Introduction to Python Programming

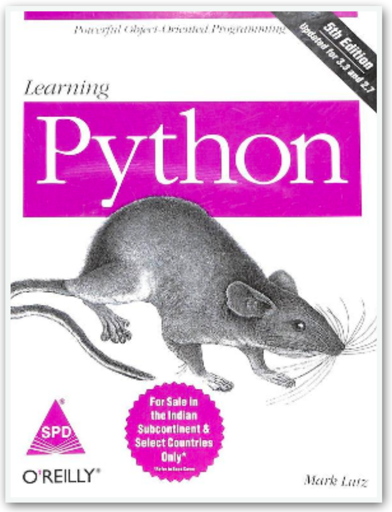
Contents

Basics of Python Programming

- "Do things with stuff"

In an informal sense, in Python, we do things with stuff. "Things" take the form of operations like addition and concatenation, and "stuff" refers to the objects on which we perform those operations.

— Learning Python

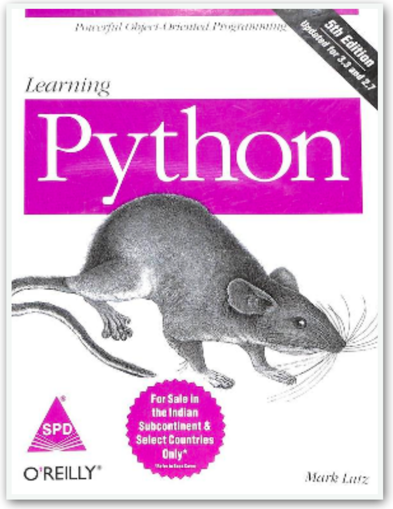


Basics of Python Programming

- "Do things with stuff"
  - What does a computer do
    - ✓ Perform calculations
    - ✓ Memorize results

In an informal sense, in Python, we do things with stuff. "Things" take the form of operations like addition and concatenation, and "stuff" refers to the objects on which we perform those operations.

— Learning Python

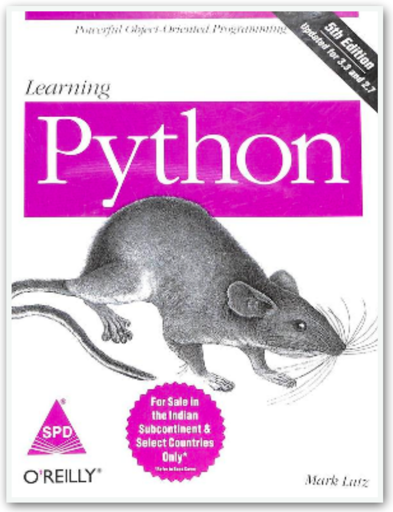


Basics of Python Programming

- "Do things with stuff"
  - How does a program work
    - ✓ A sequence of instructions stored inside computer
    - ✓ Interpreter executes each instruction in order

In an informal sense, in Python, we do things with stuff. "Things" take the form of operations like addition and concatenation, and "stuff" refers to the objects on which we perform those operations.

— Learning Python


Powerful Object-Oriented Programming
5th Edition
Learning
Python
SPD
O'REILLY
For Sale in the Indian Subcontinent & Select Countries Only
Mark Lutz

Basics of Python Programming

- “Do things with stuff”
  - How does a program work
    - ✓ A sequence of instructions stored inside computer
    - ✓ Interpreter executes each instruction in order

```python
print(1 + 2)          # Execute the 1st instruction
print('Hello!')       # Execute the 2nd instruction
print(3.5 + 2.6)      # Execute the 3rd instruction
```

```
3
Hello!
6.1
```

Basics of Python Programming

- "Do things with stuff"
  - How does a program work
    - ✓ A sequence of instructions stored inside computer
    - ✓ Interpreter executes each instruction in order

```python
print(1 + 2)          # Execute the 1st instruction
print('Hello!')       # Execute the 2nd instruction
print(3.5 + 2.6)      # Execute the 3rd instruction
```

```
3
Hello!
6.1
```

Basics of Python Programming

- "Do things with stuff"
  - How does a program work
    - ✓ A sequence of instructions stored inside computer
    - ✓ Interpreter executes each instruction in order

```
print(1 + 2)          # Execute the 1st instruction
print('Hello!')       # Execute the 2nd instruction
print(3.5 + 2.6)      # Execute the 3rd instruction
```

```
3
Hello!
6.1
```

Basics of Python Programming

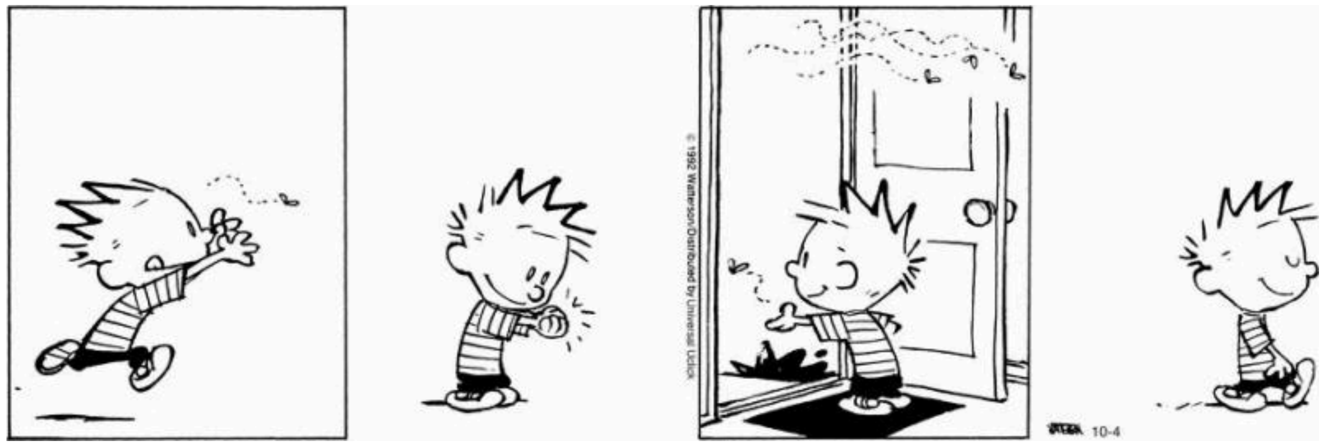
- "Do things with stuff"
  - How does a program work
    - ✓ A sequence of instructions stored inside computer
    - ✓ Interpreter executes each instruction in order

```python
print(1 + 2)           # Execute the 1st instruction
print('Hello!')        # Execute the 2nd instruction
print(3.5 + 2.6)       # Execute the 3rd instruction
```

```
3
Hello!
6.1
```

Basics of Python Programming

- "Do things with stuff"
  - Where a program goes wrong
    - ✓ Syntax errors
    - ✓ Runtime errors
    - ✓ Logic errors



"When you fix one bug, you introduce several newer bugs."

Basics of Python Programming

- Memorizing results as Python objects
  - Python objects: everything created in Python is implemented as an **object**.
  - Objects are pieces of memory, with values and associated operations.

Basics of Python Programming

- Memorizing results as Python objects
  - Types of objects
    - ✓ All Python objects are associated with a **type**
    - ✓ Function `type()` to retrieve data type

```
print(type(5))            # The type of the object 5 is int
print(type(2.3))          # The type of the object 2.3 is float
print(type('Hello!'))     # The type of the object "World" is str
```

```
<class 'int'>
<class 'float'>
<class 'str'>
```

Decimal points imply floating point numbers

It is almost equivalent to "type"

Basics of Python Programming

- Memorizing results as Python objects
  - Types of objects
    - ✓ All Python objects are associated with a **type**
    - ✓ Function `type()` to retrieve data type

```
print(type(5))              # The type of the object 5 is int
print(type(2.3))            # The type of the object 2.3 is float
print(type('Hello!'))       # The type of the object "World" is str
```

```
<class 'int'>
<class 'float'>
<class 'str'>
```

Create strings using quotation marks

Basics of Python Programming

- Memorizing results as Python objects
    - Types of objects

Notes:

The type of an object is very important because:

- It defines the possible values for objects;
- It determines the operations that the object supports;

```
print(2.3 + 5)                    # Numerical addition via "+"
print('Hello ' + 'World')    # Concatenating strings via "+"
```

```
7.3
Hello World
```

Basics of Python Programming

- Memorizing results as Python objects
  - Types of objects

Notes:

The type of an object is very important because:

- It defines the possible values for objects;
- It determines the operations that the object supports;

```
print(2.3 * 5)                  # Numerical multiplication via "*"
print('Hello ' * 5)             # Repeat the string via "*"
```

```
11.5
Hello Hello Hello Hello Hello
```

Basics of Python Programming

- Memorizing results as Python objects
  - Variables and assignment statements

Example 1: John had a saving account with a $1000 balance. He received a 1% interest and made a 1) $200 deposit and then 2) a $350 withdrawal. What is the current balance of the saving account?

```
print(1000*(1+0.01) + 200 - 350)
```

```
860.0
```

Basics of Python Programming

- Memorizing results as Python objects
  - Variables and assignment statements
    - ✓ Assignment operator `=`
    - ✓ Variable(s): name(s) on the left
    - ✓ Object(s): expression(s) on the right

```
interest = 0.01                    # Interest rate
deposit = 200                      # Deposit
withdrawal = 350                   # Withdrawal
balance = 1000                     # Balance of the account

print(balance*(1+interest) + deposit - withdrawal)
```

```
860.0
```

Basics of Python Programming

- Memorizing results as Python objects
  - Variables and assignment statements
    - ✓ Assignment operator `=`
    - ✓ Variable(s): name(s) on the left
    - ✓ Object(s): expression(s) on the right

```
interest = 0.01                         # Interest rate
deposit = 200                           # Deposit
withdrawal = 350
balance = 1000

print(balance*(1+interest)
```

```
860.0
```

Notes: syntax for variable names:

- Only one word
- Only consist of letters, numbers, and underscores
- Cannot begin with a number
- Avoid contradictions with Python keywords

Basics of Python Programming

- Memorizing results as Python objects
  - Variables and assignment statements

```
interest = 0.01
deposit = 200
withdrawal = 350

balance = 1000
balance = balance * (1+interest)
balance = balance + deposit
balance = balance - withdrawal

print(balance)
```

```
860.0
```

Basics of Python Programming

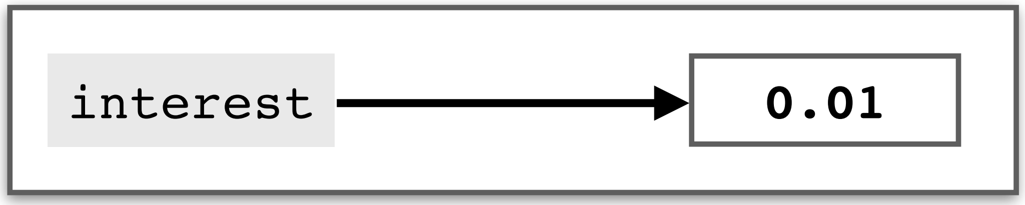
- Memorizing results as Python objects
  - Variables and assignment statements

```
interest = 0.01
deposit = 200
withdrawal = 350

balance = 1000
balance = balance * (1+interest)
balance = balance + deposit
balance = balance - withdrawal

print(balance)
```

```
860.0
```



Basics of Python Programming

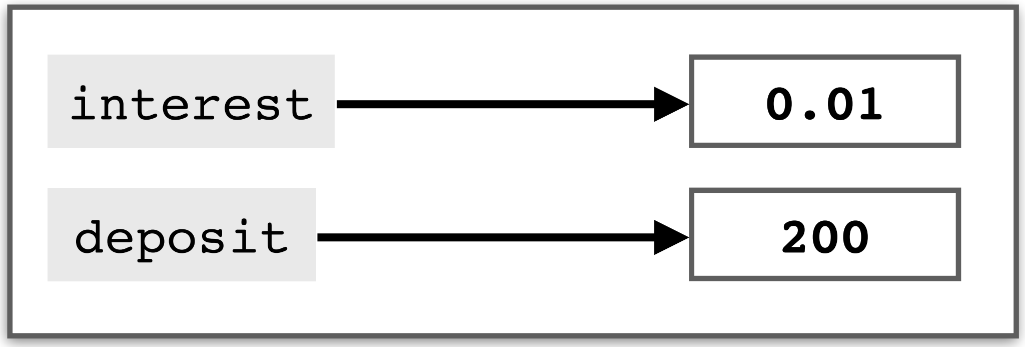
- Memorizing results as Python objects
  - Variables and assignment statements

```python
interest = 0.01
deposit = 200
withdrawal = 350

balance = 1000
balance = balance * (1+interest)
balance = balance + deposit
balance = balance - withdrawal

print(balance)
```

```
860.0
```



Basics of Python Programming

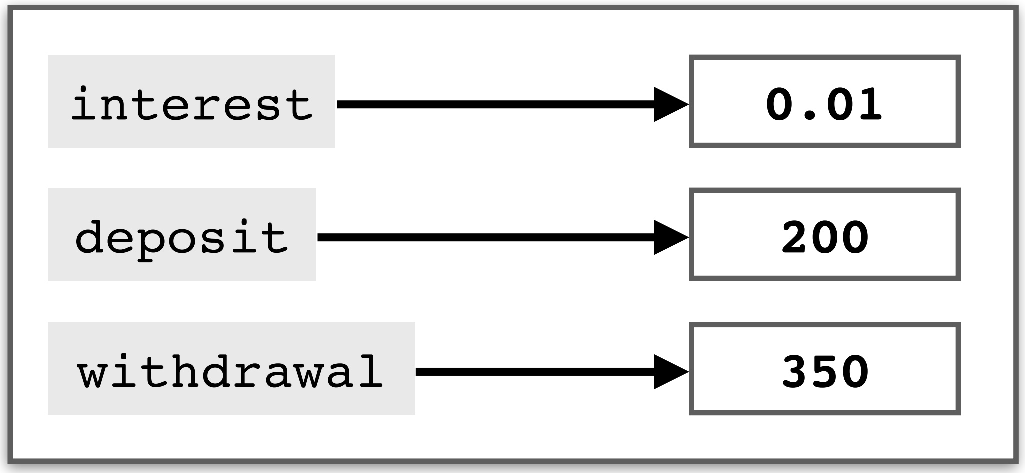
- Memorizing results as Python objects
    - Variables and assignment statements

```python
interest = 0.01
deposit = 200
withdrawal = 350

balance = 1000
balance = balance * (1+interest)
balance = balance + deposit
balance = balance - withdrawal

print(balance)
```

```
860.0
```



