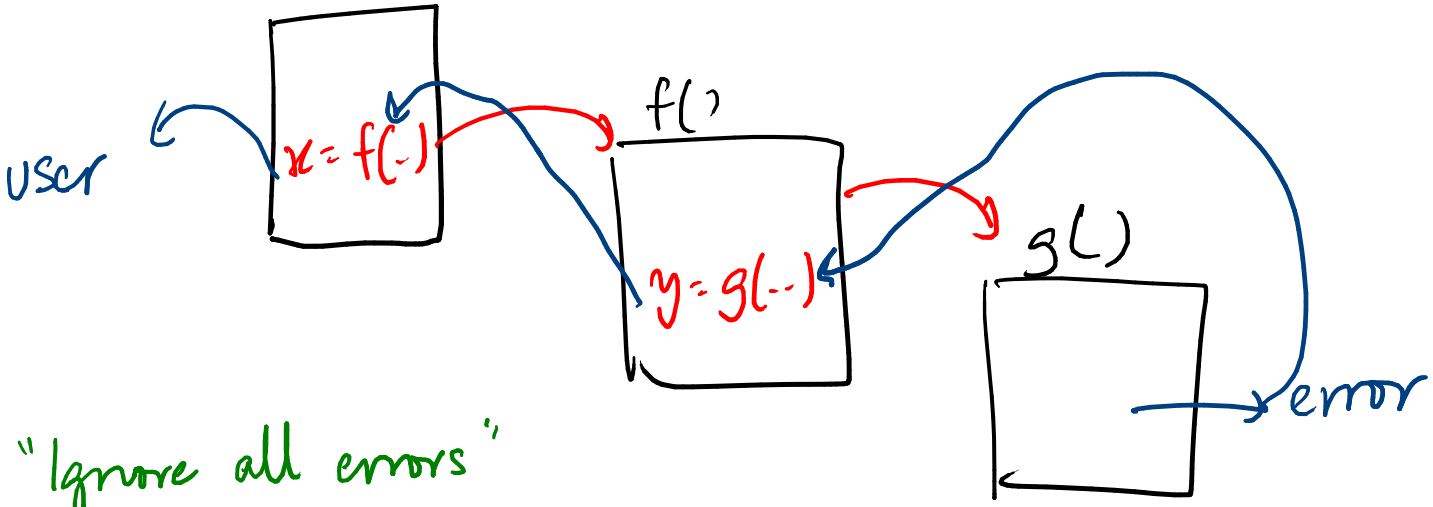
```
try:
    average = total/n   — if n is 0, abort & check except
except ZeroDivisionError:        does not return to
    "take corrective action"     top of try
```

ZeroDivisionError
IndexError — list index
KeyError - dictionary key
NameError - undefined value

Multiple exceptions



If exception/error is not "handled", passed back to code that called it



"Ignore all errors"

```
try:
    ==
except:
    pass
```

← "do nothing"

Can also generate errors

"if x outside $[0, 100]$, ValueError"

```
if x < 0 or x > 100:
    raise (ValueError)
```

Can also define new types of errors

Can be used in a "positive" way

```
if k in count.keys():
    count[k] = count[k]+1
else:
    count[k] = 1
```

Instead

```
try:
    count[k] = count[k]+1
except KeyError: # k not a key
    count[k] = 1
```

Input/output

Read from keyboard — input()

Print to screen — print()

x = input() → reads a line of input into x

Always a string!

Need to use int(), float() etc to convert to a numeric value

x = int(input())

Want to "prompt" user for input

```
x = input("Please enter an integer")
```

Add symbols like : , ␣ space

Output to screen

```
print(x)
```

Implicitly calls str(x) to convert to string

Variable # argument

```
print("The coordinates are", x, "and", y)
```

Can add special chars

\t tab

\n newline

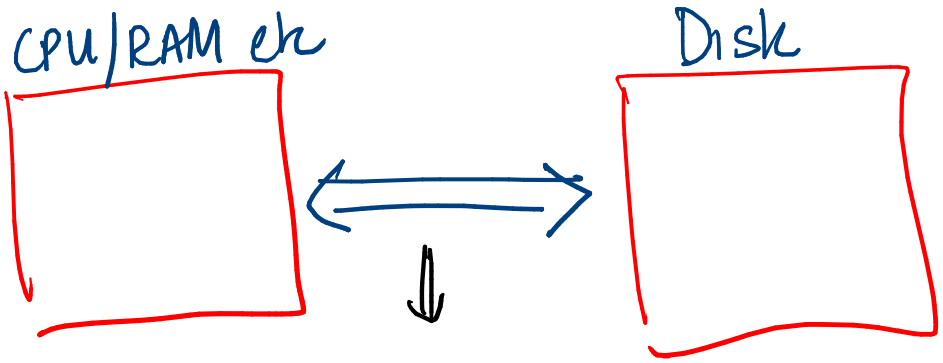
Automatic newline after last arg

" spaces between args

For further fine tuning — "formatted printing"

later, if ever

Dealing with files



Blocks

Buffers – collect data & write when block is full

Reading/writing a file is an indirect process

Program deals with file handles

- like any other variable in your program

Opening & closing files

- Open: Associates a specific file with a file handle
- Close: Removes association

Open a file - "attach" it to a file handle fh

Read/write via fh

Close file

Files are sequential text file

3 "modes" to open a file

read, write, append

↓

creates/overwrites