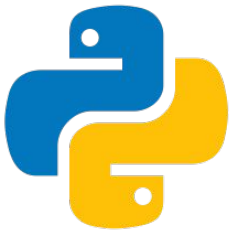
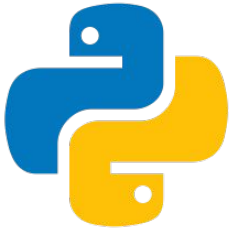
Chapter 3: Imperative Programming

Resource: Introduction to Computing Using Python by Ljubomir Perkovic



Topics to cover:

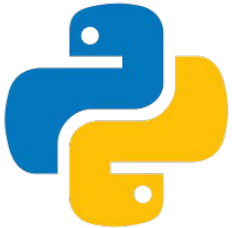
- First Python program
- 3.2 Execution Control Structures
  - One-Way Decisions
  - Two-Way Decisions
  - for loops
  - Range
- 3.3 User defined functions
- 3.4 Variable and Assignments
- 3.5 Parameter Passing
-



First Python Program

Create file **hello.py**

```
line1 = 'Hello Python developer...'
line2 = 'Welcome to the world of Python!'
print(line1)
print(line2)
```



Closer look

Hello.py = module

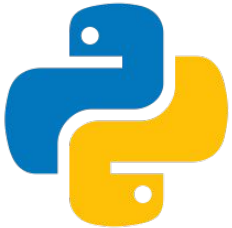
Module = file containing python code that ends in **.py** (math.py is a module)

Built-in functions:

- **print()** - prints

Create input.py

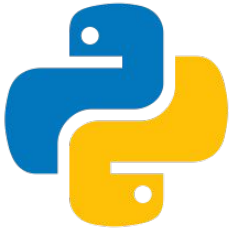
```
first = input('Enter your first name: ')
last = input('Enter your last name: ')
line1 = 'Hello '+ first + ' ' + last + '...'
print(line1)
print('Welcome to the world of Python!')
```



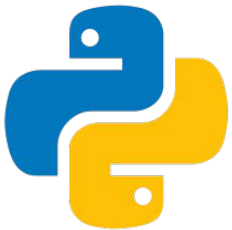
Closer look

Built-in function:

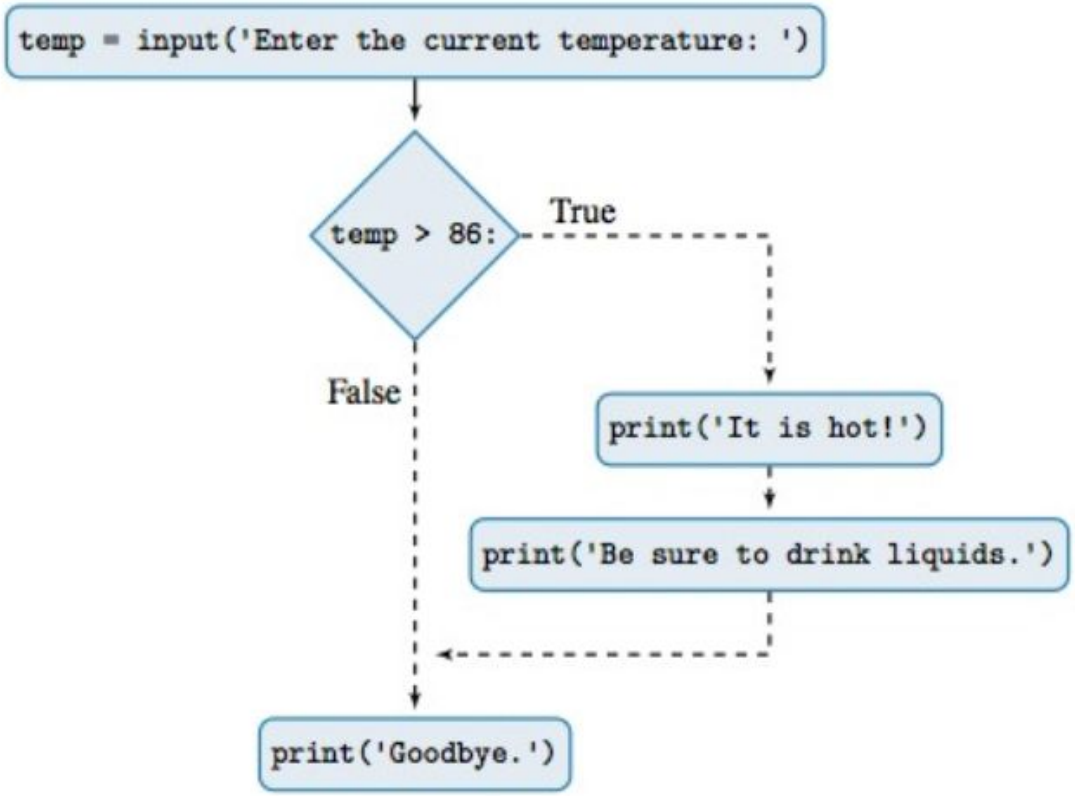
- **input()** - prints input argument and waits for user to type something and hit `Enter` or `Return`
  - Returns string **always**
- **eval()** - evaluates string as if it were a python expression
  - Can use this with input to return more than just strings