Basics of Python Programming

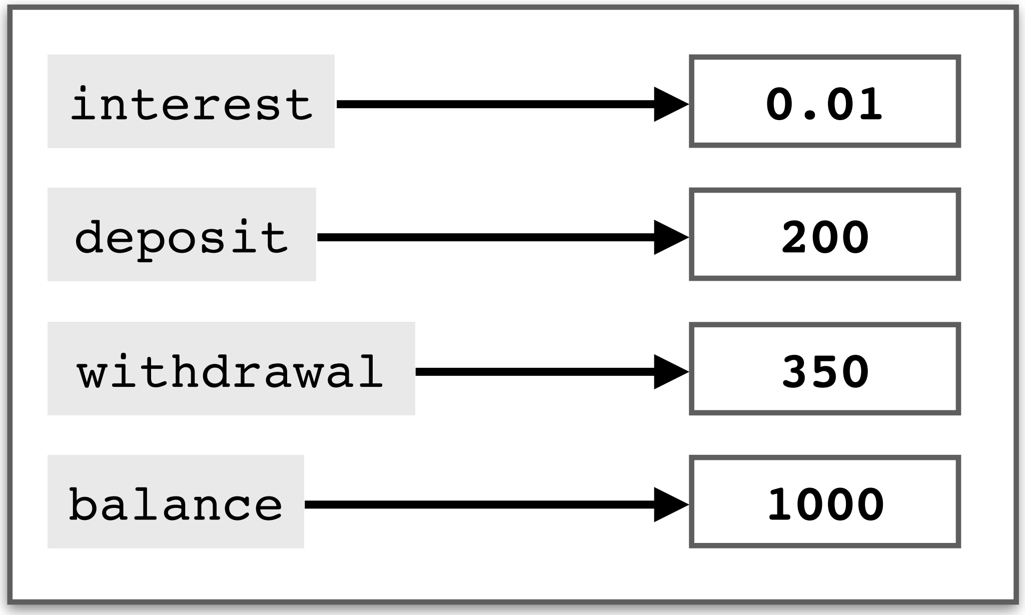
- Memorizing results as Python objects
  - Variables and assignment statements

```
interest = 0.01
deposit = 200
withdrawal = 350

balance = 1000
balance = balance * (1+interest)
balance = balance + deposit
balance = balance - withdrawal

print(balance)
```

```
860.0
```



Basics of Python Programming

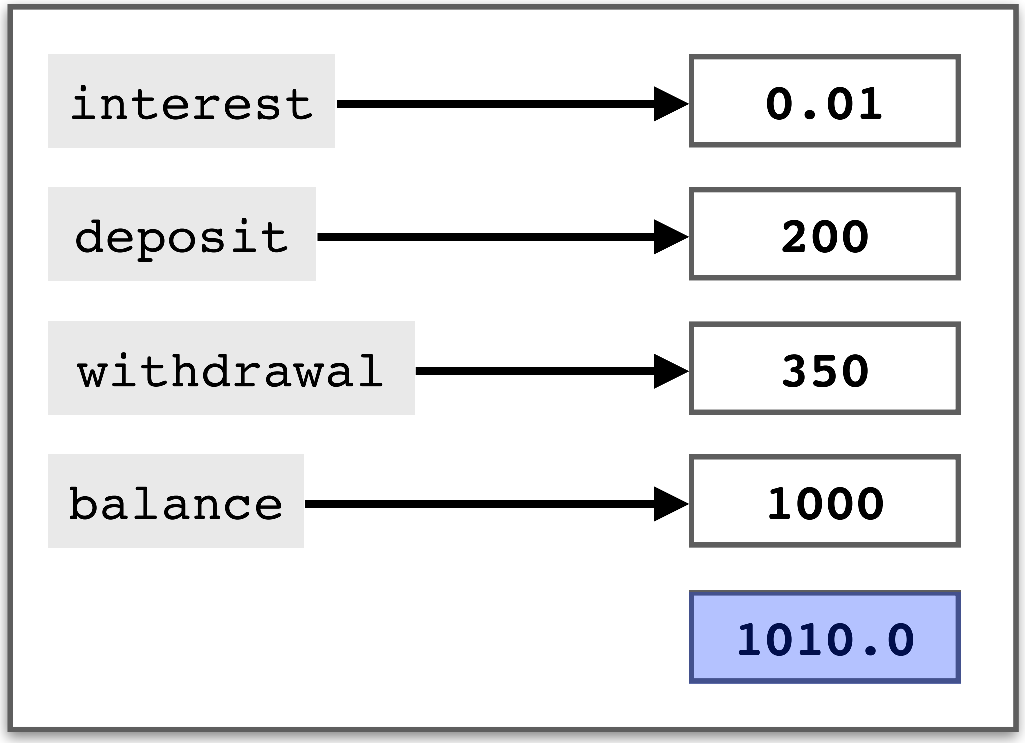
- Memorizing results as Python objects
  - Variables and assignment statements

```
interest = 0.01
deposit = 200
withdrawal = 350

balance = 1000
balance = balance * (1+interest)
balance = balance + deposit
balance = balance - withdrawal

print(balance)
```

```
860.0
```



Basics of Python Programming

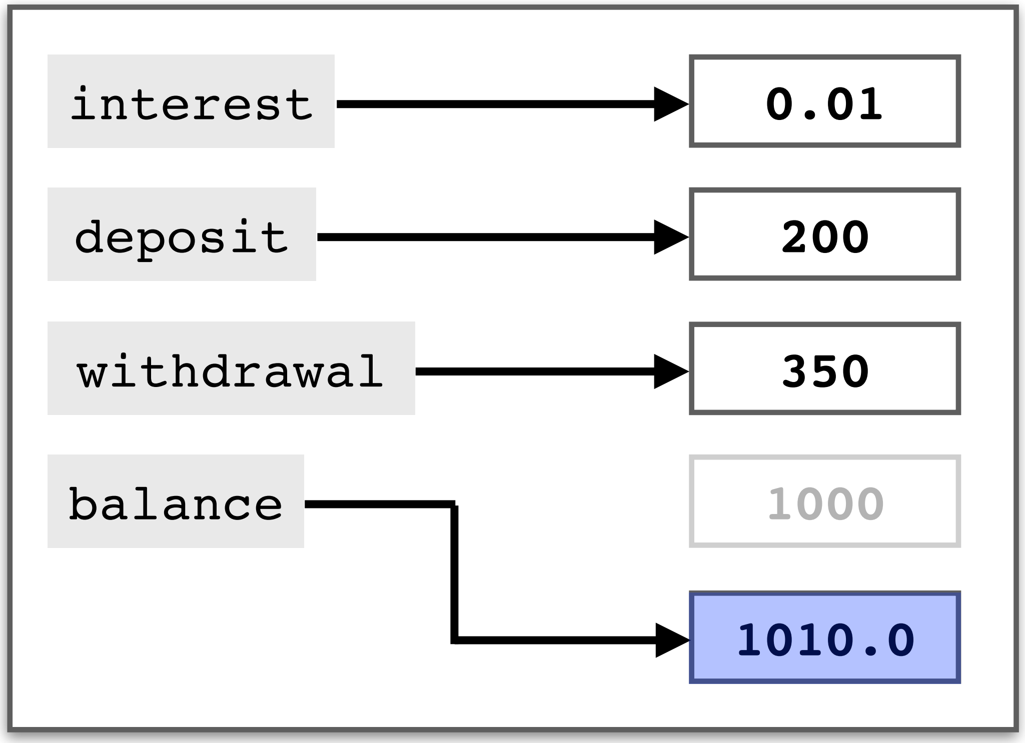
- Memorizing results as Python objects
  - Variables and assignment statements

```python
interest = 0.01
deposit = 200
withdrawal = 350

balance = 1000
balance = balance * (1+interest)
balance = balance + deposit
balance = balance - withdrawal

print(balance)
```

```
860.0
```



Basics of Python Programming

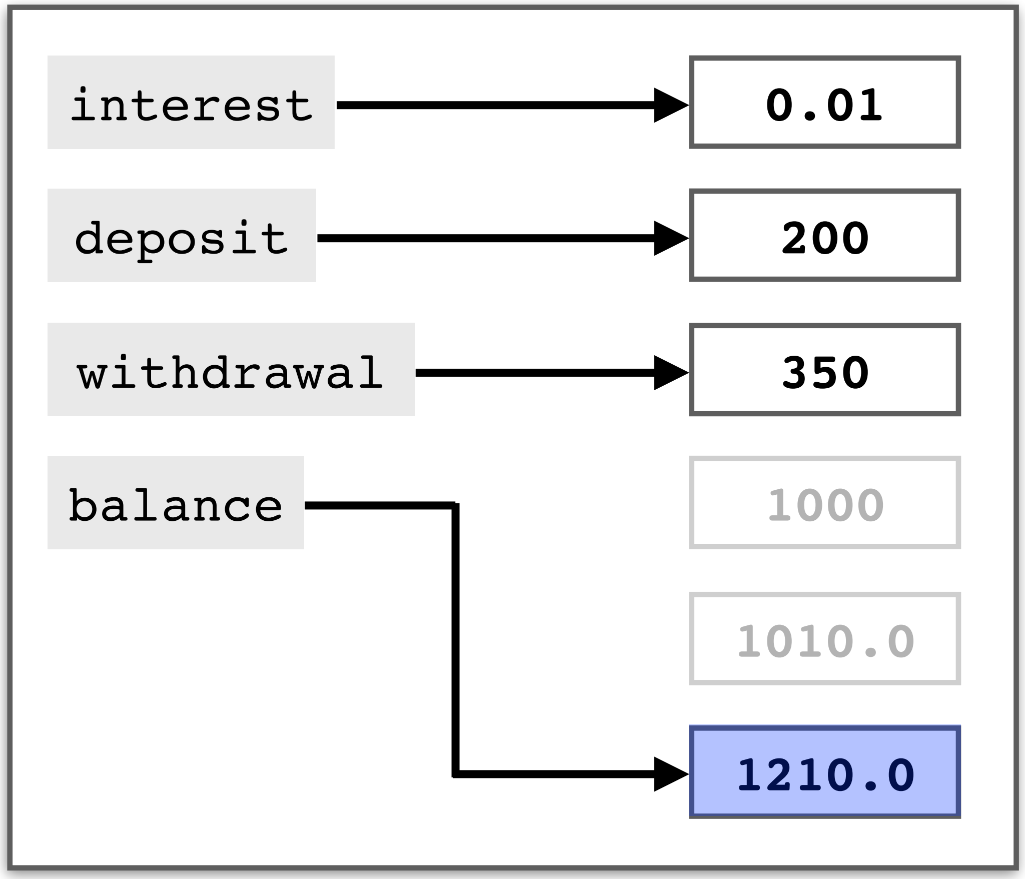
- Memorizing results as Python objects
  - Variables and assignment statements

```
interest = 0.01
deposit = 200
withdrawal = 350

balance = 1000
balance = balance * (1+interest)
balance = balance + deposit
balance = balance - withdrawal

print(balance)
```

```
860.0
```



Basics of Python Programming

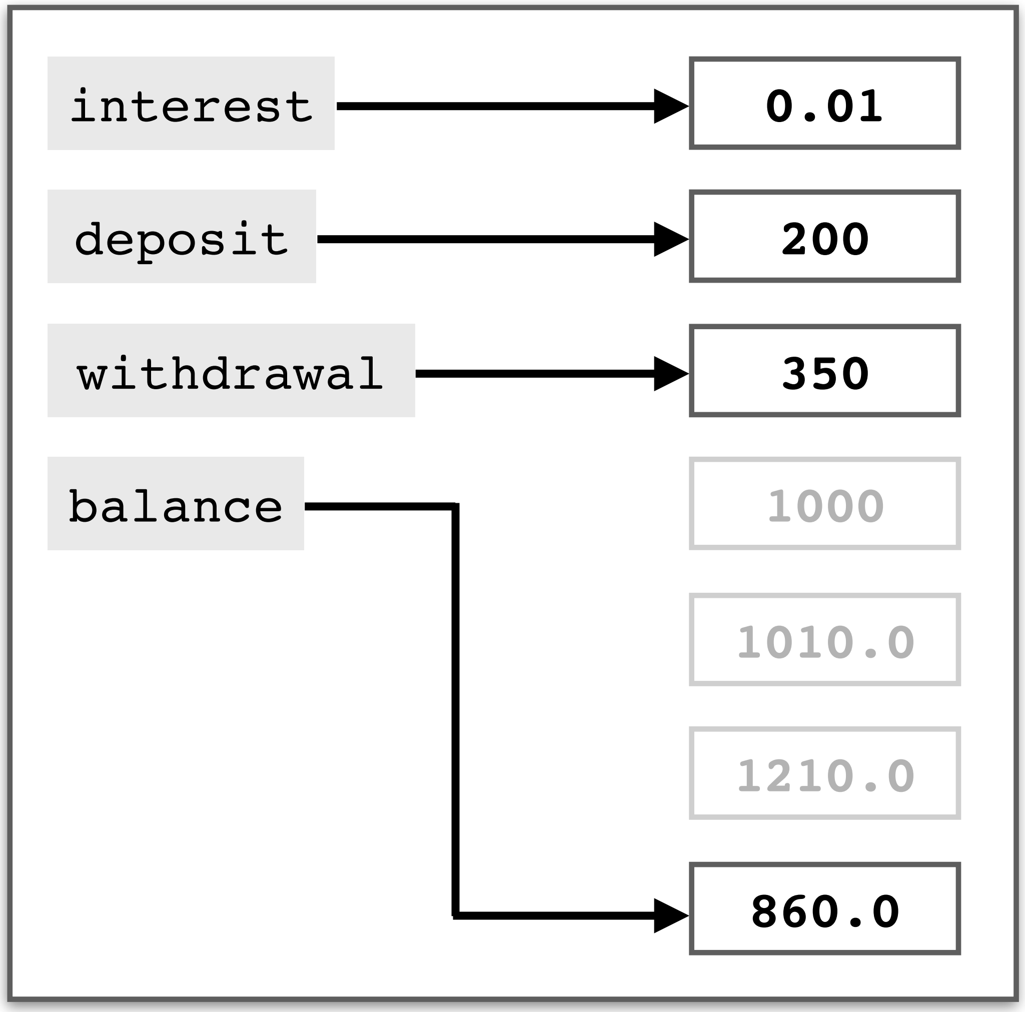
- Memorizing results as Python objects
  - Variables and assignment statements

```python
interest = 0.01
deposit = 200
withdrawal = 350

balance = 1000
balance = balance * (1+interest)
balance = balance + deposit
balance = balance - withdrawal

print(balance)
```

```
860.0
```



