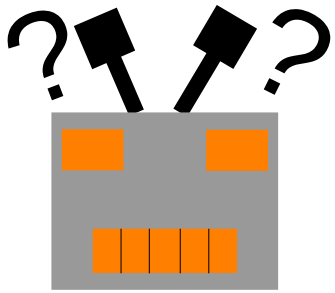
Code Club

Programming!

Computers

and Pre-programming

What is a computer?



- Examples?

What is a computer?

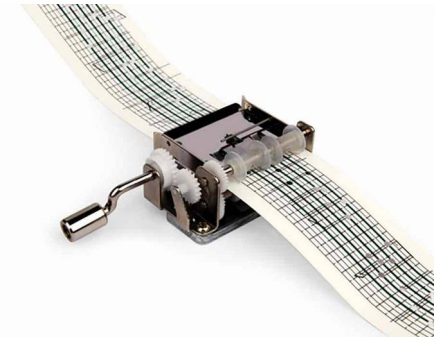
- Examples?
- PC, mobile, calculator

What is a computer?

- Examples?
- PC, mobile, calculator
- Raspberry Pi

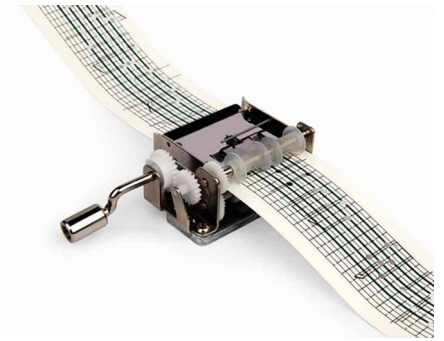
What is a computer?

- Examples?
- PC, mobile, calculator
- Raspberry Pi
- Mechanical music box



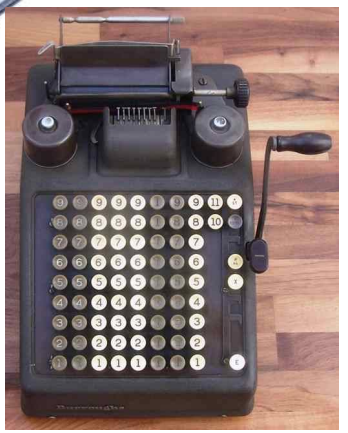
What is a computer?

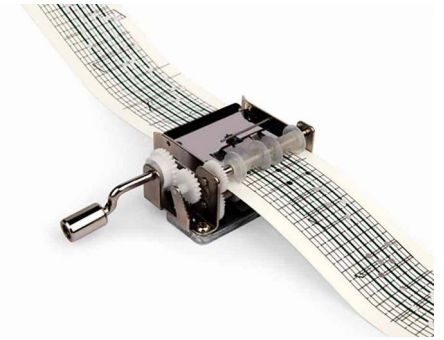
- Examples?
- PC, mobile, calculator
- Raspberry Pi
- Mechanical music box
- People!



What is a computer?

- Examples?
- PC, mobile, calculator
- Raspberry Pi
- Mechanical music box
- People!