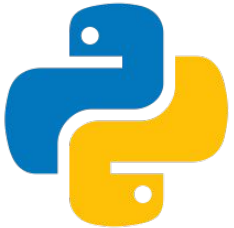


Execution Control Structures



One-Way Decisions

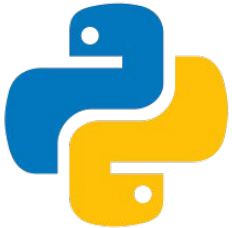




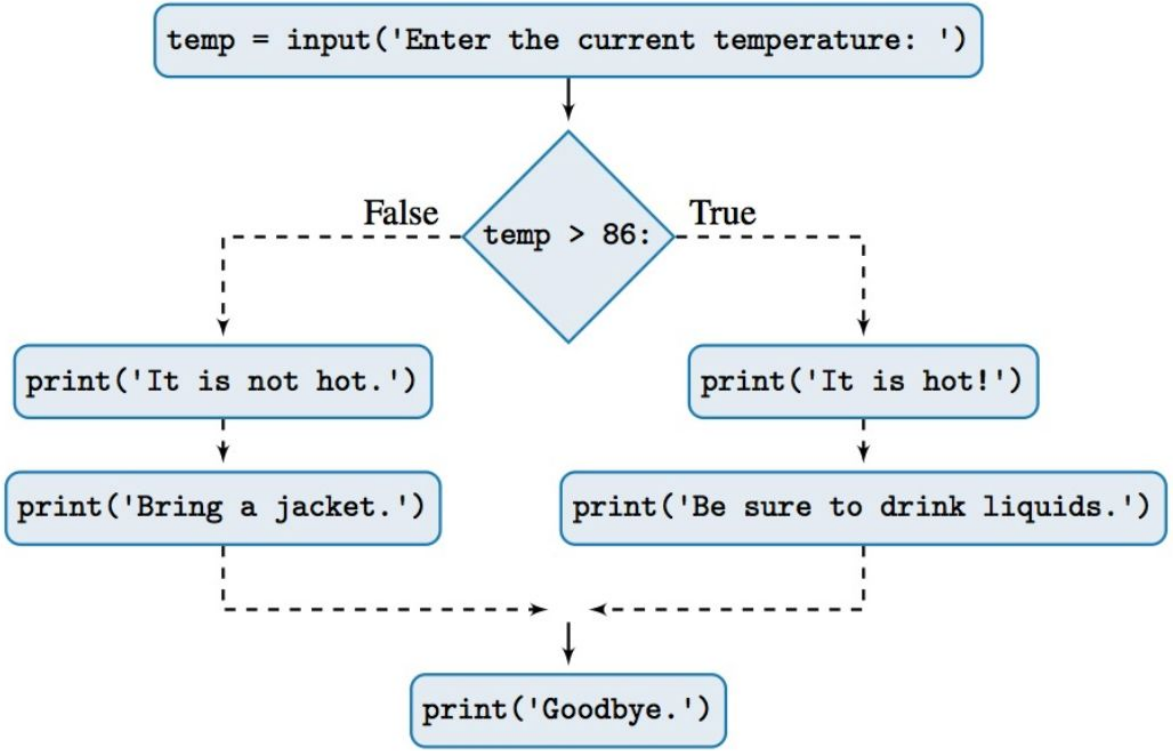
One-Way Decisions

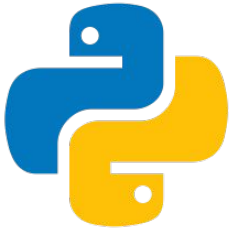
1. Evaluate boolean expression (**if** followed by boolean expression and `:`)
2. If True, execute indented code block (indented 4 spaces)
3. If False, skip indented code block
4. Execute non-indented code block

```
if <condition>:
    <indented code block>
<non-indented statement>
```



Two-Way Decisions



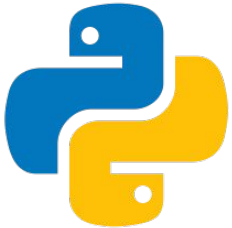


Two-Way Decisions

If statement (do if statement is true)

else clause (do if statement is not true -- false)

```
if <condition>:
    <indented code block 1>
else:
    <indented code block 2>
<non-indented statement>
```