Basics of Python Programming

- Memorizing results as Python objects
    - Variables and assignment statements

```
interest = 0.01
deposit = 200
withdrawal = 350

balance = 1000
balance = balance * (1+interest)
balance = balance + deposit
balance = balance - withdrawal

print(balance)
```

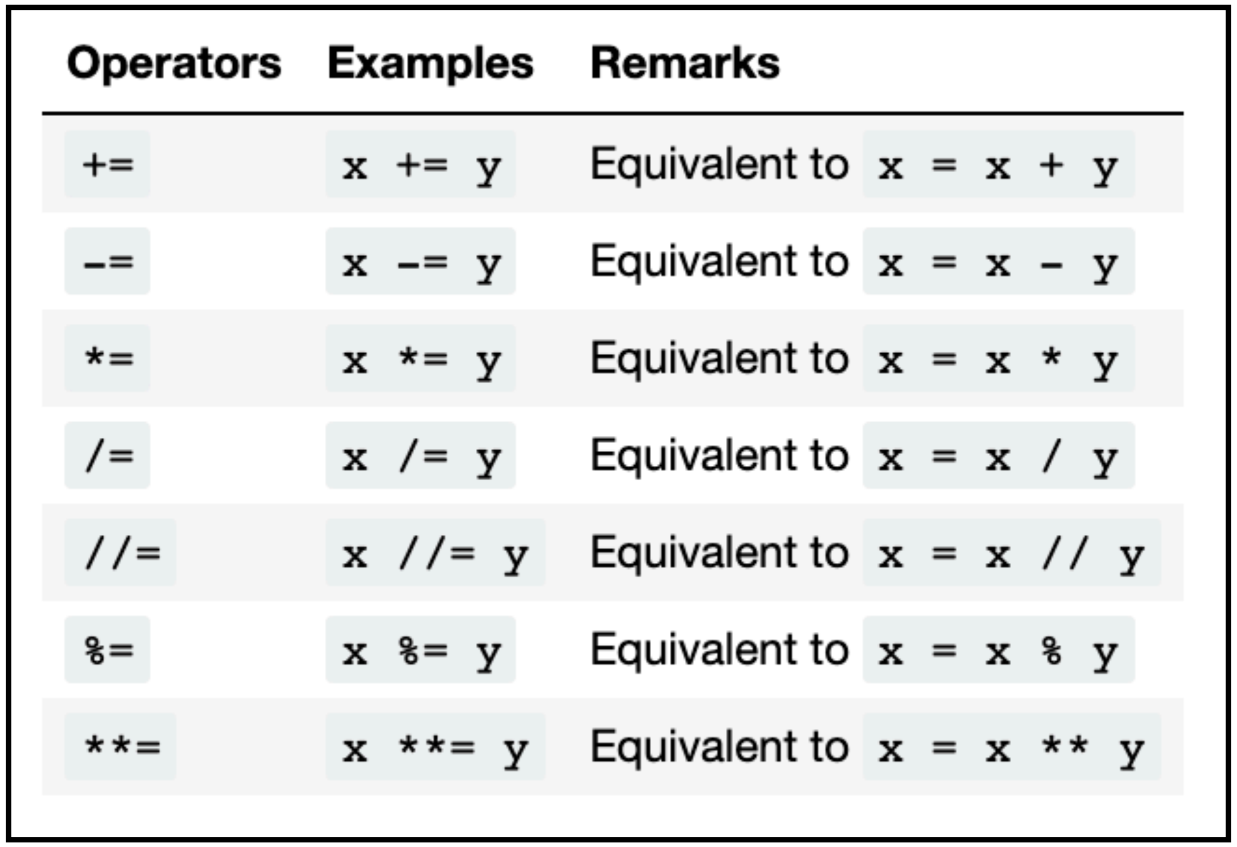
```
860.0
```

Coding Style:

- Variable names
- Whitespace in expressions and statements
- Empty lines used for better readability

Basics of Python Programming

- Memorizing results as Python objects
  - Other assignment operators



Operators	Examples	Remarks
`+=`	`x += y`	Equivalent to `x = x + y`
`-=`	`x -= y`	Equivalent to `x = x - y`
`*=`	`x *= y`	Equivalent to `x = x * y`
`/=`	`x /= y`	Equivalent to `x = x / y`
`//=`	`x //= y`	Equivalent to `x = x // y`
`%=`	`x %= y`	Equivalent to `x = x % y`
`**=`	`x **= y`	Equivalent to `x = x ** y`

Basics of Python Programming

- Memorizing results as Python objects
  - Other assignment operators

```python
interest = 0.01
deposit = 200
withdrawal = 350

balance = 1000
balance = balance * (1+interest)
balance = balance + deposit
balance = balance - withdrawal

print(balance)
```

```
860.0
```

"Lazy" version

```python
interest = 0.01
deposit = 200
withdrawal = 350

balance = 1000
balance *= 1 + interest
balance += deposit
balance -= withdrawal

print(balance)
```

```
860.0
```

Basics of Python Programming

- Memorizing results as Python objects
    - Other assignment operators

```
interest = 0.01
deposit = 200
withdrawal = 350

balance = 1000
balance = balance * (1+interest)
balance = balance + deposit
balance = balance - withdrawal

print(balance)
```

```
860.0
```

Alternative solution

```
interest = 0.01
change1 = 200
change2 = -350

balance = 1000
balance *= 1 + interest
balance += change1
balance += change2

print(balance)
```

```
860.0
```

Basics of Python Programming

- Memorizing results as Python objects

Example 2: There are two variables **`cage`** and **`travolta`** and their values are "bad guy" and "good guy", respectively. Swap the values of **`cage`** and **`travolta`** so that **`cage`** becomes "good guy" and **`travolta`** becomes "bad guy".

Basics of Python Programming

- Memorizing results as Python objects

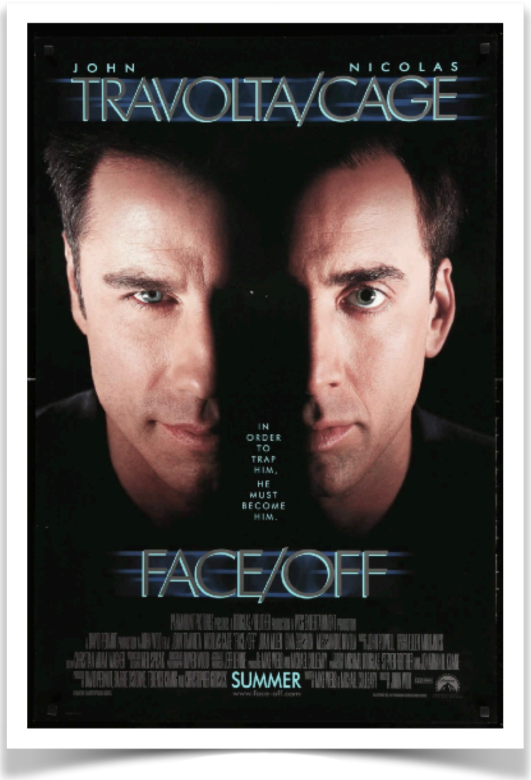
Example 2: There are two variables **`cage`** and **`travolta`** and their values are "bad guy" and "good guy", respectively. Swap the values of **`cage`** and **`travolta`** so that **`cage`** becomes "good guy" and **`travolta`** becomes "bad guy".

Incorrect solution!

```
cage = 'bad guy'
travolta = 'good guy'

cage = travolta
travolta = cage

print(cage)
print(travolta)
```



Basics of Python Programming

- Memorizing results as Python objects

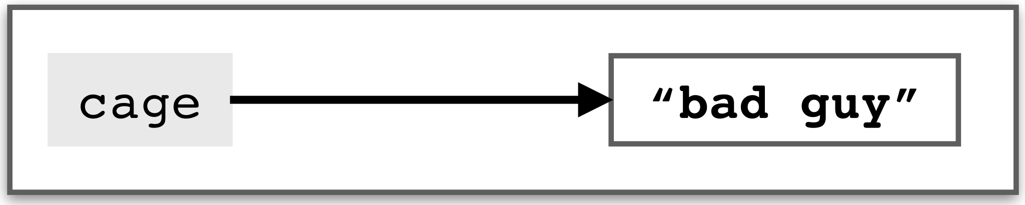
Example 2: There are two variables **`cage`** and **`travolta`** and their values are "bad guy" and "good guy", respectively. Swap the values of **`cage`** and **`travolta`** so that **`cage`** becomes "good guy" and **`travolta`** becomes "bad guy".

Incorrect solution!

```python
cage = 'bad guy'
travolta = 'good guy'

cage = travolta
travolta = cage

print(cage)
print(travolta)
```



Basics of Python Programming

- Memorizing results as Python objects

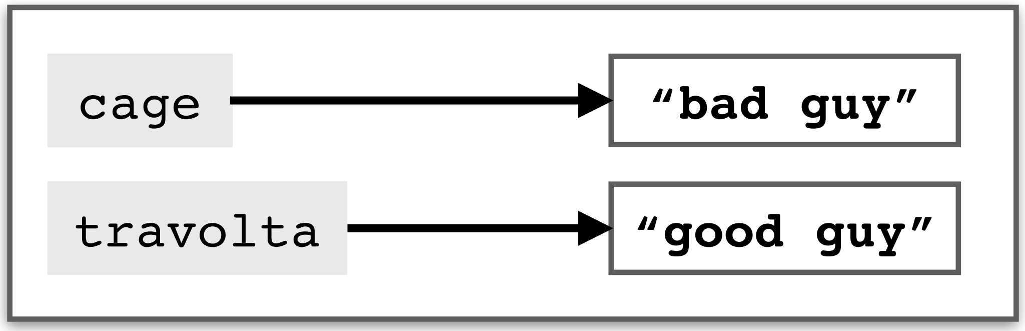
Example 2: There are two variables **`cage`** and **`travolta`** and their values are "bad guy" and "good guy", respectively. Swap the values of **`cage`** and **`travolta`** so that **`cage`** becomes "good guy" and **`travolta`** becomes "bad guy".

Incorrect solution!

```
cage = 'bad guy'
travolta = 'good guy'

cage = travolta
travolta = cage

print(cage)
print(travolta)
```



Basics of Python Programming

- Memorizing results as Python objects

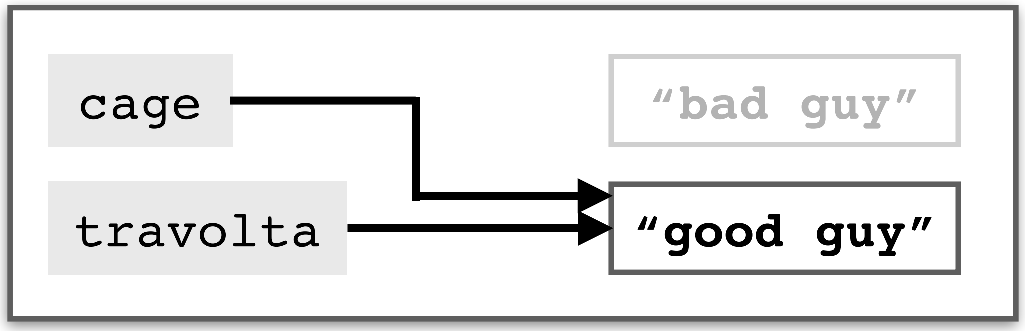
Example 2: There are two variables **`cage`** and **`travolta`** and their values are "bad guy" and "good guy", respectively. Swap the values of **`cage`** and **`travolta`** so that **`cage`** becomes "good guy" and **`travolta`** becomes "bad guy".

Incorrect solution!

```
cage = 'bad guy'
travolta = 'good guy'

cage = travolta
travolta = cage

print(cage)
print(travolta)
```



Basics of Python Programming

- Memorizing results as Python objects

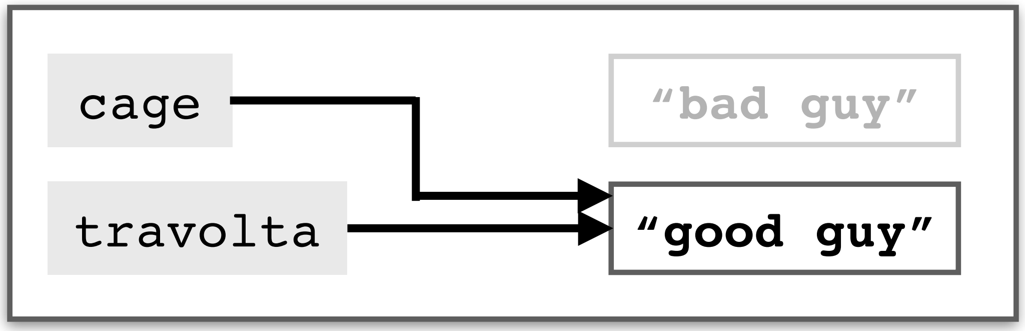
Example 2: There are two variables **`cage`** and **`travolta`** and their values are "bad guy" and "good guy", respectively. Swap the values of **`cage`** and **`travolta`** so that **`cage`** becomes "good guy" and **`travolta`** becomes "bad guy".

Incorrect solution!

```
cage = 'bad guy'
travolta = 'good guy'

cage = travolta
travolta = cage

print(cage)
print(travolta)
```



Basics of Python Programming

- Memorizing results as Python objects

Example 2: There are two variables **`cage`** and **`travolta`** and their values are "bad guy" and "good guy", respectively. Swap the values of **`cage`** and **`travolta`** so that **`cage`** becomes "good guy" and **`travolta`** becomes "bad guy".