- Inputs, memory, processor, outputs

INPUTS

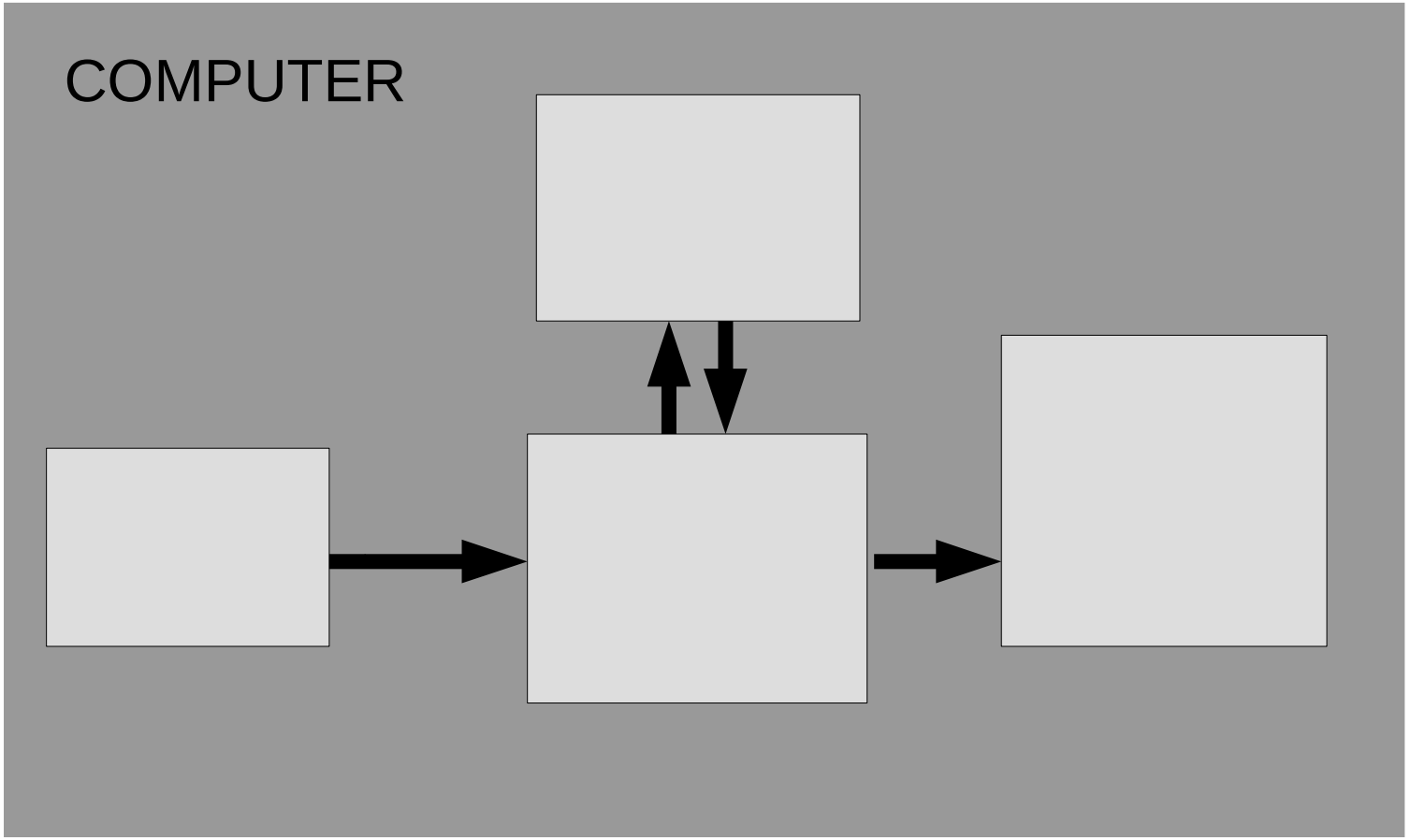
MEMORY

PROCESSOR

OUTPUTS

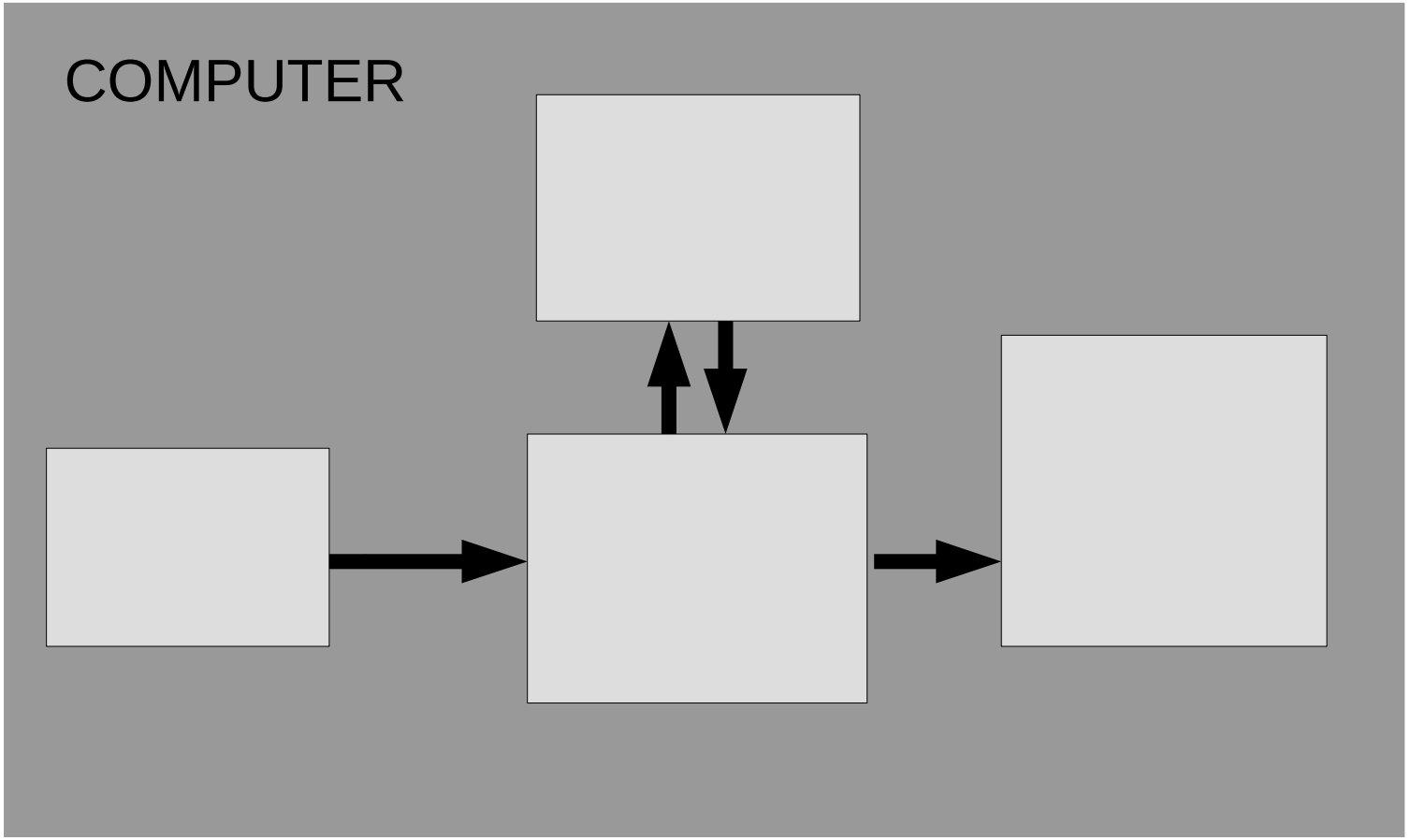
Black Box Model

- Each component is a black box



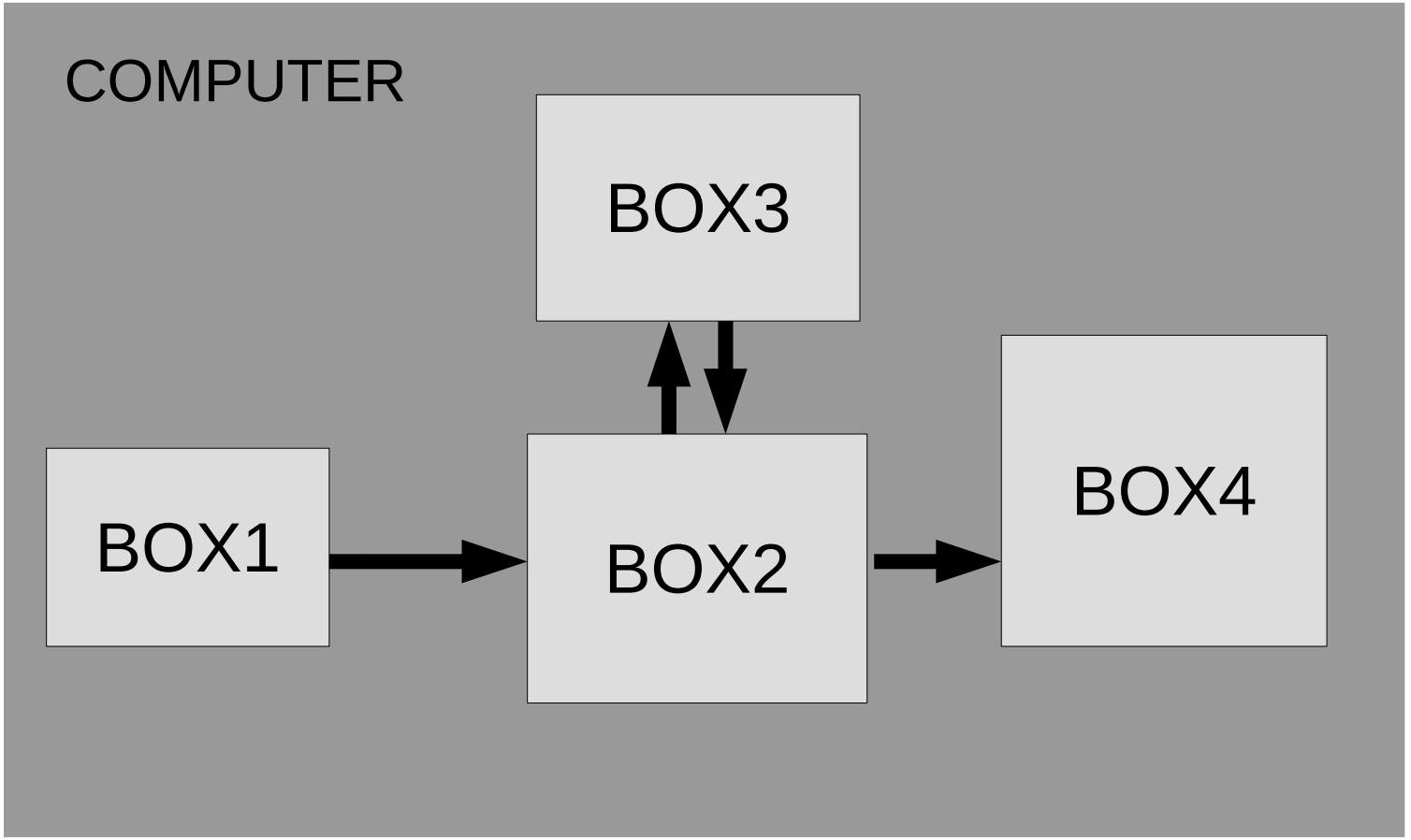
Black Box Model

- Each component is a black box
- “Black”: don’t care what “box” made of

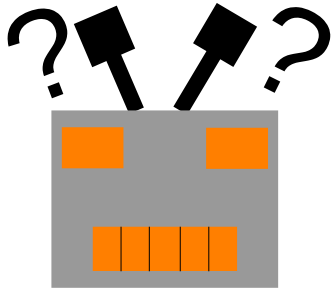


Black Box Model

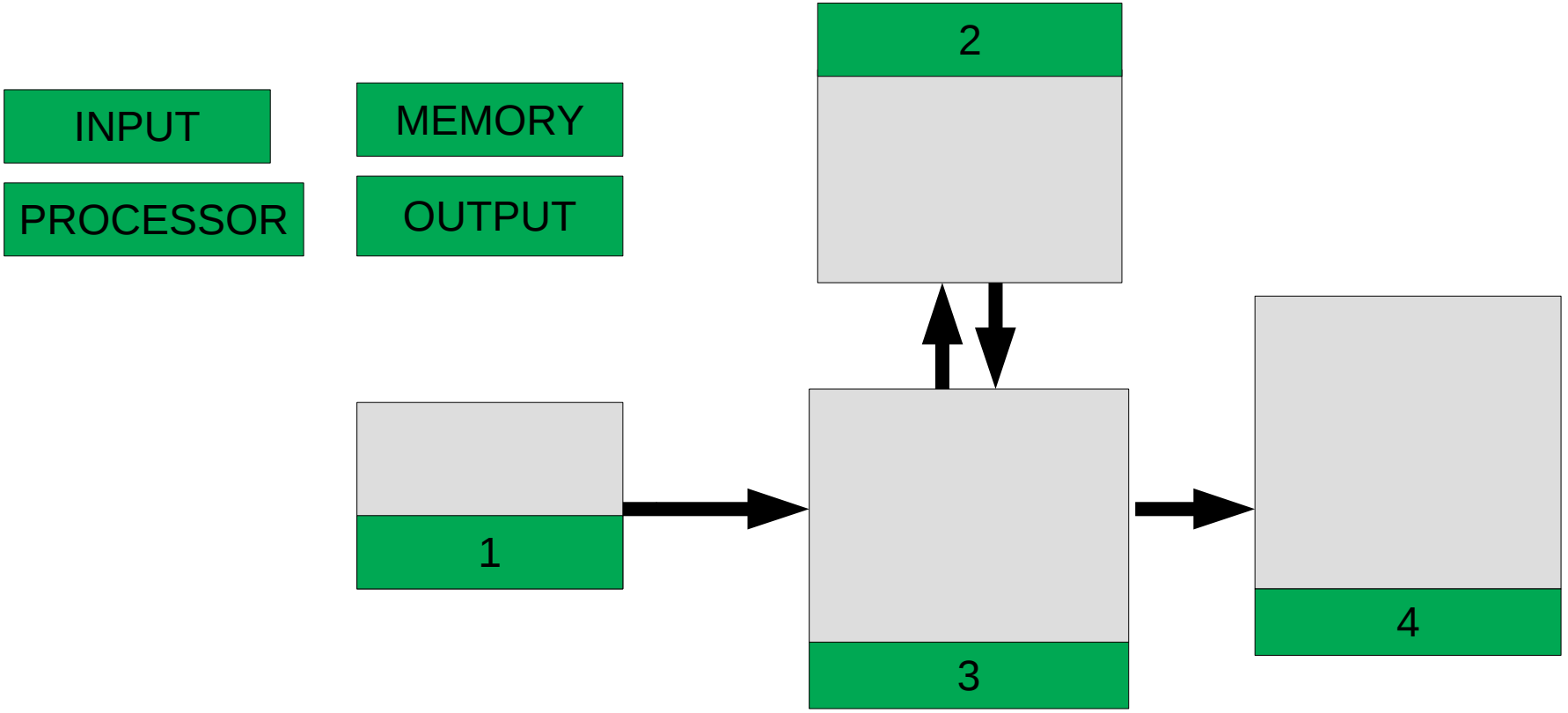
- Each component is a black box
- “Black”: don’t care what “box” made of



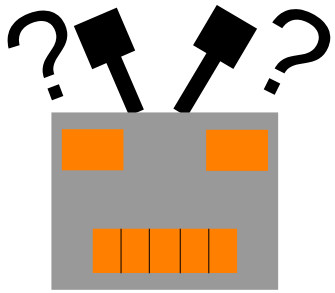
Black Box Model



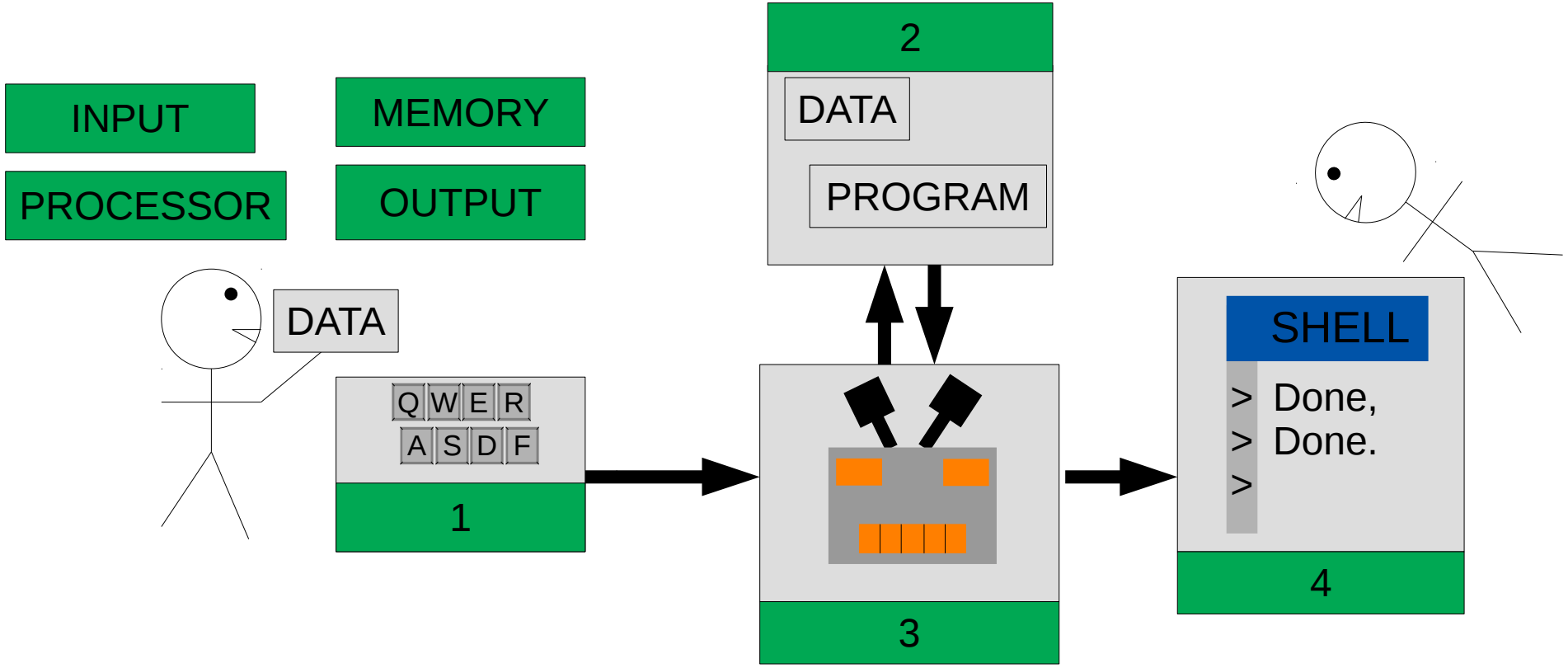
- Each component is a black box
- “Black”: don’t care what “box” made of