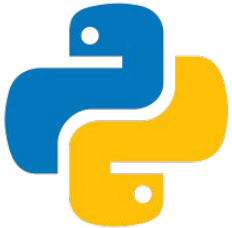


for loop - string

Suppose you want to spell the string entered by user

- Execute a print() statement for every character of the string name
- **for loop** can do this

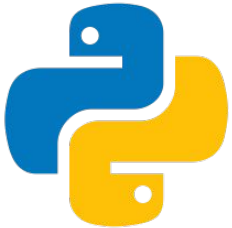
```
Enter a word: Lena
The word spelled out:
L
e
n
a
```



for loop - string

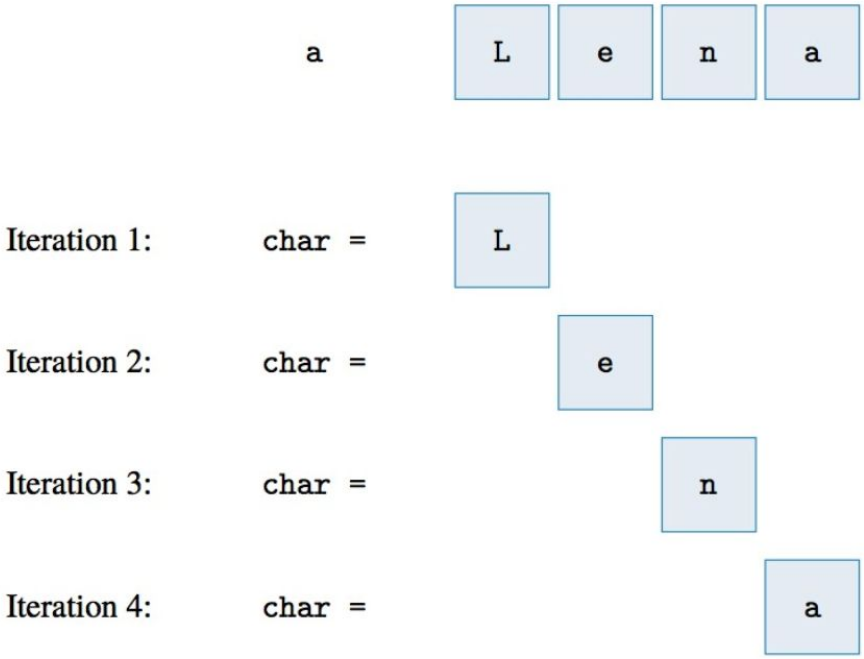
- char is a variable name
- for loop statement repeatedly assigns characters of string name to variable char
- For each value of char, indented print statement is executed

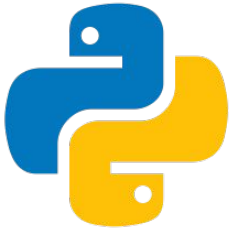
```
name = input('Enter a word: ')
print('The word spelled out: ')

for char in name:
    print(char)
```



for loop - string

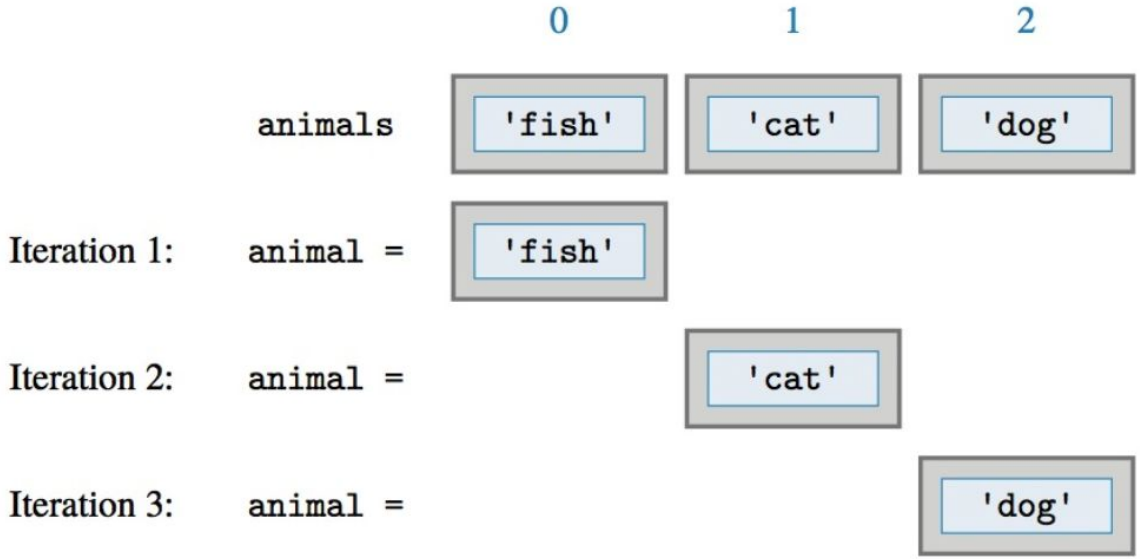
-





for loop - list

```
>>> animals = ['fish', 'cat', 'dog']
>>> for animal in animals:
        print(animal)

fish
cat
dog
```



for loops

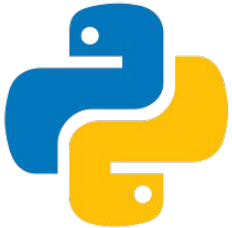
for loop will assign objects from <sequence> (string, list) to <variable> in the order they appear from left to right.