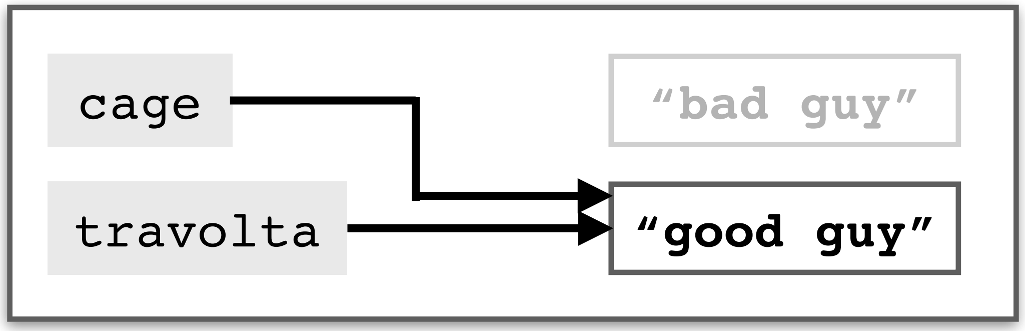
Incorrect solution!

```python
cage = 'bad guy'
travolta = 'good guy'

cage = travolta
travolta = cage

print(cage)
print(travolta)
```

```
good guy
good guy
```



Basics of Python Programming

- Memorizing results as Python objects

Example 2: There are two variables **`cage`** and **`travolta`** and their values are "bad guy" and "good guy", respectively. Swap the values of **`cage`** and **`travolta`** so that **`cage`** becomes "good guy" and **`travolta`** becomes "bad guy".

Correct solution!

```
cage = 'bad guy'
travolta = 'good guy'

temp = cage
cage = Travolta
travolta = temp

print(cage)
print(travolta)
```

Basics of Python Programming

- Memorizing results as Python objects

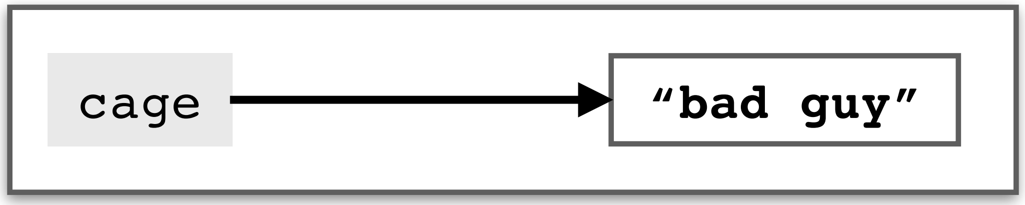
Example 2: There are two variables **`cage`** and **`travolta`** and their values are "bad guy" and "good guy", respectively. Swap the values of **`cage`** and **`travolta`** so that **`cage`** becomes "good guy" and **`travolta`** becomes "bad guy".

Correct solution!

```
cage = 'bad guy'
travolta = 'good guy'

temp = cage
cage = Travolta
travolta = temp

print(cage)
print(travolta)
```



Basics of Python Programming

- Memorizing results as Python objects

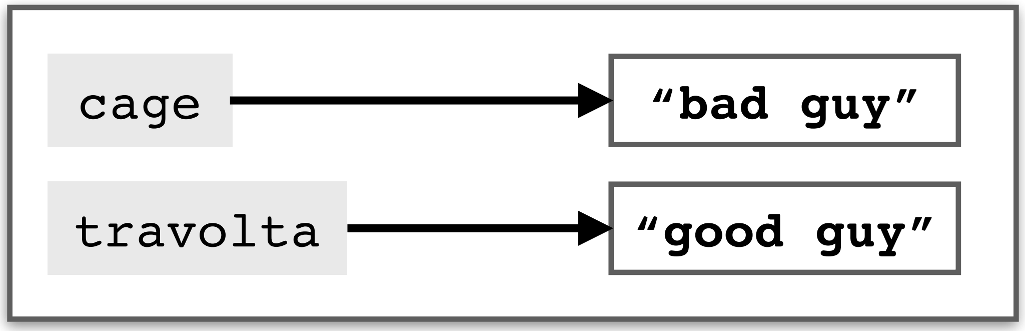
Example 2: There are two variables **`cage`** and **`travolta`** and their values are "bad guy" and "good guy", respectively. Swap the values of **`cage`** and **`travolta`** so that **`cage`** becomes "good guy" and **`travolta`** becomes "bad guy".

Correct solution!

```
cage = 'bad guy'
travolta = 'good guy'

temp = cage
cage = Travolta
travolta = temp

print(cage)
print(travolta)
```


$$\begin{tikzcd}
\texttt{cage} \arrow[r] & \boxed{\texttt{“bad guy”}} \\
\texttt{travolta} \arrow[r] & \boxed{\texttt{“good guy”}}
\end{tikzcd}$$

Basics of Python Programming

- Memorizing results as Python objects

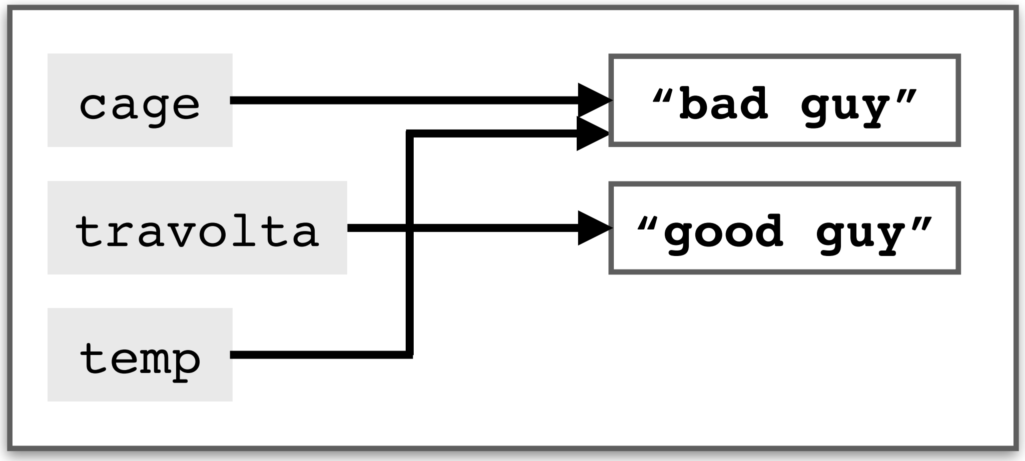
Example 2: There are two variables **`cage`** and **`travolta`** and their values are "bad guy" and "good guy", respectively. Swap the values of **`cage`** and **`travolta`** so that **`cage`** becomes "good guy" and **`travolta`** becomes "bad guy".

Correct solution!

```
cage = 'bad guy'
travolta = 'good guy'

temp = cage
cage = Travolta
travolta = temp

print(cage)
print(travolta)
```



Basics of Python Programming

- Memorizing results as Python objects

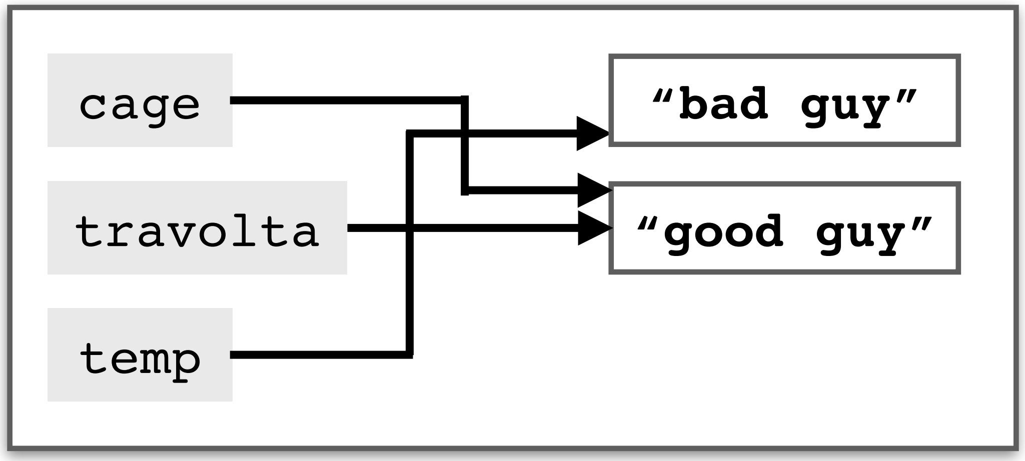
Example 2: There are two variables **`cage`** and **`travolta`** and their values are "bad guy" and "good guy", respectively. Swap the values of **`cage`** and **`travolta`** so that **`cage`** becomes "good guy" and **`travolta`** becomes "bad guy".

Correct solution!

```
cage = 'bad guy'
travolta = 'good guy'

temp = cage
cage = Travolta
travolta = temp

print(cage)
print(travolta)
```



Basics of Python Programming

- Memorizing results as Python objects

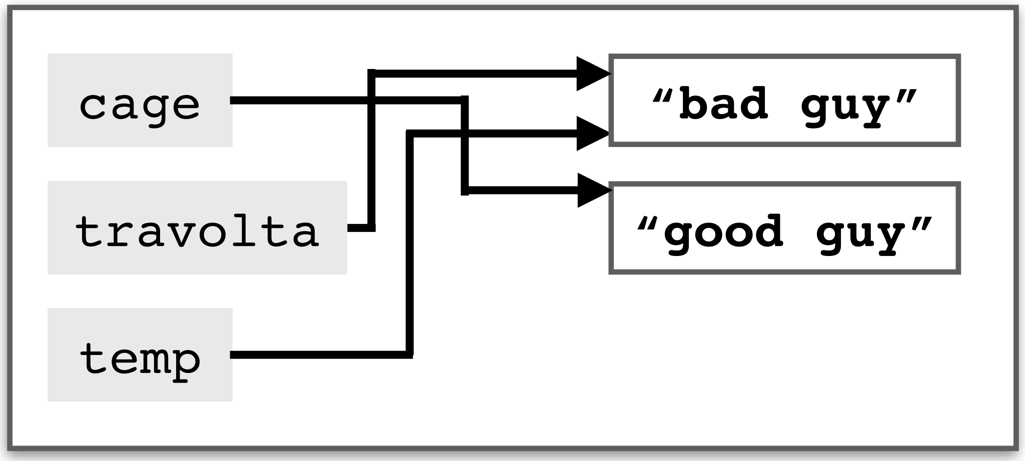
Example 2: There are two variables **`cage`** and **`travolta`** and their values are "bad guy" and "good guy", respectively. Swap the values of **`cage`** and **`travolta`** so that **`cage`** becomes "good guy" and **`travolta`** becomes "bad guy".

Correct solution!

```
cage = 'bad guy'
travolta = 'good guy'

temp = cage
cage = Travolta
travolta = temp

print(cage)
print(travolta)
```



Basics of Python Programming

- Memorizing results as Python objects

Example 2: There are two variables **`cage`** and **`travolta`** and their values are "bad guy" and "good guy", respectively. Swap the values of **`cage`** and **`travolta`** so that **`cage`** becomes "good guy" and **`travolta`** becomes "bad guy".

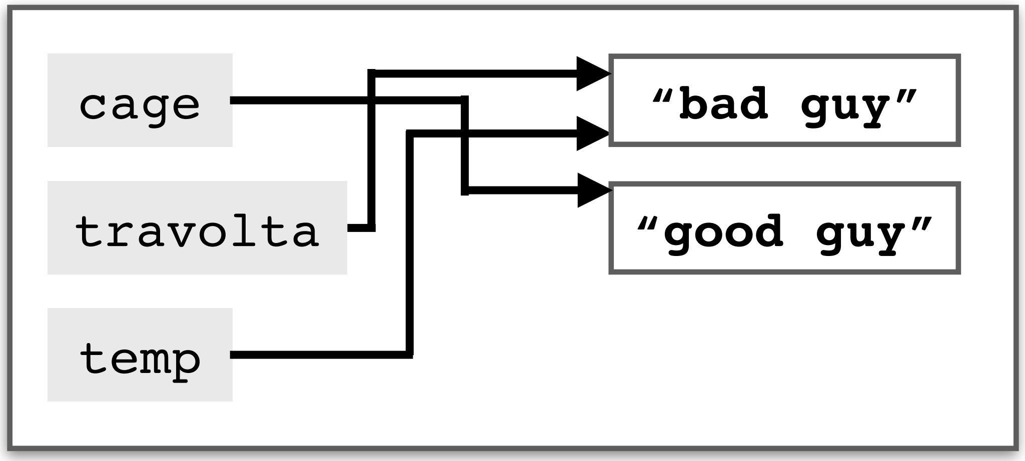
Correct solution!

```
cage = 'bad guy'
travolta = 'good guy'

temp = cage
cage = Travolta
travolta = temp

print(cage)
print(travolta)
```

```
good guy
bad guy
```



Basics of Python Programming

- Data type conversions
    - Enable other operations by another data type
    - Use the type name as the data type conversion function

Basics of Python Programming

- Data type conversions
  - Enable other operations by another data type
  - Use the type name as the data type conversion function

```python
num = '5'
print(type(num))

num_int = int(num)
print(type(num_int))

num_float = float(num)
print(type(num_float))
```

```
<class 'str'>
<class 'int'>
<class 'float'>
```

The variable `num` is a `str` type object

```python
print(num * 3)
print(num_int * 3)
print(num_float * 3)
```

```
555
15
15.0
```

Duplicate the string with the multiplication operator

Basics of Python Programming

- Data type conversions
  - Enable other operations by another data type
  - Use the type name as the data type conversion function

```
num = '5'
print(type(num))

num_int = int(num)
print(type(num_int))

num_float = float(num)
print(type(num_float))
```

```
<class 'str'>
<class 'int'>
<class 'float'>
```

The variable `num_int` is created to be an `int` type object

```
print(num * 3)
print(num_int * 3)
print(num_float * 3)
```

```
555
15
15.0
```

Numeric multiplication on the integer object

Basics of Python Programming

- Data type conversions
    - Enable other operations by another data type
    - Use the type name as the data type conversion function

```python
num = '5'
print(type(num))

num_int = int(num)
print(type(num_int))

num_float = float(num)
print(type(num_float))
```

```
<class 'str'>
<class 'int'>
<class 'float'>
```

The variable `num_float` is created to be a `float` type object

```python
print(num * 3)
print(num_int * 3)
print(num_float * 3)
```

```
555
15
15.0
```