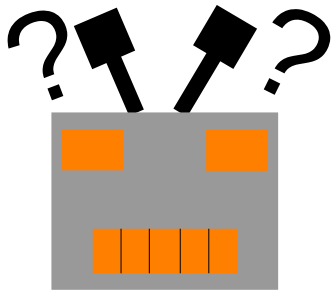
Black Box Model



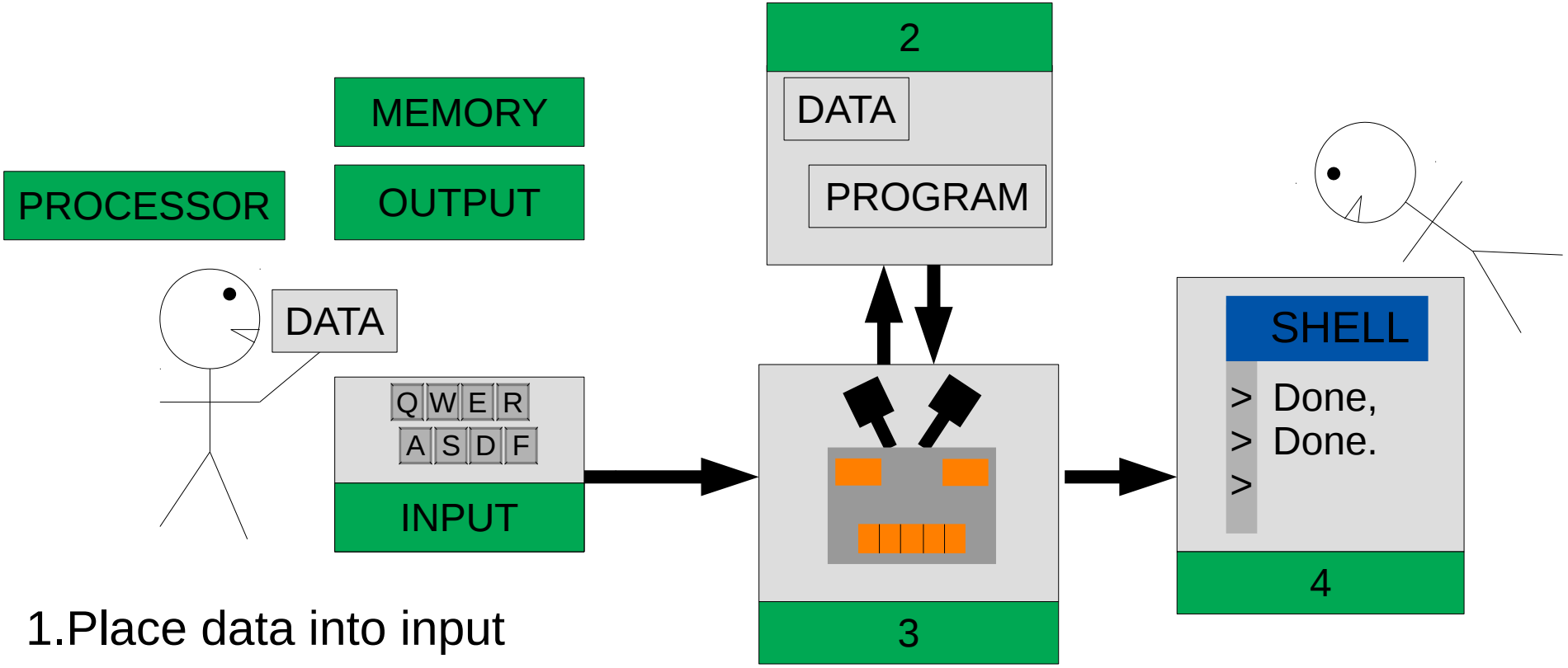
- Each component is a black box
- “Black”: don’t care what “box” made of