Indented code block = **body** executed once for every value of <variable>

for loop ***iterates*** through objects in sequence

After indented code block executed for last time, execute non-indented code block

```
for <variable> in <sequence>:
    <indented code block >
<non-indented code block>
```

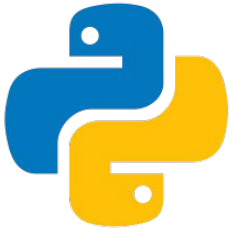


Nested Control Flow Structures

Write an application that starts by asking the user to enter a phrase. After the user has done so, the program will print all the vowels in the phrase, and no other letter.

Combine **for loop** and **if**

```
>>>
Enter a phrase: test case
e
a
e
```



range

Built-in function **range(stop)** - starts at 0

Used for getting a sequence of numbers in a given range

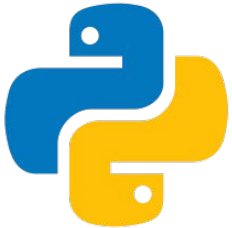
Typically iterates over the integer sequence 0, 1, 2, ... n -1

Can be used with **for loops**

range(start, end)

range(start, end, step)

https://docs.python.org/3/library/functions.html#func-range



Functions so far

len()

max()

sum()

print()

input()

eval()

min()

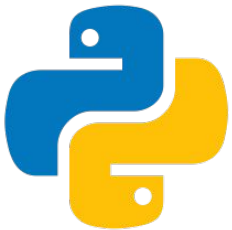
Function takes 0 or more input **arguments** and returns a result

Functions called in one line

Developer does not need to know what those statements are just what the input is and what the result is

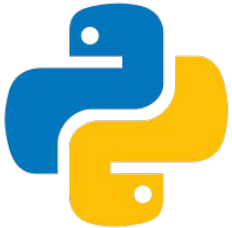
Functions simplify development programs

Cleaner. Make reading code easier



User-Defined Functions

Make a Python function named **f** that takes a number *x* as input and computes and returns the value **x**2 + 1**



User-Defined Functions - single argument

Starts with **def** keyword

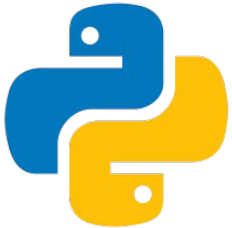
Followed by name of function

In parentheses are the names of the input arguments, if any

Colon at the end

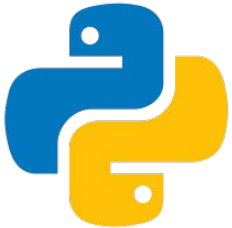
Indented code = body. Body is a set of statements that implement the function. Return statement here if function returns a value.

```
def <function name> (<0 or more variables>):
    <indented function body>
```



User-Defined Functions - multiple arguments

Suppose we want to define a function called square_sum() that takes 2 numbers x and y as input and returns the sum of their squares **x**2 + y**2**

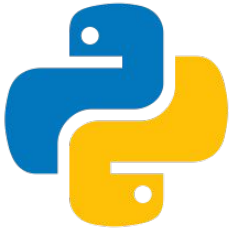


return vs print()

Print prints a value

Return returns a value

Only to call print() inside functions just be sure you print when you want to print and return when you want to return



Functions cont.

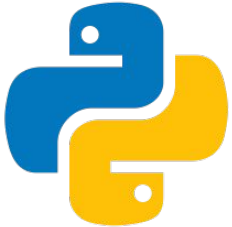
First define the function before you try to use it.

Function definitions are assignment statements

Document your code well!

- Developer understands the program
- User understands the program

Comments: used to identify main components of program and explain trickier parts

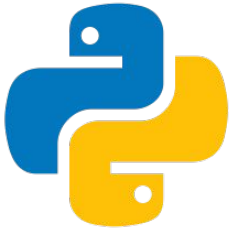


Functions cont.

Docstrings: functions should be documented for function users

help() - gives you the information in the docstring

String should describe what the function does and must be placed directly below the first line of the function definition



Parameter Passing

Calling program - function being called

Input arguments in function call are names of objects created in the calling program

Can be mutable or immutable