Numeric multiplication on the floating point number

Basics of Python Programming

- Data type conversions

Example 3: Write a program that allows the user to input a temperature in Celsius, then convert the value to Fahrenheit temperature. The equation for the conversion is $T_{\text{Fahrenheit}} = T_{\text{Celsius}} \times 1.8 + 32$.

```
temp_c_str = input('Celsius: ')
temp_c = float(temp_c_str)
temp_f = temp_c*1.8 + 32
print('Fahrenheit: ' + str(temp_f))
```

```
Celsius: 36.5
Fahrenheit: 97.7
```

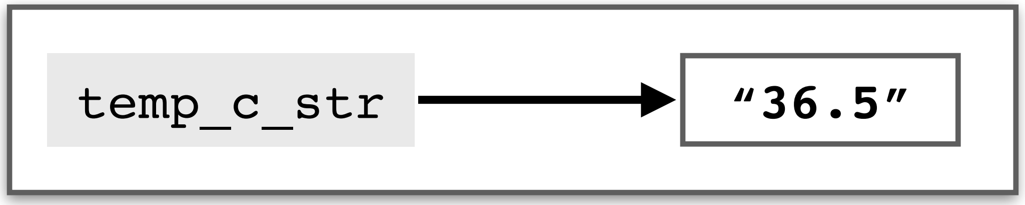
Basics of Python Programming

- Data type conversions

Example 3: Write a program that allows the user to input a temperature in Celsius, then convert the value to Fahrenheit temperature. The equation for the conversion is $T_{\text{Fahrenheit}} = T_{\text{Celsius}} \times 1.8 + 32$.

```python
temp_c_str = input('Celsius: ')
temp_c = float(temp_c_str)
temp_f = temp_c*1.8 + 32
print('Fahrenheit: ' + str(temp_f))
```

```
Celsius: 36.5
Fahrenheit: 97.7
```

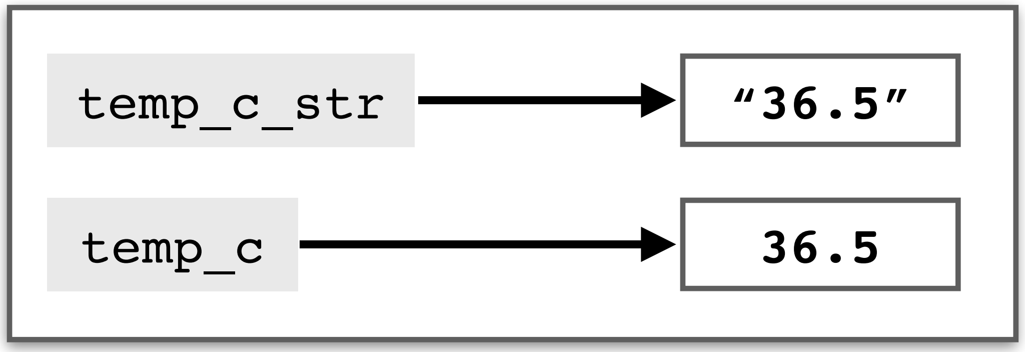


Basics of Python Programming

- Data type conversions

Example 3: Write a program that allows the user to input a temperature in Celsius, then convert the value to Fahrenheit temperature. The equation for the conversion is $T_{\text{Fahrenheit}} = T_{\text{Celsius}} \times 1.8 + 32$.

```
temp_c_str = input('Celsius: ')
temp_c = float(temp_c_str)
temp_f = temp_c*1.8 + 32
print('Fahrenheit: ' + str(temp_f))
```



```
Celsius: 36.5
Fahrenheit: 97.7
```

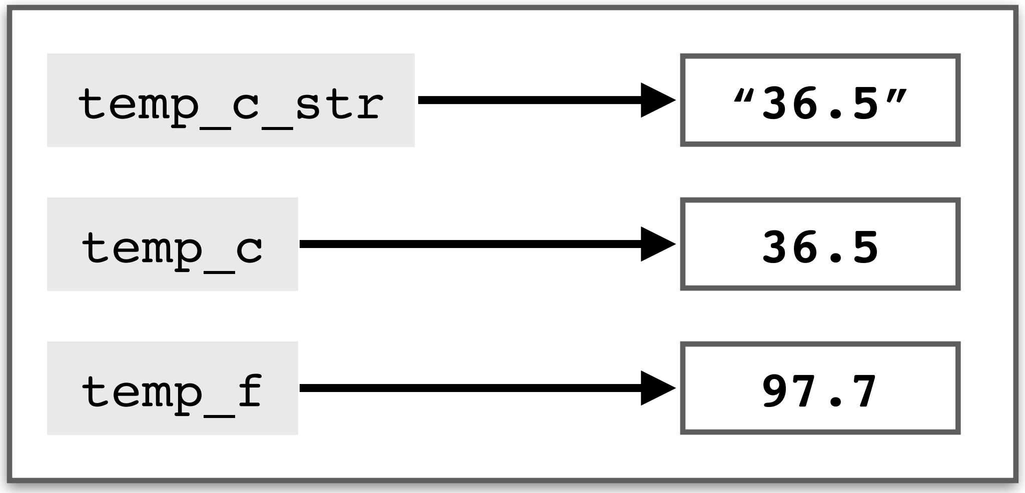
Basics of Python Programming

- Data type conversions

Example 3: Write a program that allows the user to input a temperature in Celsius, then convert the value to Fahrenheit temperature. The equation for the conversion is $T_{\text{Fahrenheit}} = T_{\text{Celsius}} \times 1.8 + 32$.

```python
temp_c_str = input('Celsius: ')
temp_c = float(temp_c_str)
temp_f = temp_c*1.8 + 32
print('Fahrenheit: ' + str(temp_f))
```

```
Celsius: 36.5
Fahrenheit: 97.7
```



Basics of Python Programming

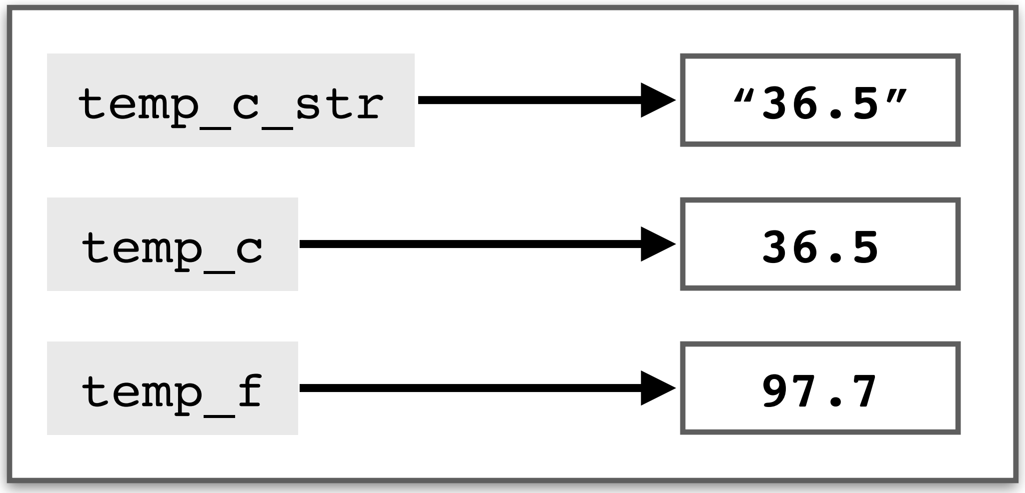
- Data type conversions

Example 3: Write a program that allows the user to input a temperature in Celsius, then convert the value to Fahrenheit temperature. The equation for the conversion is $T_{\text{Fahrenheit}} = T_{\text{Celsius}} \times 1.8 + 32$.

```python
temp_c_str = input('Celsius: ')
temp_c = float(temp_c_str)
temp_f = temp_c*1.8 + 32
print('Fahrenheit: ' + str(temp_f))
```

```
Celsius: 36.5
Fahrenheit: 97.7
```

Convert the number into `str` type for string concatenation



Basics of Python Programming

- Data type conversions

Question 1: Write a program that ask the user to input a word and an integer, then it will print the word repeated by the given number of times. For example, if the user inputs "Go" and "5", then the printed message will be "Go Go Go Go Go”.

Basics of Python Programming

- Data type conversions

Question 1: Write a program that ask the user to input a word and an integer, then it will print the word repeated by the given number of times. For example, if the user inputs "Go" and "5", then the printed message will be "Go Go Go Go Go”.

```
word = input('Key in a word: ')
repeat = input('Key in an integer: ')

print((word + ' ') * int(repeat))
```

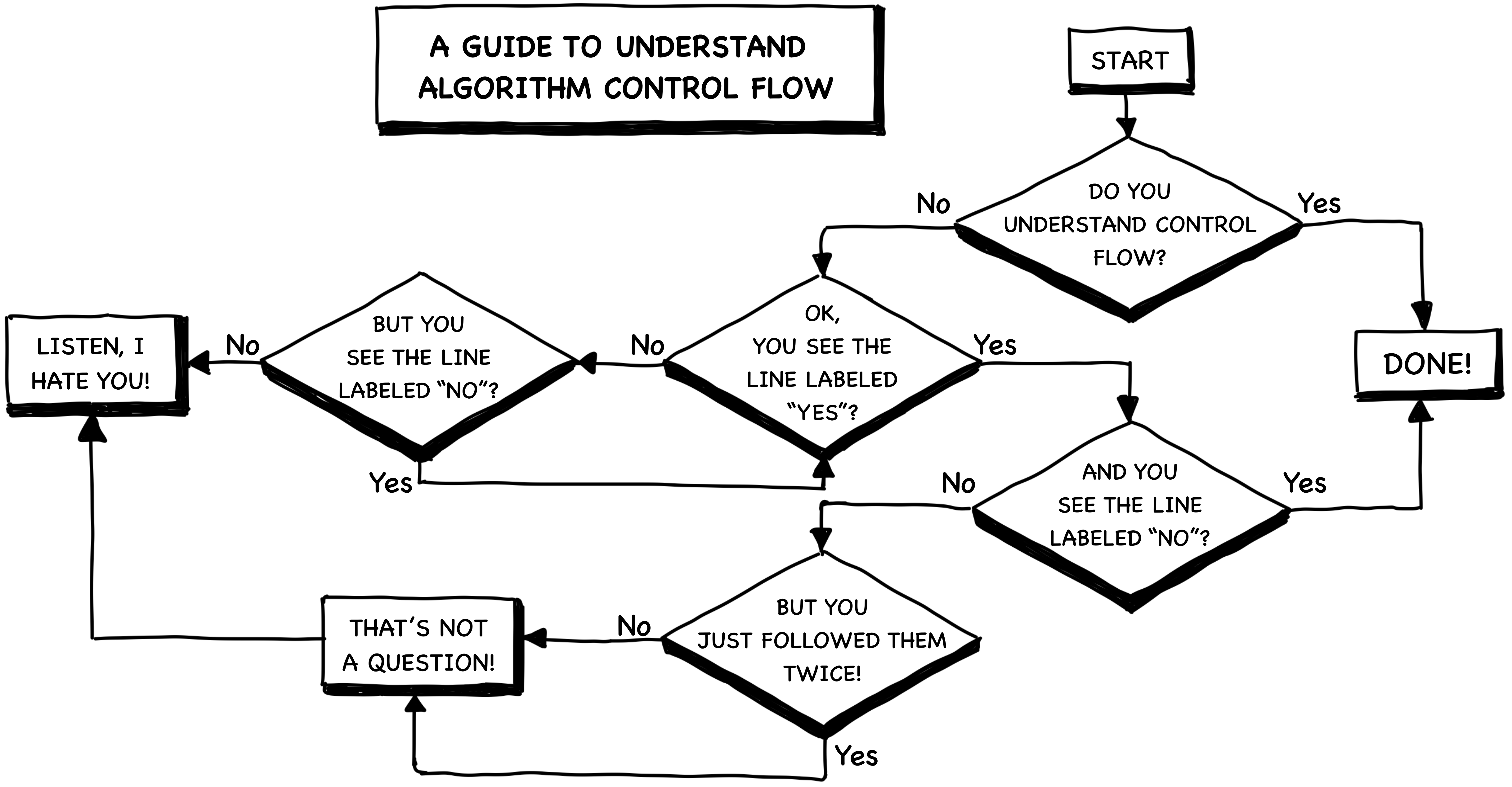
```
Key in a word: Go
Key in an integer: 5
Go Go Go Go Go
```

Introduction to Control Flows

- Computational thinking
  - ‣ What does a computer do
    - ✓ Perform calculations
    - ✓ Memorize results
  - ‣ Control flows: the order in which the program’s code executes
    - ✓ Conditional statements
    - ✓ Loops