Black Box Model

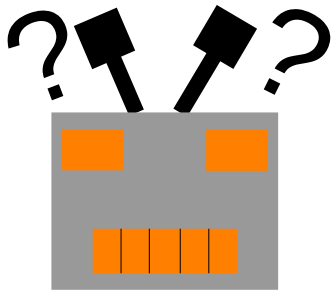


- Each component is a black box
- “Black”: don’t care what “box” made of

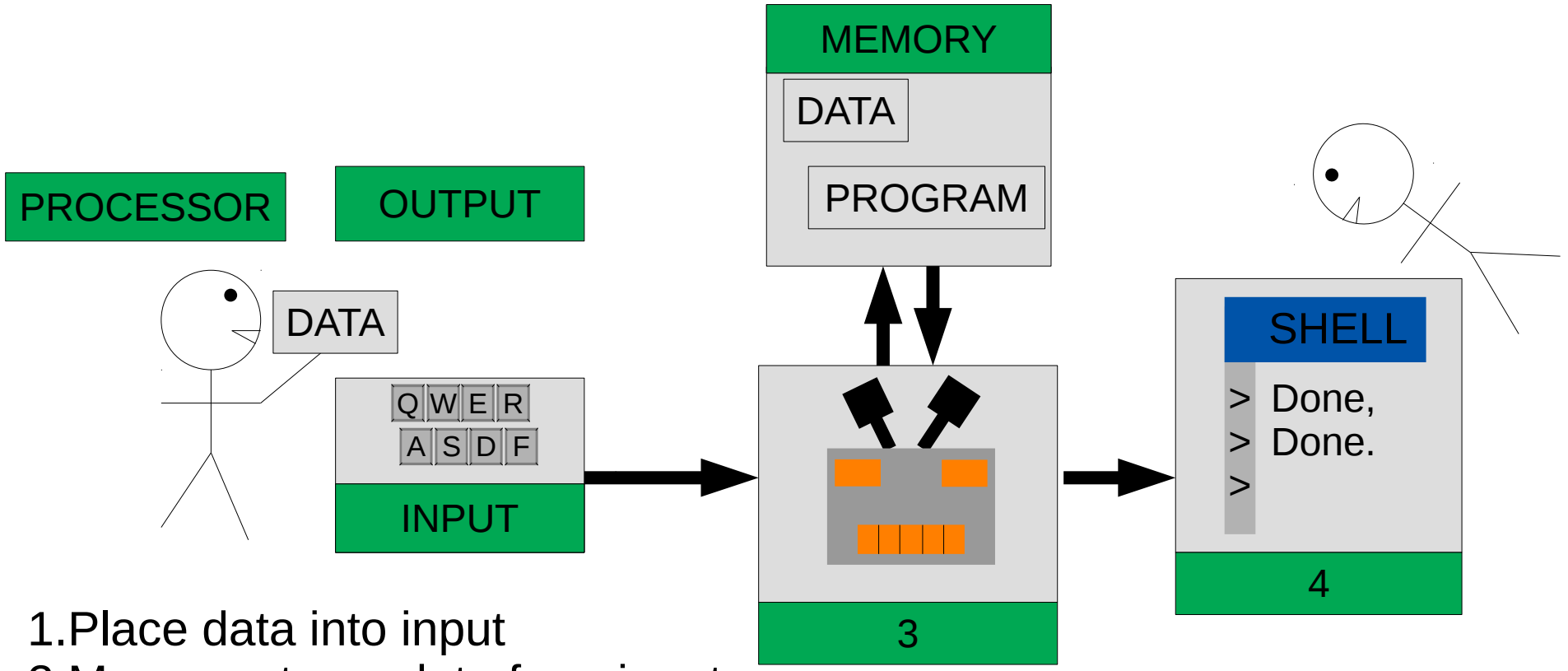


1.Place data into input

Black Box Model

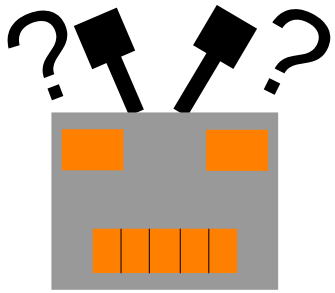


- Each component is a black box
- “Black”: don’t care what “box” made of

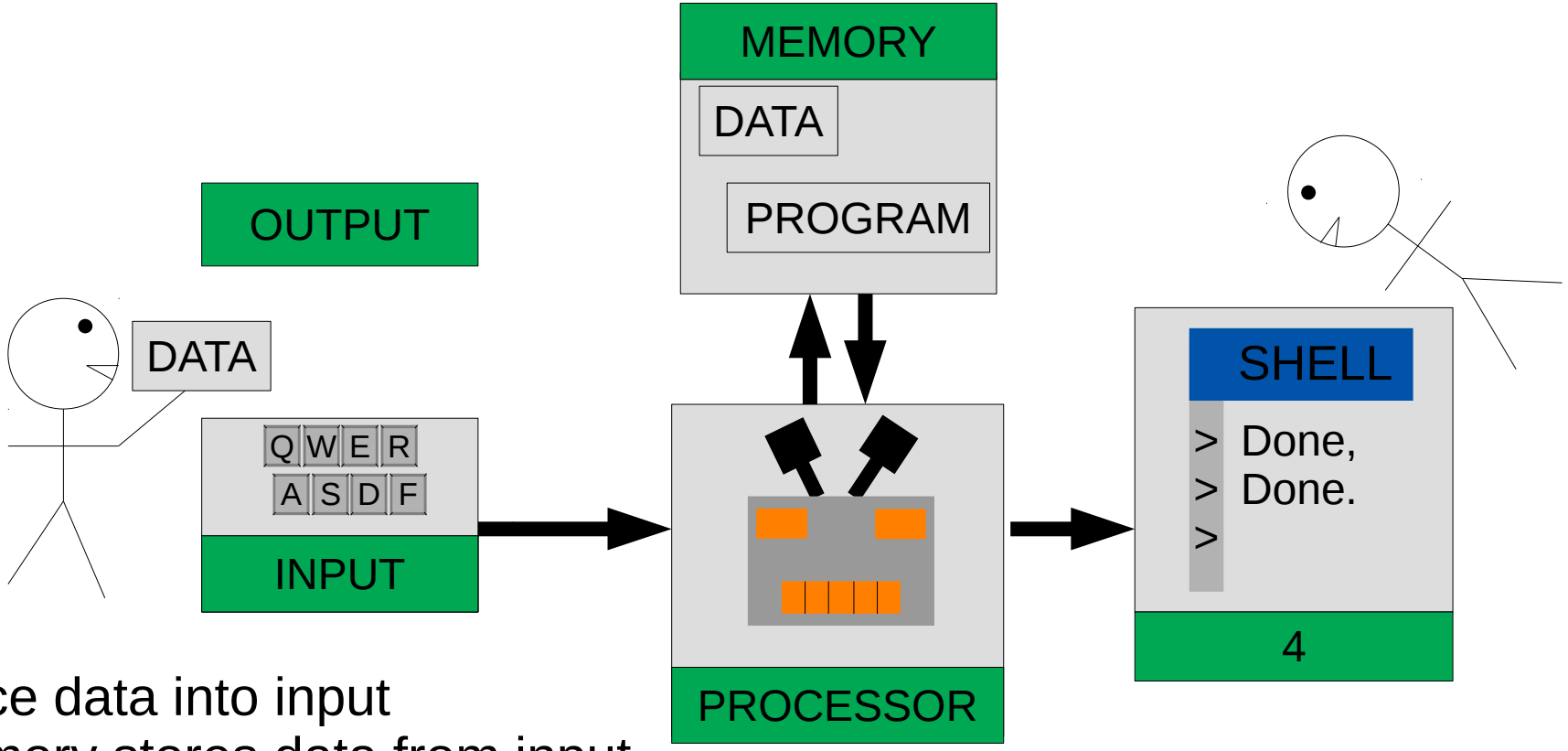


1.Place data into input
2.Memory stores data from input

Black Box Model

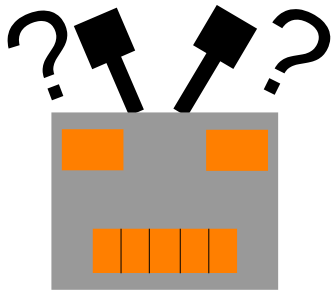


- Each component is a black box
- “Black”: don’t care what “box” made of

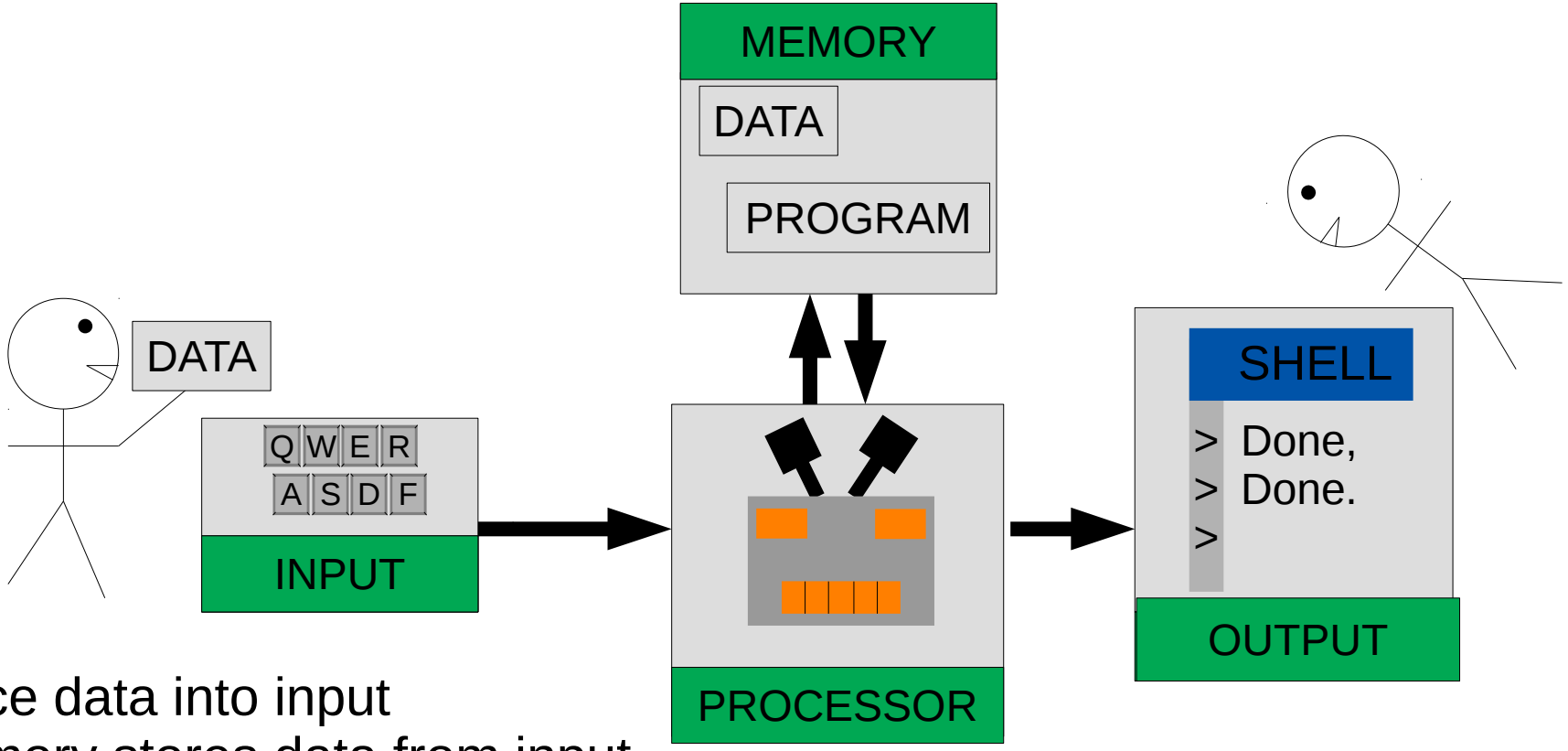


1.Place data into input
2.Memory stores data from input
3.Processor runs program on data in memory

Black Box Model

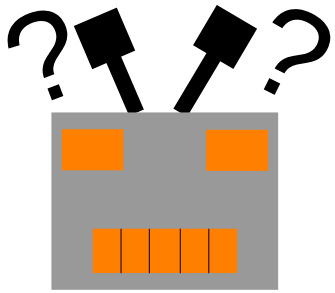


- Each component is a black box
- “Black”: don’t care what “box” made of

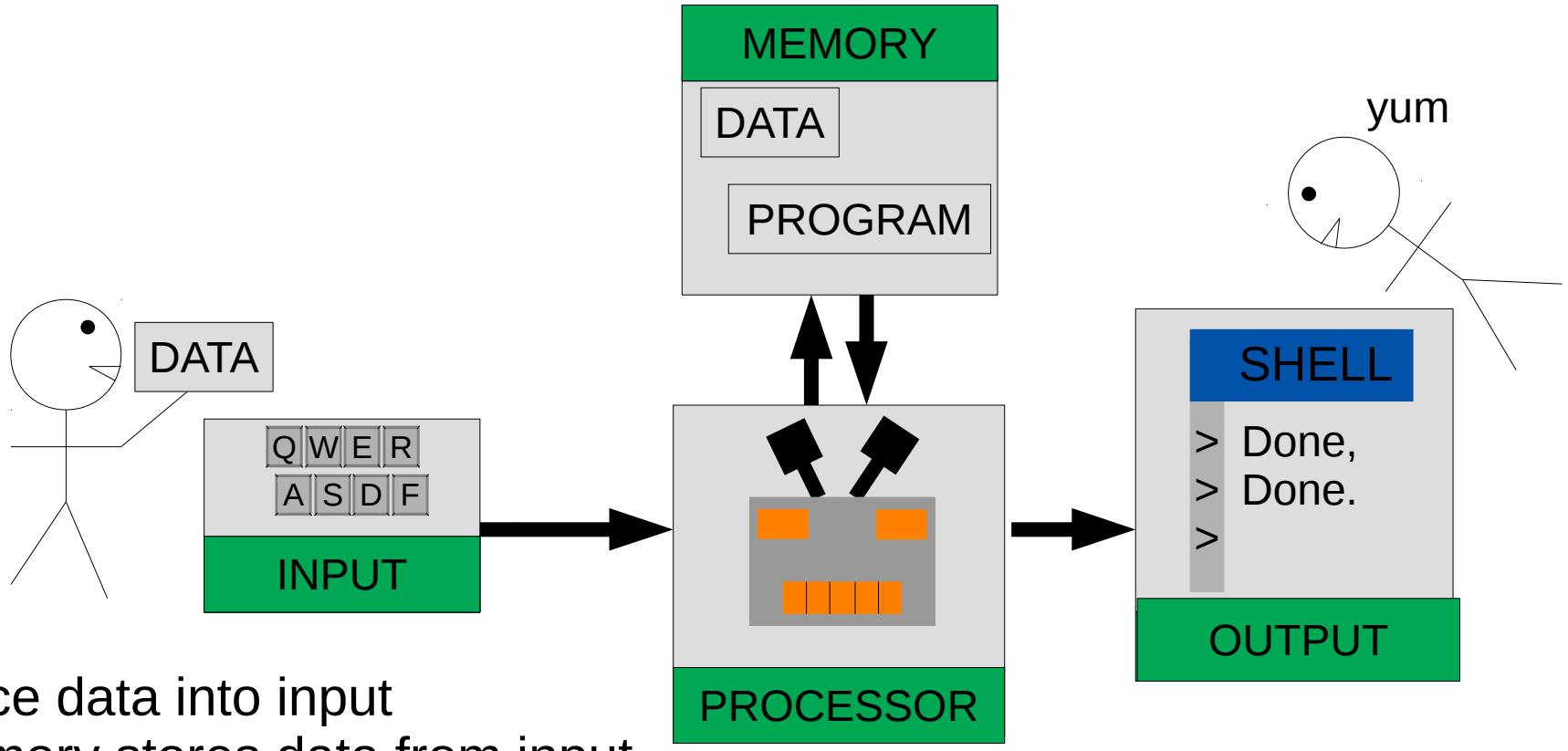


1. Place data into input
2. Memory stores data from input
3. Processor runs program on data in memory
4. Output puts out data

Black Box Model

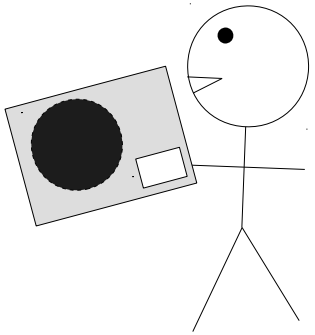


- Each component is a black box
- “Black”: don’t care what “box” made of

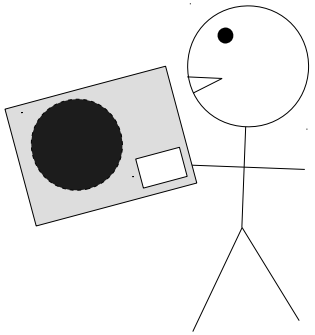


1. Place data into input
2. Memory stores data from input
3. Processor runs program on data in memory
4. Output puts out data