Introduction to Control Flows

- Computational thinking

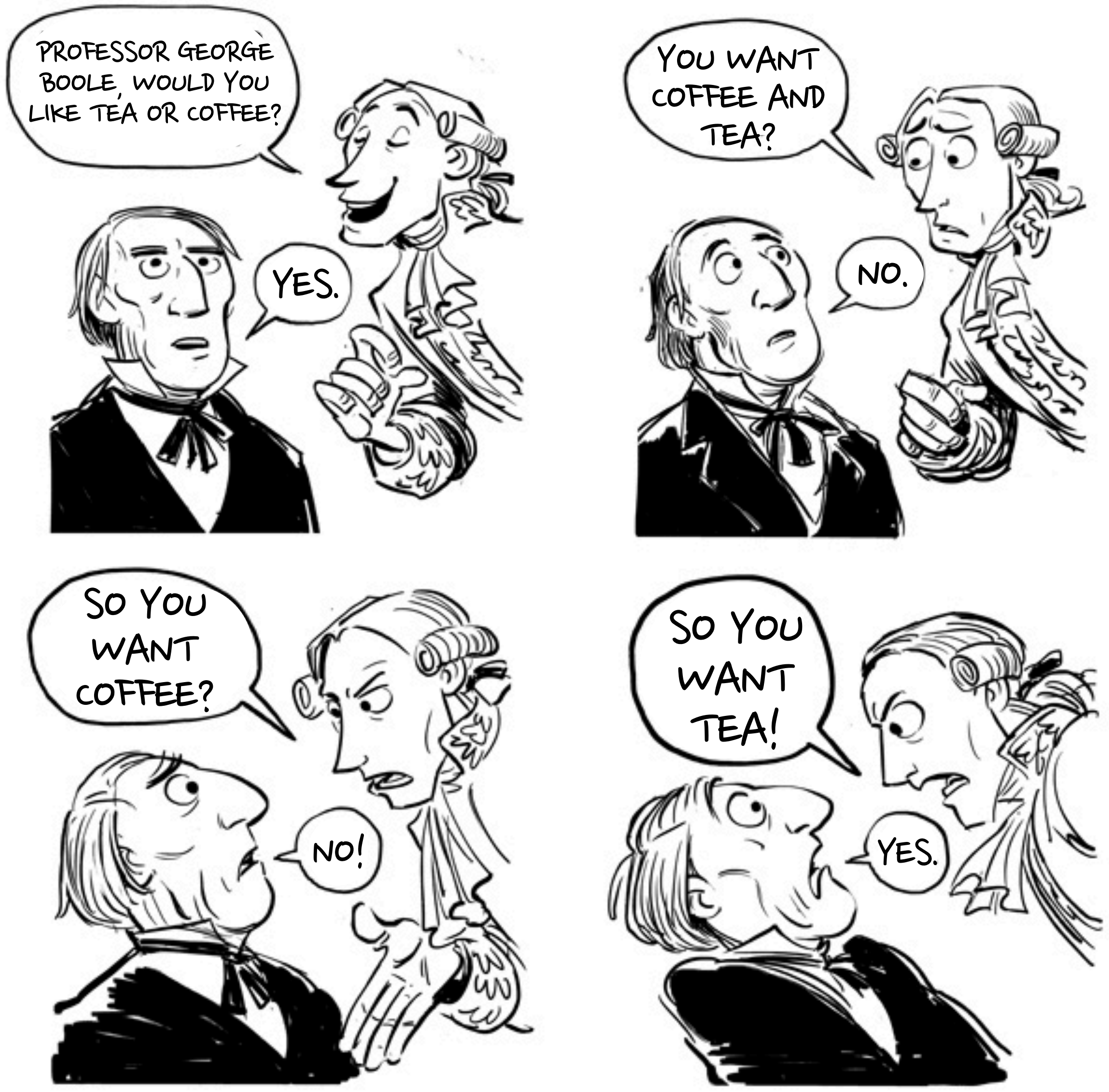

A GUIDE TO UNDERSTAND ALGORITHM CONTROL FLOW
START
DO YOU UNDERSTAND CONTROL FLOW?
No
Yes
DONE!
OK, YOU SEE THE LINE LABELED "YES"?
No
Yes
BUT YOU SEE THE LINE LABELED "NO"?
No
Yes
LISTEN, I HATE YOU!
AND YOU SEE THE LINE LABELED "NO"?
No
Yes
BUT YOU JUST FOLLOWED THEM TWICE!
No
Yes
THAT'S NOT A QUESTION!

Introduction to Control Flows

- Boolean type expressions

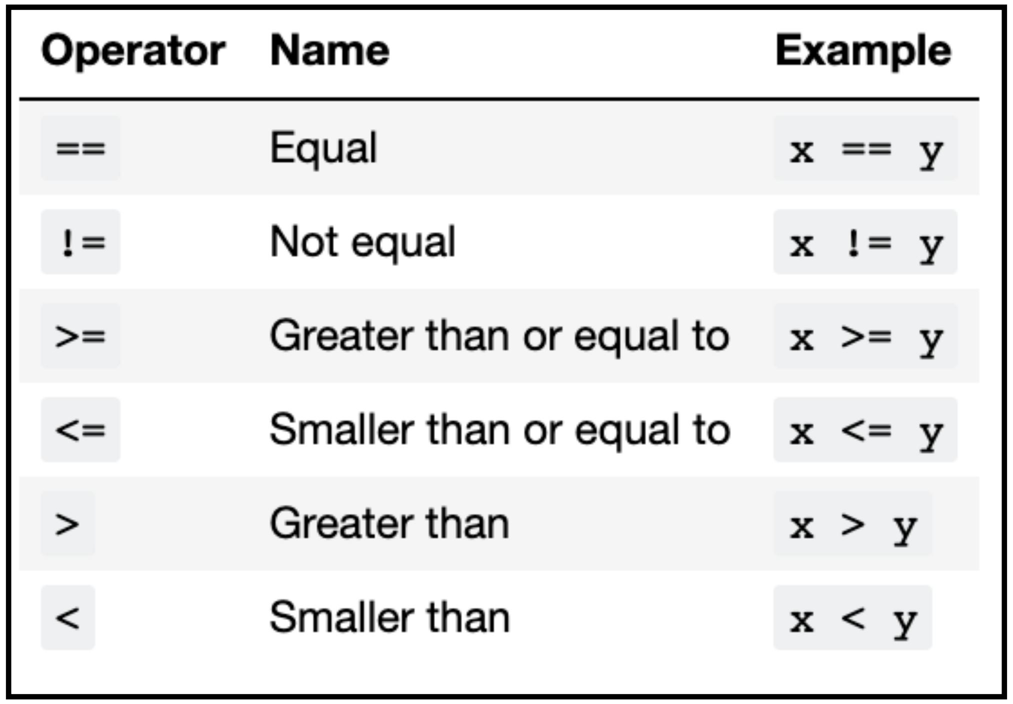
```python
print(type(True))
print(type(False))
```

```
<class 'bool'>
<class 'bool'>
```



Introduction to Control Flows

- Boolean type expressions
  - Comparison operators



Operator	Name	Example
`==`	Equal	`x == y`
`!=`	Not equal	`x != y`
`>=`	Greater than or equal to	`x >= y`
`<=`	Smaller than or equal to	`x <= y`
`>`	Greater than	`x > y`
`<`	Smaller than	`x < y`

Introduction to Control Flows

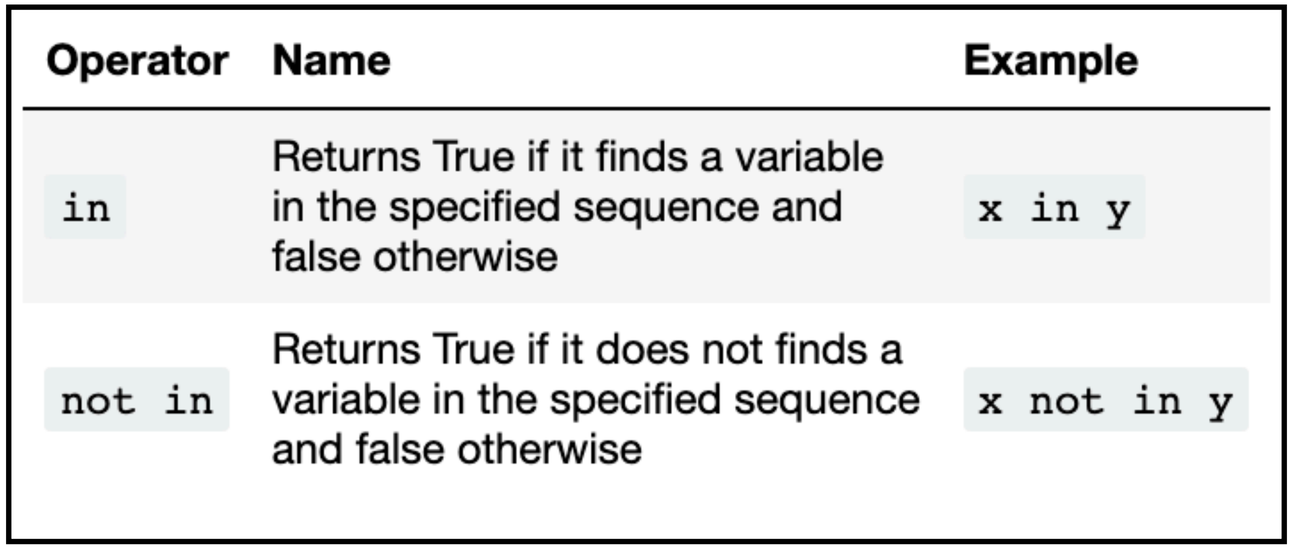
- Boolean type expressions
  - Comparison operators

```python
print(2 <= 3)         # 2 <= 3 is True
print(3.5 > 4)        # 3.5 > 4 is False
print(2 == 2.0)       # 2 == 2.0 is True
print(2 != 2.0)       # 2 != 2.0 is False
```

```
True
False
True
False
```

Introduction to Control Flows

- Boolean type expressions
  - Membership operators



Operator	Name	Example
`in`	Returns True if it finds a variable in the specified sequence and false otherwise	`x in y`
`not in`	Returns True if it does not finds a variable in the specified sequence and false otherwise	`x not in y`

Introduction to Control Flows

- Boolean type expressions
  - Membership operators

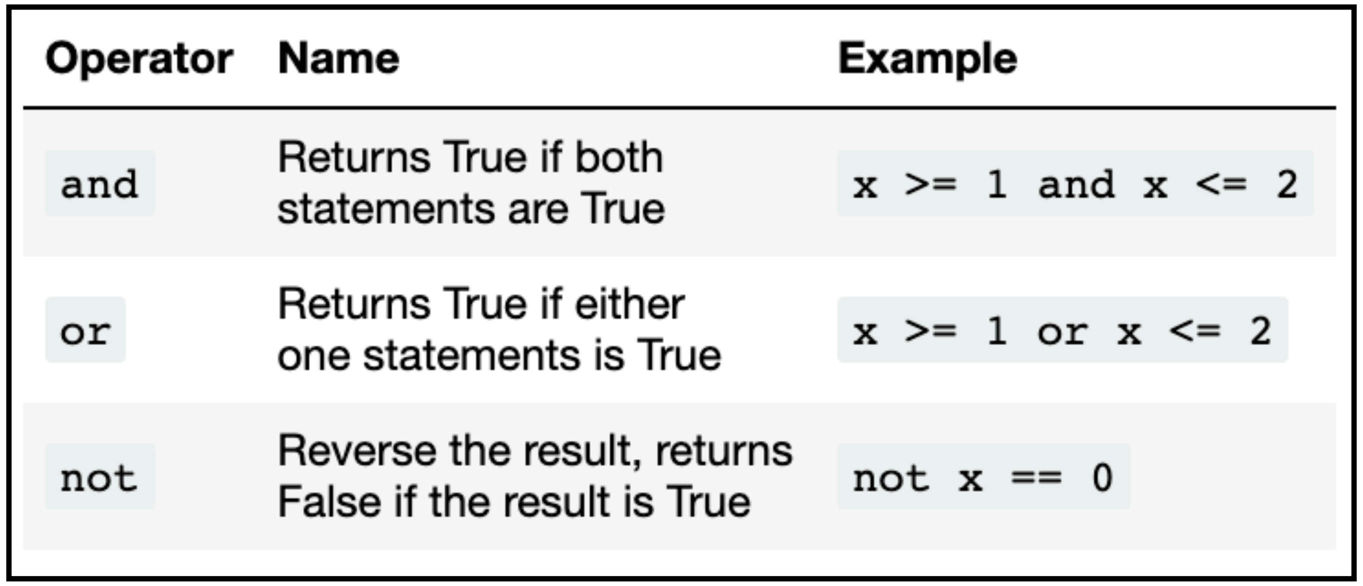
```python
line = "All work and no play makes Jack a dull boy."

print('work' in line)        # True as "work" is in line
print('Work' in line)        # False as "Work" is not in line
print('John' not in line)    # True as "John" is not in line
```

```
True
False
True
```

Introduction to Control Flows

- Boolean type expressions
  - Logic operators



Operator	Name	Example
`and`	Returns True if both statements are True	`x >= 1 and x <= 2`
`or`	Returns True if either one statements is True	`x >= 1 or x <= 2`
`not`	Reverse the result, returns False if the result is True	`not x == 0`

Introduction to Control Flows

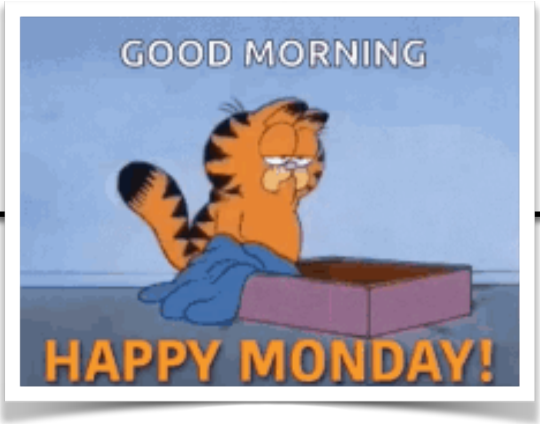
- Boolean type expressions
  - Logic operators

```
is_monday = True
is_public_holiday = False
is_weekends = False                 False          False

no_school = is_public_holiday or is_weekends
stressful_day = is_monday and not is_public_holiday

print(no_school)                    True           True
print(stressful_day)
```

```
False
True
```



Introduction to Control Flows

- Boolean type expressions

Question 2:

- Given **`x`** as a number, create a boolean expression that is **`True`** if **`x`** is no smaller than 0 and no larger than 10, and **`False`** otherwise.

Introduction to Control Flows

- Boolean type expressions

Question 2:

- Given **`x`** as a number, create a boolean expression that is **`True`** if **`x`** is no smaller than 0 and no larger than 10, and **`False`** otherwise.

```
x = 11.0

x >= 0 and x <= 10
```

```
False
```

Introduction to Control Flows

- Boolean type expressions

Question 2:

- Given **`x`** as a positive integer, create a boolean expression that is **`True`** if **`x`** is an even number, and is **`False`** if **`x`** is an odd number.

Introduction to Control Flows

- Boolean type expressions

Question 2:
- Given **`x`** as a positive integer, create a boolean expression that is **`True`** if **`x`** is an even number, and is **`False`** if **`x`** is an odd number.

```
x = 10

x % 2 == 0
```

```
True
```

Introduction to Control Flows

- Boolean type expressions

Question 2:

- Given `s` as a string in lower cases, create a boolean expression that is `True` if it contains "a" and "b", otherwise the boolean expression is `False`.

Introduction to Control Flows

- Boolean type expressions

Question 2:

- Given **`s`** as a string in lower cases, create a boolean expression that is **`True`** if it contains "a" and "b", otherwise the boolean expression is **`False`**.

```
s = 'about'

'a' in s and 'b' in s
```

```
True
```

Introduction to Control Flows

- Boolean type expressions

Question 2:

- Given `x` as a number, create a boolean expression that is `True` if it contains the digit 3 or 5, otherwise the boolean expression is `False`.

Introduction to Control Flows

- Boolean type expressions

Question 2:
- Given **`x`** as a number, create a boolean expression that is **`True`** if it contains the digit 3 or 5, otherwise the boolean expression is **`False`**.

```python
x = 25.67

'3' in str(x) or '5' in str(x)
```

```
True
```

Introduction to Control Flows

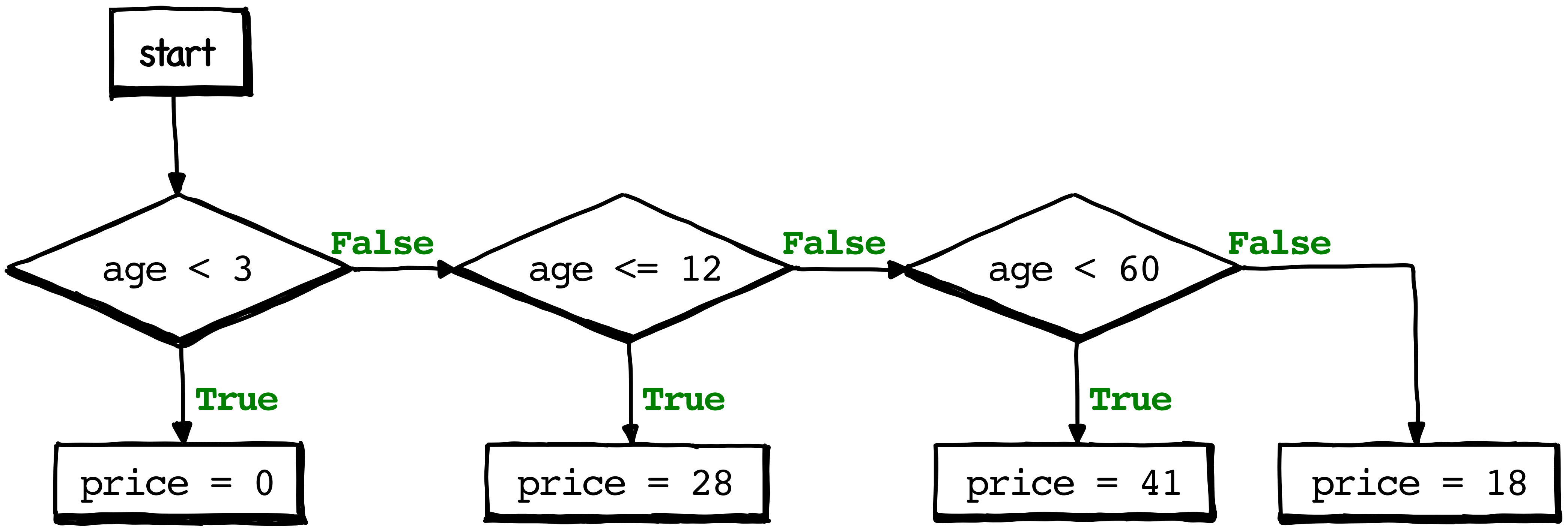
- Problem-solving with algorithms

Example 4: The price of ticket for admission to Singapore Zoo is $41 for adults, and $18 for senior citizens, who are 60 years old or above. The ticket price for children who aged 3 to 12 years old, is $28, and children under 3 years old can enjoy a free admission to the zoo. Given a visitor's age, draw the flowchart to determine the ticket price.

Introduction to Control Flows

- Problem-solving with algorithms

Example 4: The price of ticket for admission to Singapore Zoo is $41 for adults, and $18 for senior citizens, who are 60 years old or above. The ticket price for children who aged 3 to 12 years old, is $28, and children under 3 years old can enjoy a free admission to the zoo. Given a visitor's age, draw the flowchart to determine the ticket price.



Introduction to Control Flows

- Problem-solving with algorithms

Question 3: According to the Gregorian calendar, the leap years are identified as:

1. The year is evenly divisible by 4 but not evenly divisible by 100;
2. Or the year is evenly divisible by 400.

Otherwise the year is a common year. Draw the flowchart that determines if a given year is a leap year or a common year.

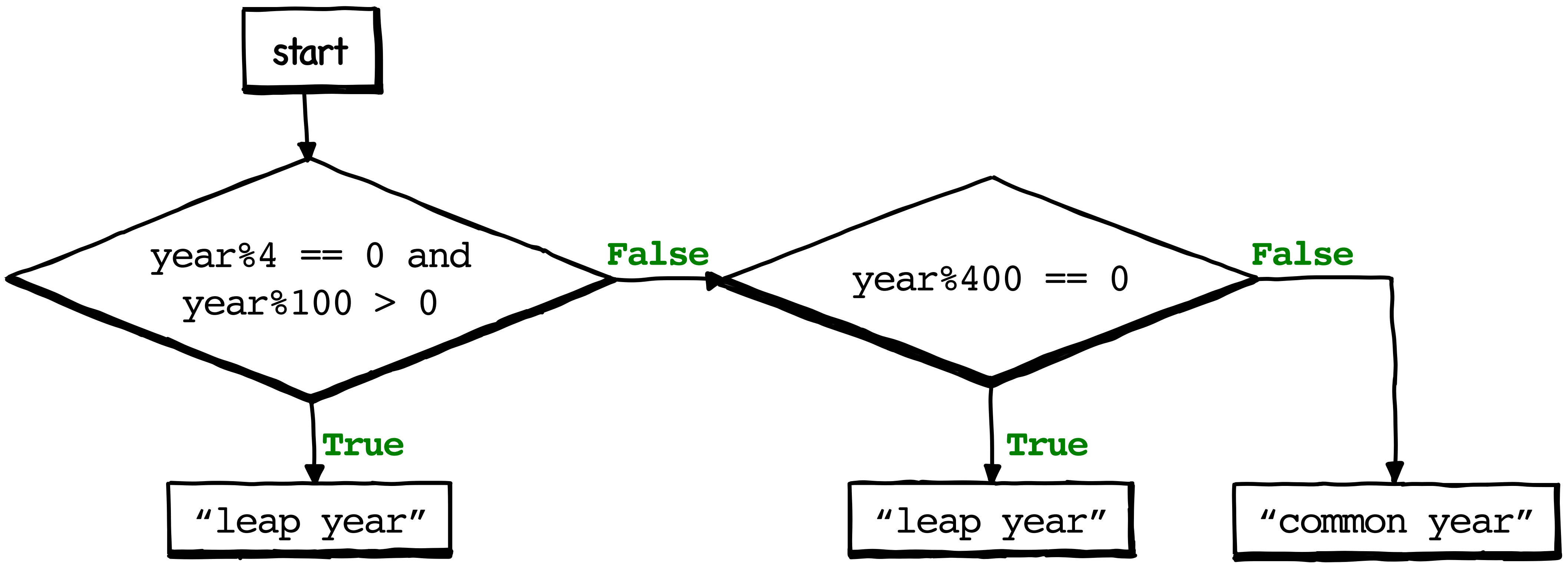
Introduction to Control Flows

- Problem-solving with algorithms

Question 3: According to the Gregorian calendar, the leap years are identified as:

1. The year is evenly divisible by 4 but not evenly divisible by 100;
2. Or the year is evenly divisible by 400.

Otherwise the year is a common year. Draw the flowchart that determines if a given year is a leap year or a common year.



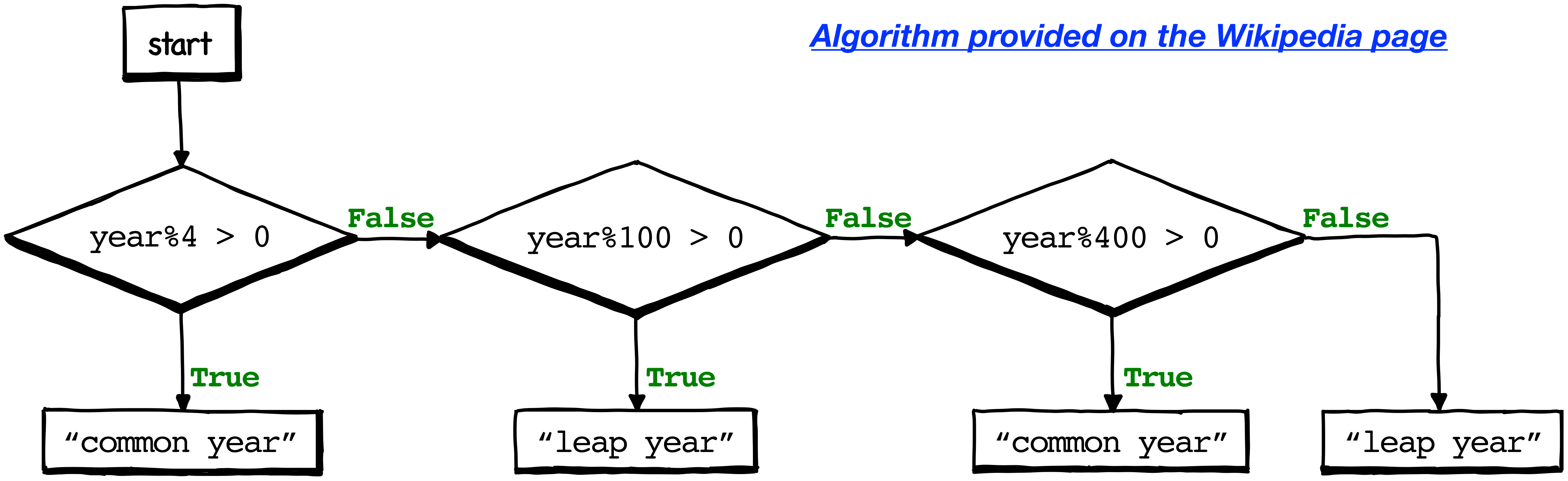
Introduction to Control Flows

- Problem-solving with algorithms

Question 3: According to the Gregorian calendar, the leap years are identified as:
1. The year is evenly divisible by 4 but not evenly divisible by 100;
2. Or the year is evenly divisible by 400.

Otherwise the year is a common year. Draw the flowchart that determines if a given year is a leap year or a common year.