Pseudo code

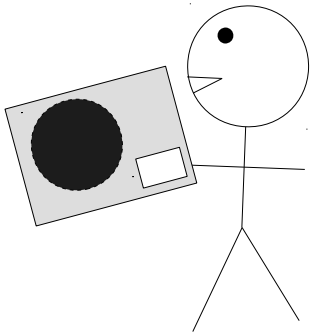


Pseudo code

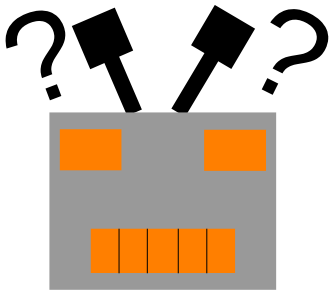


Pseudo code

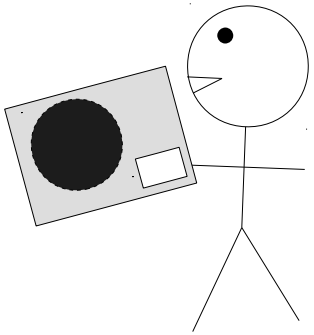
- Fix up holiday photos



Pseudo code



- Fix up holiday photos

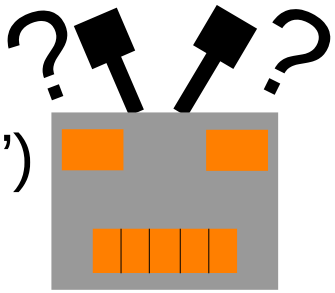


script.pseudo

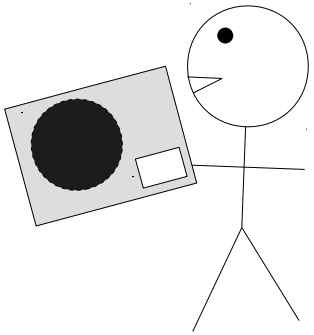
```
1
2
3
4
5
6
7
8
9
```

Pseudo code

Robot.read(“script.pseudo”)



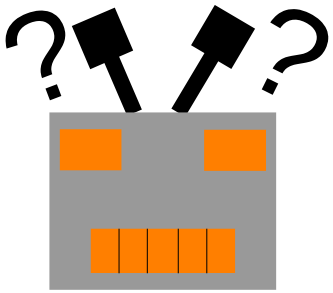
- Fix up holiday photos



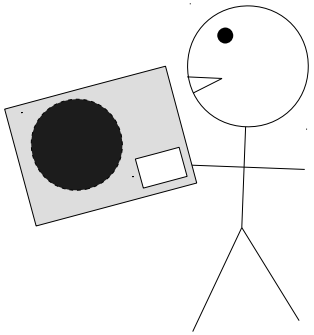
script.pseudo

```
1
2
3
4
5
6
7
8
9
```

Pseudo code



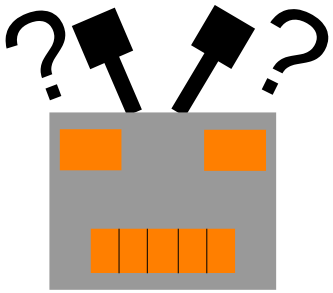
- Fix up holiday photos
- Break into steps...



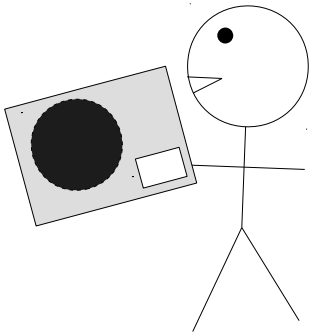
script.pseudo

```
1
2
3
4
5
6
7
8
9
```

Pseudo code



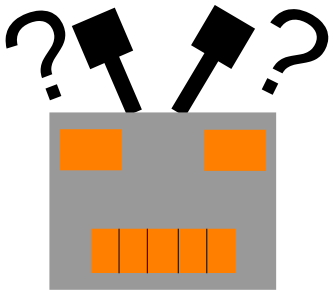
- Fix up holiday photos
- Break into steps...



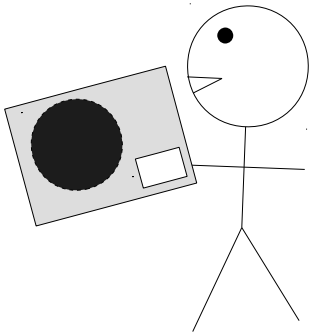
script.pseudo

```
1  for each photo,
2
3
4
5
6
7
8
9
```

Pseudo code



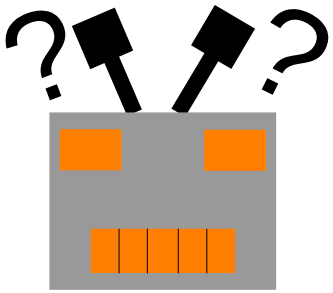
- Fix up holiday photos
- Break into steps...
- Flip upside down



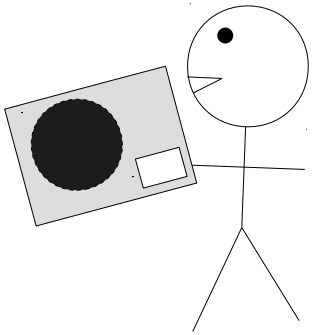
script.pseudo

```
1  for each photo,
2
3      flip upside_down
4
5
6
7
8
9
```

Pseudo code



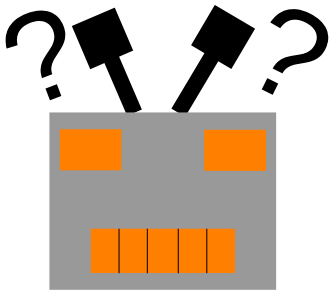
- Fix up holiday photos
- Break into steps...
- Flip upside down



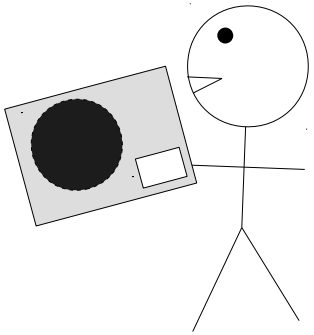
script.pseudo

```
1   for each photo,
2
3       flip upside_down
4
5       flip left_right
6
7
8
9
```

Pseudo code



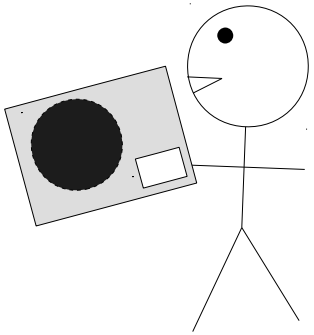
- Fix up holiday photos
- Break into steps...
- Flip upside down
- Flip left to right



script.pseudo

```
1    for each photo,
2
3        flip upside_down
4
5        flip left_right
6
7        leave photo
8
9
```

Pseudo code

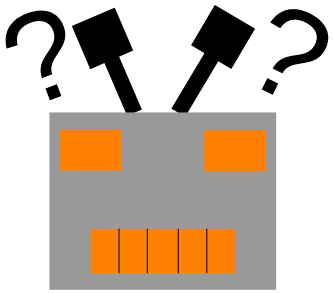


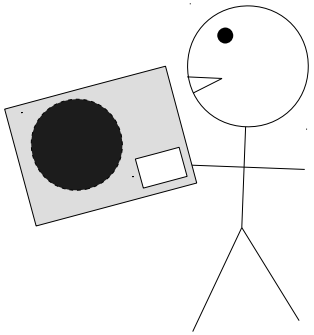
- Fix up holiday photos
- Break into steps...
- Flip upside down
- Flip left to right
- Decision making

script.pseudo

```
1   for each photo,
2
3       flip upside_down
4
5       flip left_right
6
7       leave photo
8
9
```

Pseudo code



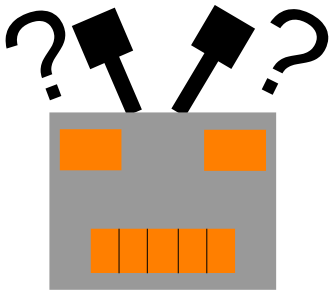


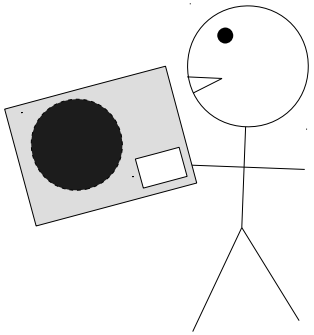
- Fix up holiday photos
- Break into steps…
- Flip upside down
- Flip left to right
- Decision making

script.pseudo

```
1   for each photo,
2     if photo upside_down,
3       flip upside_down
4
5       flip left_right
6
7       leave photo
8
9
```

Pseudo code



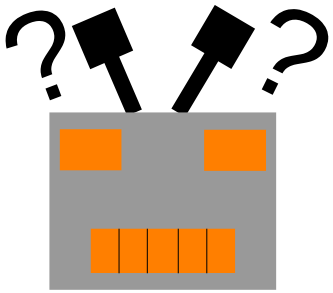


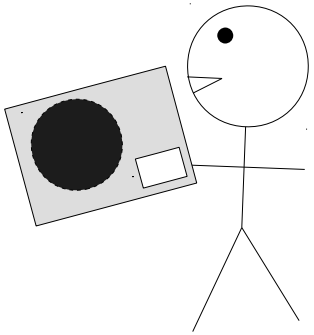
- Fix up holiday photos
- Break into steps...
- Flip upside down
- Flip left to right
- Decision making

script.pseudo

```
1    for each photo,
2       if photo upside_down,
3          flip upside_down
4       else if photo reversed,
5          flip left_right
6
7          leave photo
8
9
```

Pseudo code



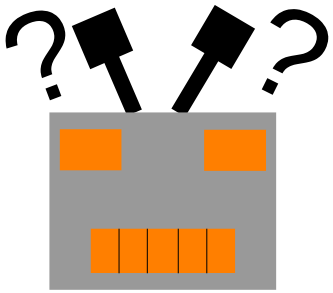


- Fix up holiday photos
- Break into steps...
- Flip upside down
- Flip left to right
- Decision making

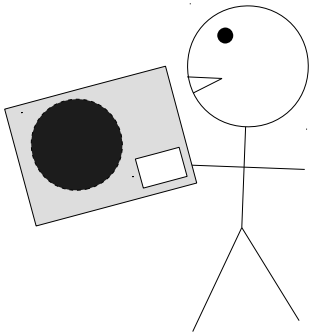
script.pseudo

```
1   for each photo,
2     if photo upside_down,
3       flip upside_down
4     else if photo reversed,
5       flip left_right
6     else,
7       leave photo
8
9
```

Pseudo code



- Fix up holiday photos
- Break into steps...
- Flip upside down
- Flip left to right
- Decision making



script.pseudo

```
1    for each photo,
2       if photo upside_down,
3          flip upside_down
4       else if photo reversed,
5          flip left_right
6       else,
7          leave photo
8       output “done”
9    output “finished”
```

Black Box Model

script.pseudo

```
1  for each photo,
2      if photo upside_down,
3          flip upside_down
4      else if photo reversed,
5          flip left_right
6      else,
7          leave photo
8      output “done”
9  output “finished”
```

MEMORY

DATA

PROGRAM

Q W E R

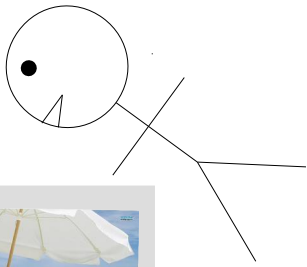
A S D F

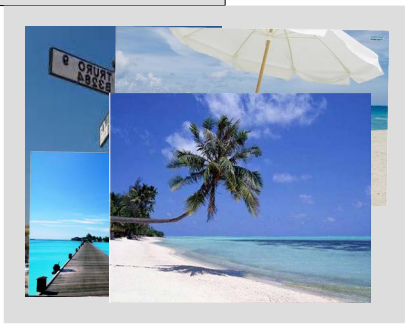
INPUT

PROCESSOR

SHELL

> Done,
> Done.
> Finished.