Algorithm provided on the Wikipedia page



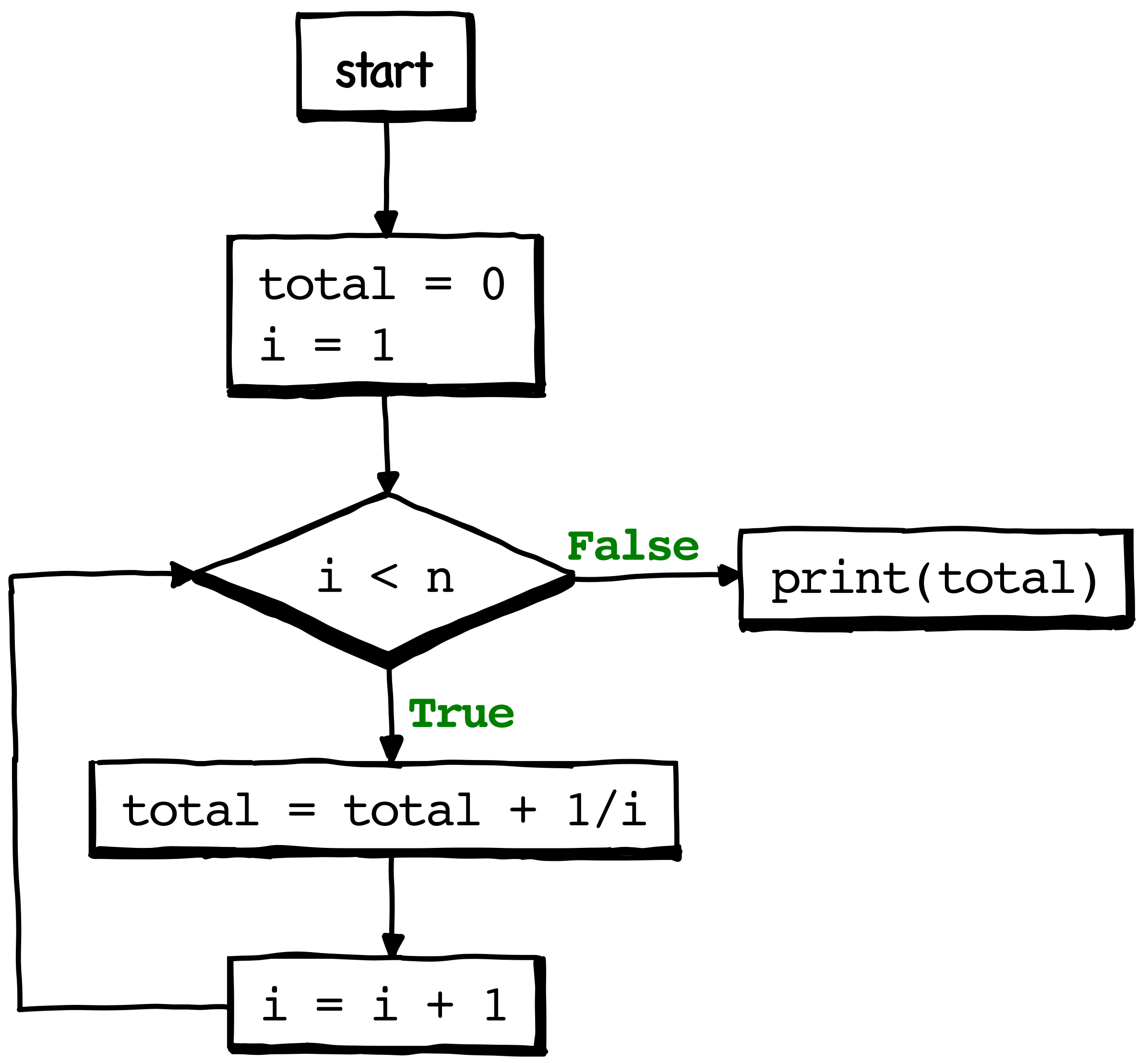
Introduction to Control Flows

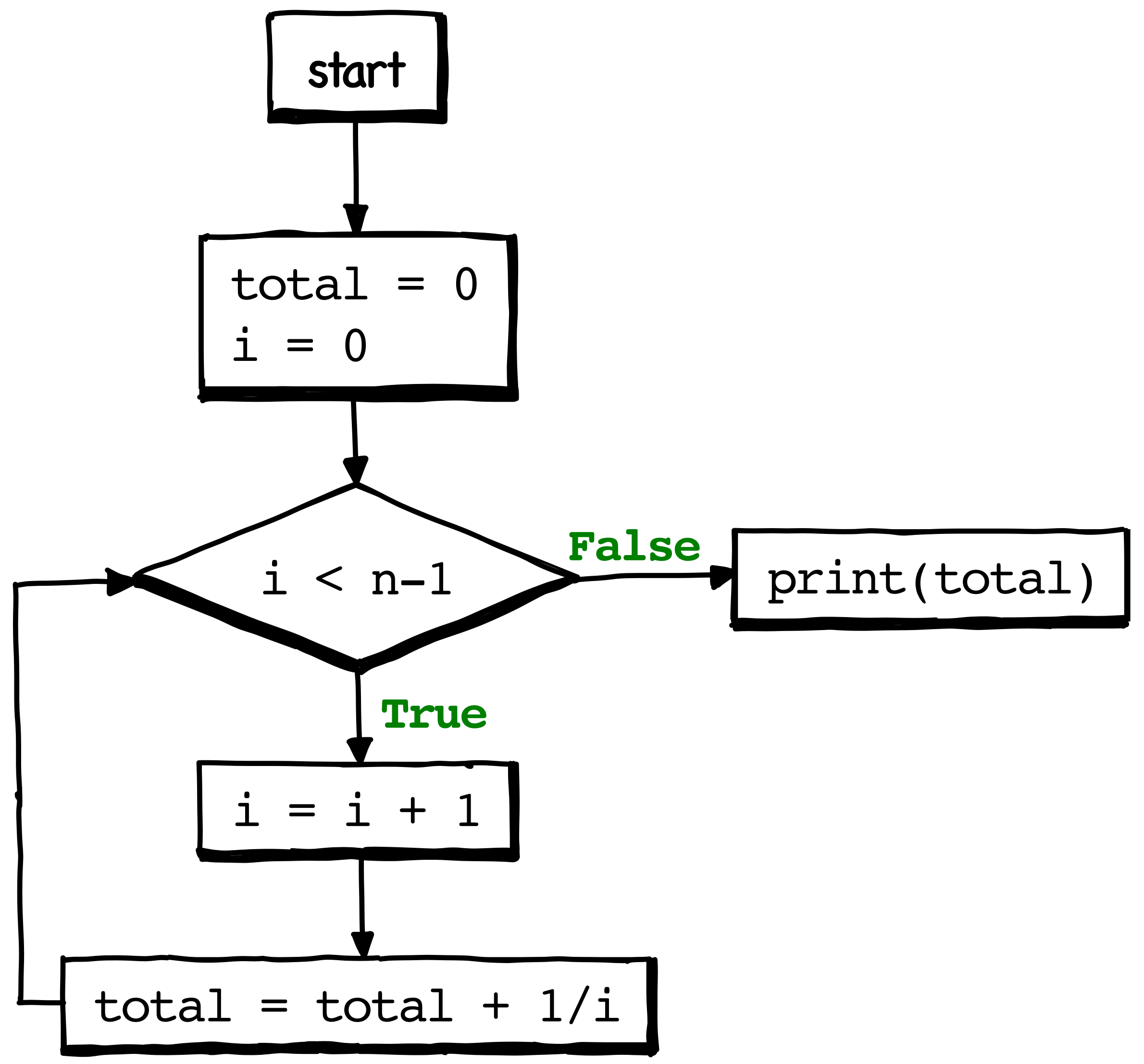
- Problem-solving with algorithms

Example 5: PUBG is an online game where n players are killing each others and the final survivor is the winner. If each player is equally good, we can prove that the expected number of kills made by the winner is $1 + 1/2 + 1/3 + \ldots + 1/(n-1)$. Draw the flowchart for calculating this expected number of kills.

Introduction to Control Flows

- Problem-solving with algorithms

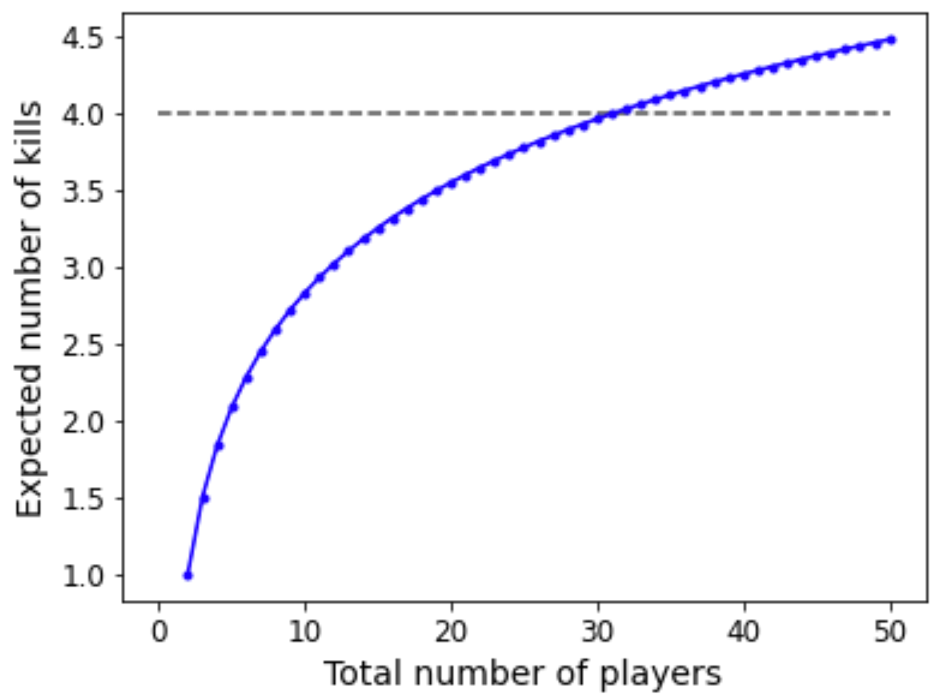




Introduction to Control Flows

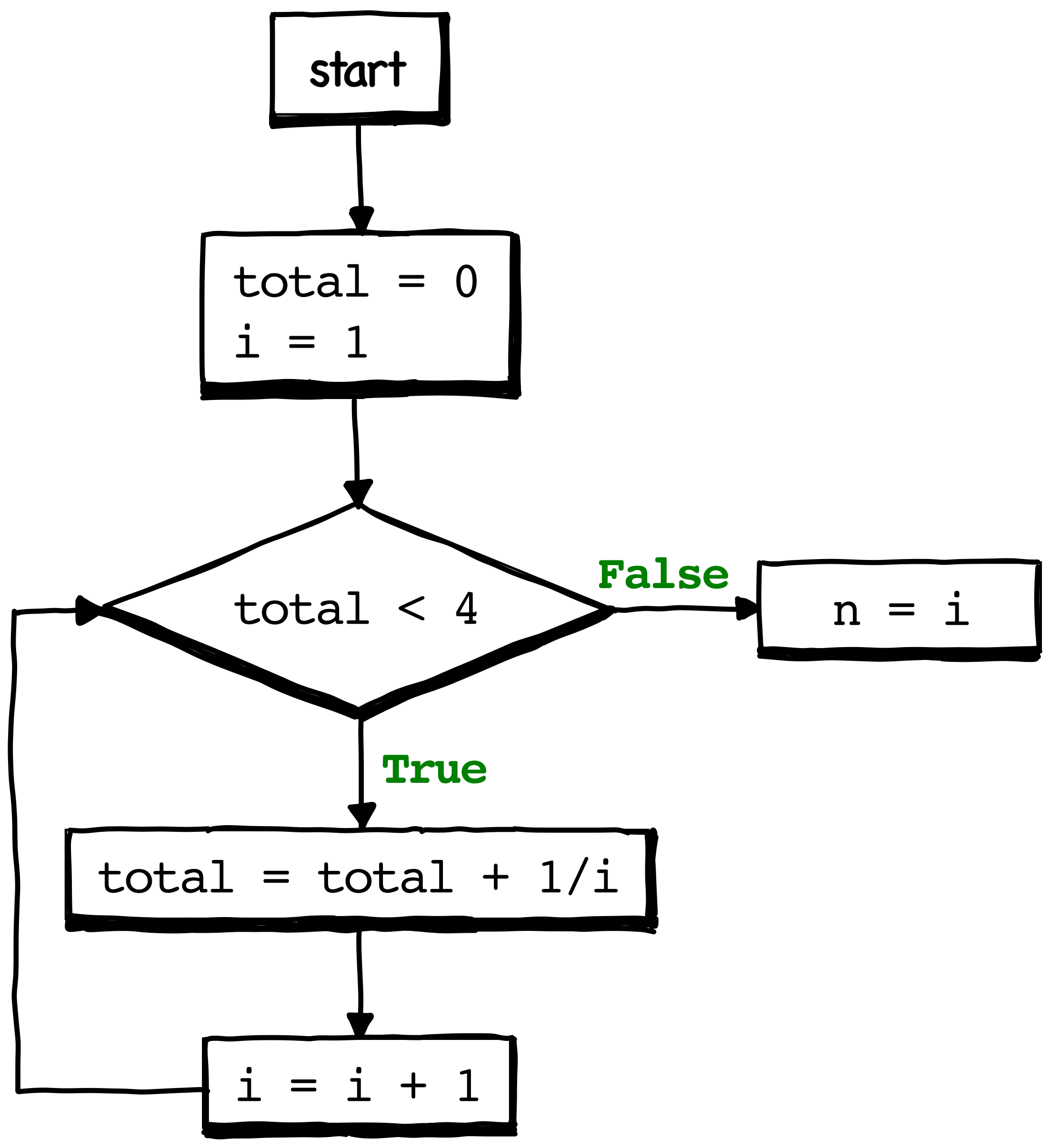
- Problem-solving with algorithms

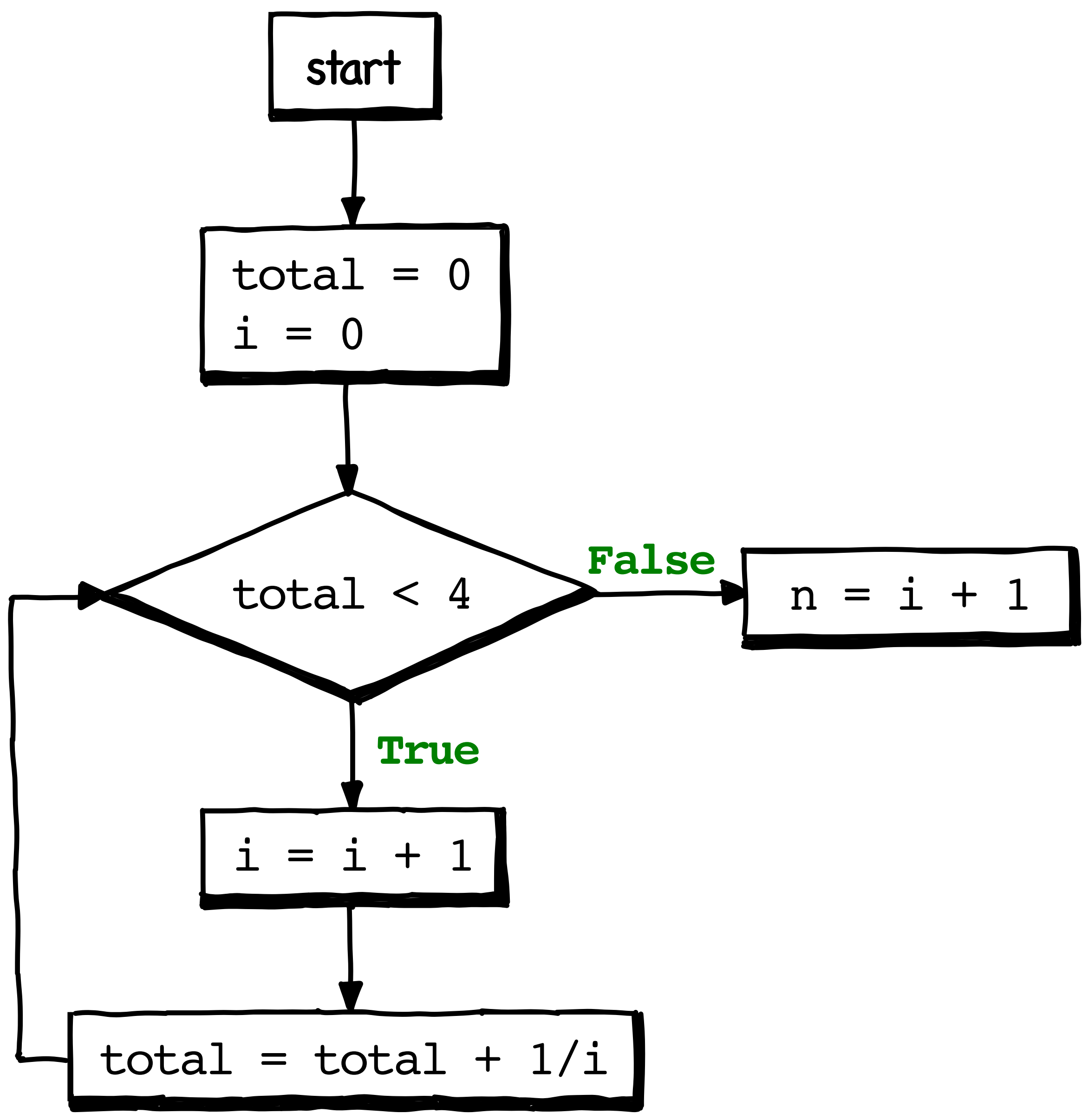
Question 4: As the developer of the PUBG game, you want to make sure that the expected number of kills made by the winner of the game is at least four, assuming all players are equally good. Let n be the total number of players in one game, draw the flowchart for calculating the minimum value of n.



Introduction to Control Flows

- Problem-solving with algorithms


start
total = 0
i = 1
total < 4
False
n = i
True
total = total + 1/i
i = i + 1


start
total = 0
i = 0
total < 4
False
n = i + 1
True
i = i + 1
total = total + 1/i