1. Place data into input
2. Memory stores data from input
3. Processor runs program on data in memory
4. Output puts out data

Computer Programming

with Python

Programming

- Programming == writing computer instructions

Programming

- Programming == writing computer instructions

script

```
1
2
3
4
5
6
7
8
9
10
11
```

Programming

- Programming == writing computer instructions

script

```
1  If this()
2  then
3      that()
4  done
5
6
7
8
9
10
11
```

Programming

- Programming == writing computer instructions

  Script == just a text file

script

```
1  If this()
2  then
3      that()
4  done
5
6
7
8
9
10
11
```

Programming

- Programming == writing computer instructions

  Script == just a text file

script.txt

```
1  If this()
2  then
3      that()
4  done
5
6
7
8
9
10
11
```

Programming

- Programming == writing computer instructions

  Script == just a text file

- Rougly one instruction per line

script.txt

```
1  If this()
2  then
3      that()
4  done
5
6
7
8
9
10
11
```

Programming

- Programming == writing computer instructions

  Script == just a text file

- Rougly one instruction per line

script.txt

```
1  If this()
2  then
3      that()
4  done
5
6
7  while True
8  do
9
10
11
```

Programming

- Programming == writing computer instructions

  Script == just a text file

- Rougly one instruction per line

script.txt

```
1  If this()
2  then
3      that()
4  done
5
6
7  while True
8  do
9     this()
10
11
```

Programming

- Programming == writing computer instructions

  Script == just a text file

- Rougly one instruction per line

script.txt

```
1   If this()
2   then
3       that()
4   done
5
6
7   while True
8   do
9       this()
10      that()
11
```

Programming

- Programming == writing computer instructions

  Script == just a text file

- Rougly one instruction per line

script.txt

```
1   If this()
2   then
3       that()
4   done
5
6
7   while True
8   do
9       this()
10      that()
11  done
```

Programming

- Programming == writing computer instructions

  Script == just a text file

- Rougly one instruction per line
- Python (C, Javascript, Perl…)

script.txt

```
1   If this()
2   then
3       that()
4   done
5
6
7   while True
8   do
9       this()
10      that()
11  done
```

●●●

Programming

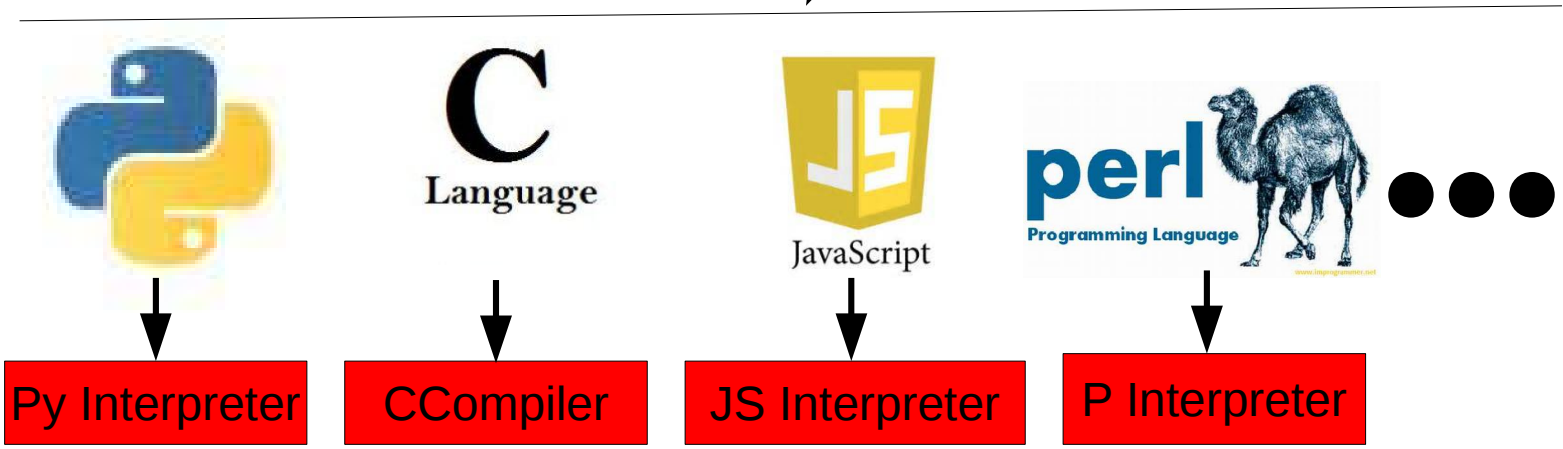
- Programming == writing computer instructions

  Script == just a text file

- Rougly one instruction per line
- Python (C, Javascript, Perl…)

script.txt

```
1   If this()
2   then
3       that()
4   done
5
6
7   while True
8   do
9       this()
10      that()
11  done
```



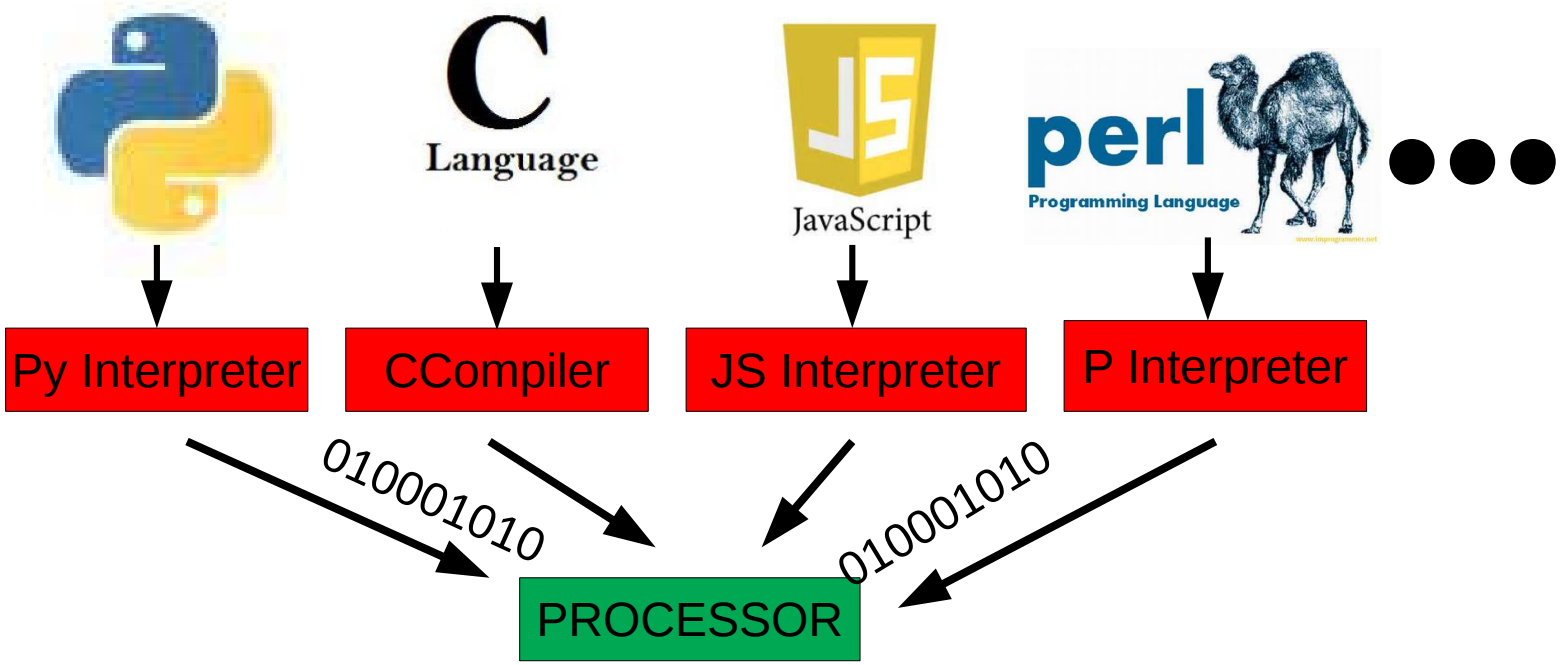
Programming

- Programming == writing computer instructions

  Script == just a text file
- Rougly one instruction per line
- Python (C, Javascript, Perl…)
- Convert to same processor language

script.txt

```
1   If this()
2   then
3       that()
4   done
5
6
7   while True
8   do
9       this()
10      that()
11  done
```



Programming

- Programming == writing computer instructions

  Script == just a text file

- Rougly one instruction per line
- Python (C, Javascript, Perl…)
- Convert to same processor language

script.txt

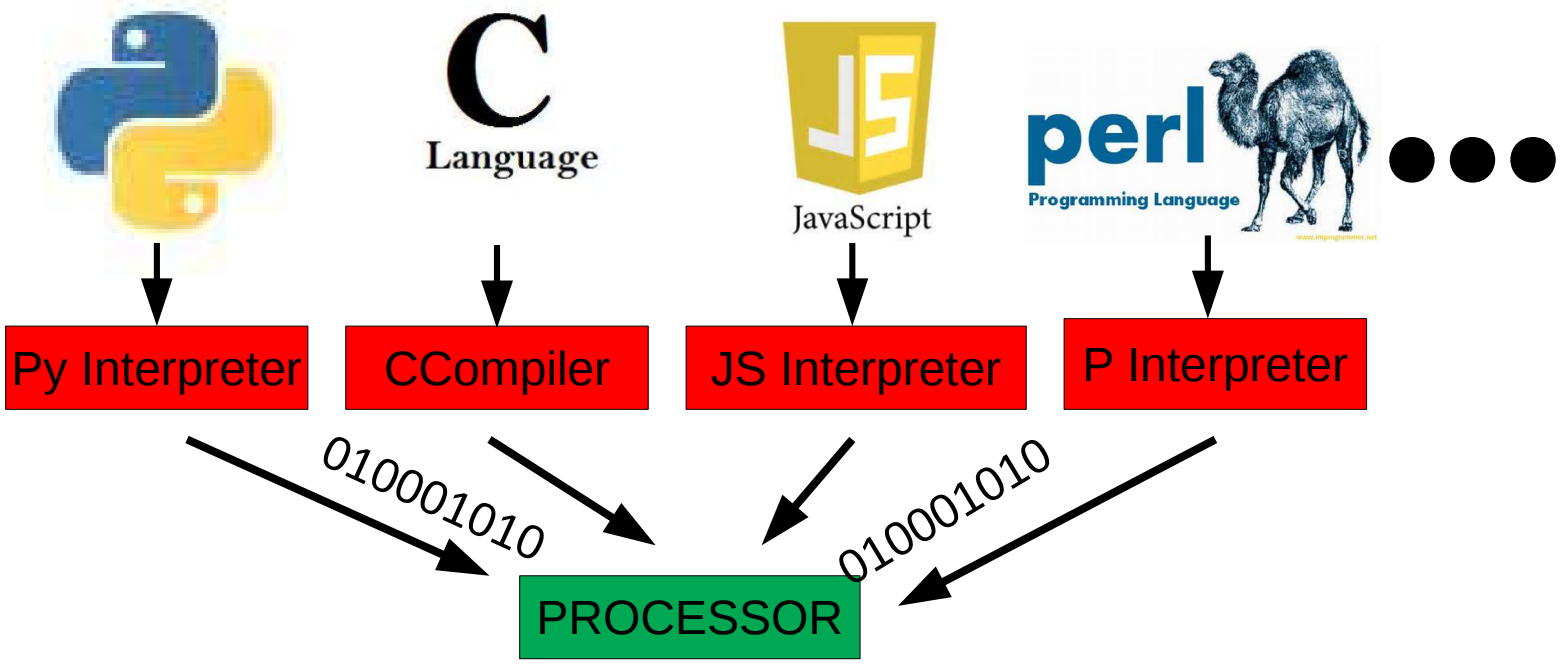
```
1   If this()
2   then
3       that()
4   done
5
6
7   while True
8   do
9       this()
10      that()
11  done
```

Editor

```
while True:
    print("banana")
```

Shell

```
> banana
> banana
> banana
```



Programming

- Programming == writing computer instructions

  Script == just a text file
- Rougly one instruction per line
- Python (C, Javascript, Perl…)
- Convert to same processor language
- Python IDLE

Editor

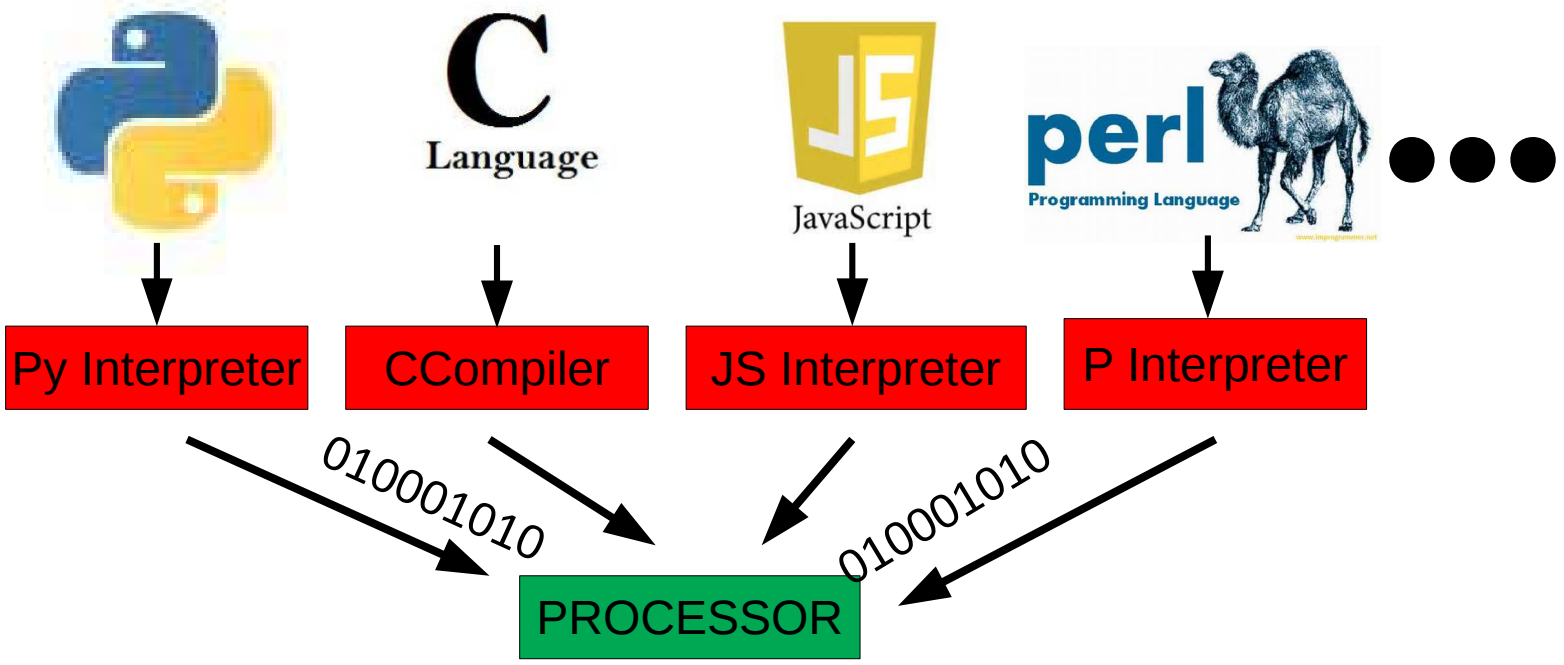
```
while True:
    print("banana")
```

Shell

```
> banana
> banana
> banana
```

script.txt

```
1   If this()
2   then
3       that()
4   done
5
6
7   while True
8   do
9       this()
10      that()
11  done
```



Programming in Python

- Complete specific task

Programming in Python

- Complete specific task

ask.py

```
1
2
3
4
5
6
7
8
9
10
```

Programming in Python

- Complete specific task

ask.py

```
1
2
3    ques = “What should I print? ”
4    print(ques)
5
6    resp = input()
7
8    output = “Ok, printing ” + resp
9    print(output)
10
```

Programming in Python

- Complete specific task

ask.py

```
1
2
3   ques = “What should I print? ”
4   print(ques)
5
6   resp = input()
7
8   output = “Ok, printing ” + resp
9   print(output)
10
```

ask.py

```

```

Programming in Python

- Complete specific task

ask.py

```
1
2
3   ques = “What should I print? ”
4   print(ques)
5
6   resp = input()
7
8   output = “Ok, printing ” + resp
9   print(output)
10
```

ask.py

```
What should I print?
```

Programming in Python

- Complete specific task

ask.py

```
1
2
3   ques = “What should I print? ”
4   print(ques)
5
6   resp = input()
7
8   output = “Ok, printing ” + resp
9   print(output)
10
```

ask.py

```
What should I print? Asdlfasdf
```

Programming in Python

- Complete specific task

ask.py

```
1
2
3   ques = “What should I print? ”
4   print(ques)
5
6   resp = input()
7
8   output = “Ok, printing ” + resp
9   print(output)
10
```

ask.py

```
What should I print? Asdlfasdf
Ok, printing Asdlfasdf
```

Programming in Python

- Complete specific task
- Function == quickly reusable task

ask.py

```
1   def question():
2
3       ques = “What should I print? ”
4       print(ques)
5
6       resp = input()
7
8       output = “Ok, printing ” + resp
9       print(output)
10
```

ask.py

```

```

Programming in Python

- Complete specific task
- Function == quickly reusable task

ask.py

```
1   def question():
2
3       ques = “What should I print? ”
4       print(ques)
5
6       resp = input()
7
8       output = “Ok, printing ” + resp
9       print(output)
10  question()
```

ask.py

```

```

Programming in Python

- Complete specific task
- Function == quickly reusable task

ask.py

```
1   def question():
2
3       ques = “What should I print? ”
4       print(ques)
5
6       resp = input()
7
8       output = “Ok, printing ” + resp
9       print(output)
10  question()
```

ask.py

```
What should I print?
```

Programming in Python

- Complete specific task
- Function == quickly reusable task

ask.py

```
1   def question():
2
3       ques = “What should I print? ”
4       print(ques)
5
6       resp = input()
7
8       output = “Ok, printing ” + resp
9       print(output)
10  question()
```

ask.py

```
What should I print? Asdlfasdf
```

Programming in Python

- Complete specific task
- Function == quickly reusable task

ask.py

```
1   def question():
2
3       ques = “What should I print? ”
4       print(ques)
5
6       resp = input()
7
8       output = “Ok, printing ” + resp
9       print(output)
10  question()
```

ask.py

```
What should I print? Asdlfasdf
Ok, printing Asdlfasdf
```

Programming in Python

- Complete specific task
- Function == quickly reusable task

ask.py

```
1   def question():
2
3       ques = “What should I print? ”
4       print(ques)
5
6       resp = input()
7
8       output = “Ok, printing ” + resp
9       print(output)
10  question()
```

ask.py

```
What should I print? Asdlfasdf
Ok, printing Asdlfasdf
```

script.py

```
1
2
3
4
5
6
```

Programming in Python

- Complete specific task
- Function == quickly reusable task

ask.py

```
1   def question():
2
3       ques = “What should I print? ”
4       print(ques)
5
6       resp = input()
7
8       output = “Ok, printing ” + resp
9       print(output)
10  question()
```

script.py

```
1
2
3   import ask
4
5
6
```

ask.py

```
What should I print? Asdlfasdf
Ok, printing Asdlfasdf
```

Programming in Python

- Complete specific task
- Function == quickly reusable task

ask.py

```
1   def question():
2
3       ques = “What should I print? ”
4       print(ques)
5
6       resp = input()
7
8       output = “Ok, printing ” + resp
9       print(output)
10  question()
```

script.py

```
1
2
3   import ask
4   ask.question()
5   ask.question()
6
```

ask.py

```
What should I print? Asdlfasdf
Ok, printing Asdlfasdf
```

Programming in Python

- Complete specific task
- Function == quickly reusable task

ask.py

```
1   def question():
2
3       ques = “What should I print? ”
4       print(ques)
5
6       resp = input()
7
8       output = “Ok, printing ” + resp
9       print(output)
10  question()
```

ask.py

```
What should I print? Asdlfasdf
Ok, printing Asdlfasdf
```

script.py

```
1
2
3   import ask
4   ask.question()
5   ask.question()
6
```

script.py

```
>
>
>
>
```

Programming in Python

- Complete specific task
- Function == quickly reusable task

ask.py

```
1   def question():
2
3       ques = “What should I print? ”
4       print(ques)
5
6       resp = input()
7
8       output = “Ok, printing ” + resp
9       print(output)
10  question()
```

ask.py

```
What should I print? Asdlfasdf
Ok, printing Asdlfasdf
```

script.py

```
1
2
3   import ask
4   ask.question()
5   ask.question()
6
```

script.py

```
>   What should I print?
>
>
>
```

Programming in Python

- Complete specific task
- Function == quickly reusable task

ask.py

```
1   def question():
2
3       ques = “What should I print? ”
4       print(ques)
5
6       resp = input()
7
8       output = “Ok, printing ” + resp
9       print(output)
10  question()
```

script.py

```
1
2
3   import ask
4   ask.question()
5   ask.question()
6
```

ask.py

```
What should I print? Asdlfasdf
Ok, printing Asdlfasdf
```

script.py

```
>   What should I print? ban
>
>
>
```

Programming in Python

- Complete specific task
- Function == quickly reusable task

ask.py

```
1   def question():
2
3       ques = “What should I print? ”
4       print(ques)
5
6       resp = input()
7
8       output = “Ok, printing ” + resp
9       print(output)
10  question()
```

ask.py

```
What should I print? Asdlfasdf
Ok, printing Asdlfasdf
```

script.py

```
1
2
3   import ask
4   ask.question()
5   ask.question()
6
```

script.py

```
>   What should I print? banana
>
>
>
```

Programming in Python

- Complete specific task
- Function == quickly reusable task

ask.py

```
1   def question():
2
3       ques = “What should I print? ”
4       print(ques)
5
6       resp = input()
7
8       output = “Ok, printing ” + resp
9       print(output)
10  question()
```

ask.py

```
What should I print? Asdlfasdf
Ok, printing Asdlfasdf
```

script.py

```
1
2
3   import ask
4   ask.question()
5   ask.question()
6
```

script.py

```
>   What should I print? banana
>   Ok, printing banana
>   What should I print?
>
```

Programming in Python

- Complete specific task
- Function == quickly reusable task

ask.py

```
1   def question():
2
3       ques = “What should I print? ”
4       print(ques)
5
6       resp = input()
7
8       output = “Ok, printing ” + resp
9       print(output)
10  question()
```

ask.py

```
What should I print? Asdlfasdf
Ok, printing Asdlfasdf
```

script.py

```
1
2
3   import ask
4   ask.question()
5   ask.question()
6
```

script.py

```
>   What should I print? banana
>   Ok, printing banana
>   What should I print? banana please
>
```

Programming in Python

- Complete specific task
- Function == quickly reusable task

ask.py

```
1   def question():
2
3       ques = “What should I print? ”
4       print(ques)
5
6       resp = input()
7
8       output = “Ok, printing ” + resp
9       print(output)
10  question()
```

ask.py

```
What should I print? Asdlfasdf
Ok, printing Asdlfasdf
```

script.py

```
1
2
3   import ask
4   ask.question()
5   ask.question()
6
```

script.py

```
>   What should I print? banana
>   Ok, printing banana
>   What should I print? banana please
>   Ok, printing banana please
```

Programming in Python

script.py

```
1  from gpiozero import Button
2  myButton1 = Button(21)
3  myVar1 = “Button has been ”
4
5  while True:
6      myButton1.wait_for_press()
7      print(myVar1 + "pressed")
8      myButton1.wait_for_release()
9      print(myVar1 + "released")
```

Programming in Python

script.py

```
1  from gpiozero import Button
2  myButton1 = Button(21)
3  myVar1 = “Button has been ”
4
5  while True:
6      myButton1.wait_for_press()
7      print(myVar1 + "pressed")
8      myButton1.wait_for_release()
9      print(myVar1 + "released")
```

Programming in Python

script.py

```
1  from gpiozero import Button
2  myButton1 = Button(21)
3  myVar1 = “Button has been ”
4
5  while True:
6      myButton1.wait_for_press()
7      print(myVar1 + "pressed")
8      myButton1.wait_for_release()
9      print(myVar1 + "released")
```

gpiozero.py

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
```

Programming in Python

script.py

```
1  from gpiozero import Button
2  myButton1 = Button(21)
3  myVar1 = “Button has been ”
4
5  while True:
6      myButton1.wait_for_press()
7      print(myVar1 + "pressed")
8      myButton1.wait_for_release()
9      print(myVar1 + "released")
```

gpiozero.py

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
```

Programming in Python

script.py

```
1  from gpiozero import Button
2  myButton1 = Button(21)
3  myVar1 = “Button has been ”
4
5  while True:
6      myButton1.wait_for_press()
7      print(myVar1 + "pressed")
8      myButton1.wait_for_release()
9      print(myVar1 + "released")
```

gpiozero.py

```
1
2
3
4  class Button(pin_number):
5
6      def wait_for_press():
7
8          pause until signal on pin {pin_number}
9
10         return
11
12
13     def wait_for_release():
14         ...
15         return
16
17
18
19
```

Programming in Python

script.py

```
1  from gpiozero import Button
2  myButton1 = Button(21)
3  myVar1 = “Button has been ”
4
5  while True:
6      myButton1.wait_for_press()
7      print(myVar1 + "pressed")
8      myButton1.wait_for_release()
9      print(myVar1 + "released")
```

gpiozero.py

```
1
2
3
4  class Button(pin_number):
5
6      def wait_for_press():
7
8          pause until signal on pin {pin_number}
9
10         return
11
12
13     def wait_for_release():
14         ...
15         return
16
17
18
19
```

Programming in Python

script.py

```
1  from gpiozero import Button
2  myButton1 = Button(21)
3  myVar1 = “Button has been ”
4
5  while True:
6      myButton1.wait_for_press()
7      print(myVar1 + "pressed")
8      myButton1.wait_for_release()
9      print(myVar1 + "released")
```

gpiozero.py

```
1
2
3
4  class Button(pin_number):
5
6      def wait_for_press():
7
8          pause until signal on pin {pin_number}
9
10         return
11
12
13     def wait_for_release():
14         ...
15         return
16
17
18
19
```

Programming in Python

script.py

```
1  from gpiozero import Button
2  myButton1 = Button(21)
3  myVar1 = “Button has been ”
4
5  while True:
6      myButton1.wait_for_press()
7      print(myVar1 + "pressed")
8      myButton1.wait_for_release()
9      print(myVar1 + "released")
```

gpiozero.py

```
1
2
3
4  class Button(pin_number):
5
6      def wait_for_press():
7
8          pause until signal on pin {pin_number}
9
10         return
11
12
13     def wait_for_release():
14         ...
15         return
16
17
18
19
```

Programming in Python

script.py

```
1  from gpiozero import Button
2  myButton1 = Button(21)
3  myVar1 = “Button has been ”
4
5  while True:
6      myButton1.wait_for_press()
7      print(myVar1 + "pressed")
8      myButton1.wait_for_release()
9      print(myVar1 + "released")
```

gpiozero.py

```
1   #other classes are defined in this script
2   ...
3
4   class Button(pin_number):
5
6       def wait_for_press():
7
8           pause until signal on pin {pin_number}
9
10          return
11
12
13      def wait_for_release():
14          ...
15          return
16
17  ...
18  class Power():
19          ...
```

Programming in Python

script.py

```
1  from gpiozero import Button
2  myButton1 = Button(21)
3  myVar1 = “Button has been ”
4
5  while True:
6      myButton1.wait_for_press()
7      print(myVar1 + "pressed")
8      myButton1.wait_for_release()
9      print(myVar1 + "released")
```

gpiozero.py

```
1   #other classes are defined in this script
2   ...
3
4   class Button(pin_number):
5
6       def wait_for_press():
7
8           pause until signal on pin {pin_number}
9
10          return
11
12
13      def wait_for_release():
14          ...
15          return
16
17  ...
18  class Power():
19          ...
```

Programming in Python

script.py

```
1  from gpiozero import Button
2  myButton1 = Button(21)
3  myVar1 = “Button has been ”
4
5  while True:
6      myButton1.wait_for_press()
7      print(myVar1 + "pressed")
8      myButton1.wait_for_release()
9      print(myVar1 + "released")
```

gpiozero.py

```
1   #other classes are defined in this script
2   ...
3
4   class Button(pin_number):
5
6       def wait_for_press():
7
8           pause until signal on pin {pin_number}
9
10          return
11
12
13      def wait_for_release():
14          ...
15          return
16
17  ...
18  class Power():
19          ...
```

Programming in Python

script.py

```
1  from gpiozero import Button
2  myButton1 = Button(21)
3  myVar1 = “Button has been ”
4
5  while True:
6      myButton1.wait_for_press()
7      print(myVar1 + "pressed")
8      myButton1.wait_for_release()
9      print(myVar1 + "released")
```

gpiozero.py

```
1   #other classes are defined in this script
2   ...
3
4   class Button(pin_number):
5
6       def wait_for_press():
7
8           pause until signal on pin {pin_number}
9
10          return
11
12
13      def wait_for_release():
14          ...
15          return
16
17  ...
18  class Power():
19          ...
```

Programming in Python

script.py

```
1  from gpiozero import Button
2  myButton1 = Button(21)
3  myVar1 = “Button has been ”
4
5  while True:
6      myButton1.wait_for_press()
7      print(myVar1 + "pressed")
8      myButton1.wait_for_release()
9      print(myVar1 + "released")
```

gpiozero.py

```
1   #other classes are defined in this script
2   ...
3
4   class Button(pin_number):
5
6       def wait_for_press():
7
8           pause until signal on pin {pin_number}
9
10          return
11
12
13      def wait_for_release():
14          ...
15          return
16
17  ...
18  class Power():
19          ...
```

Programming in Python

script.py

```
1  from gpiozero import Button
2  myButton1 = Button(21)
3  myVar1 = “Button has been ”
4
5  while True:
6      myButton1.wait_for_press()
7      print(myVar1 + "pressed")
8      myButton1.wait_for_release()
9      print(myVar1 + "released")
```

gpiozero.py

```
1   #other classes are defined in this script
2   ...
3
4   class Button(pin_number):
5
6       def wait_for_press():
7
8           pause until signal on pin {pin_number}
9
10          return
11
12
13      def wait_for_release():
14          ...
15          return
16
17  ...
18  class Power():
19          ...
```

Programming in Python

script.py

```
1  from gpiozero import Button
2  myButton1 = Button(21)
3  myVar1 = “Button has been ”
4
5  while True:
6      myButton1.wait_for_press()
7      print(myVar1 + "pressed")
8      myButton1.wait_for_release()
9      print(myVar1 + "released")
```

script.py

```
> > > >
```

gpiozero.py

```
1   #other classes are defined in this script
2   ...
3
4   class Button(pin_number):
5
6       def wait_for_press():
7
8           pause until signal on pin {pin_number}
9
10          return
11
12
13      def wait_for_release():
14          ...
15          return
16
17  ...
18  class Power():
19          ...
```

Programming in Python

script.py

```
1  from gpiozero import Button
2  myButton1 = Button(21)
3  myVar1 = “Button has been ”
4
5  while True:
6      myButton1.wait_for_press()
7      print(myVar1 + "pressed")
8      myButton1.wait_for_release()
9      print(myVar1 + "released")
```

script.py

```
>  Button has been pressed
>  Button has been released
>
>
```

gpiozero.py

```
1   #other classes are defined in this script
2   ...
3
4   class Button(pin_number):
5
6       def wait_for_press():
7
8           pause until signal on pin {pin_number}
9
10          return
11
12
13      def wait_for_release():
14          ...
15          return
16
17  ...
18  class Power():
19          ...
```

Programming in Python

script.py

```
1  from gpiozero import Button
2  myButton1 = Button(21)
3  myVar1 = “Button has been ”
4
5  while True:
6      myButton1.wait_for_press()
7      print(myVar1 + "pressed")
8      myButton1.wait_for_release()
9      print(myVar1 + "released")
```

script.py

```
>  Button has been pressed
>  Button has been released
>  Button has been pressed
>  Button has been released
```

gpiozero.py

```
1   #other classes are defined in this script
2   ...
3
4   class Button(pin_number):
5
6       def wait_for_press():
7
8           pause until signal on pin {pin_number}
9
10          return
11
12
13      def wait_for_release():
14          ...
15          return
16
17  ...
18  class Power():
19          ...
```

Get Coding!