The Rules of Programming Languages

Simon Allardice
STAFF AUTHOR, PLURALSIGHT
@allardice   www.pluralsight.com

Examining Syntax

Case sensitivity

And why you notice capitalization differences

Statements

How to write (and end) each instruction

Whitespace

What it means and why it's useful

Comments

Because source code isn't always obvious

Keywords

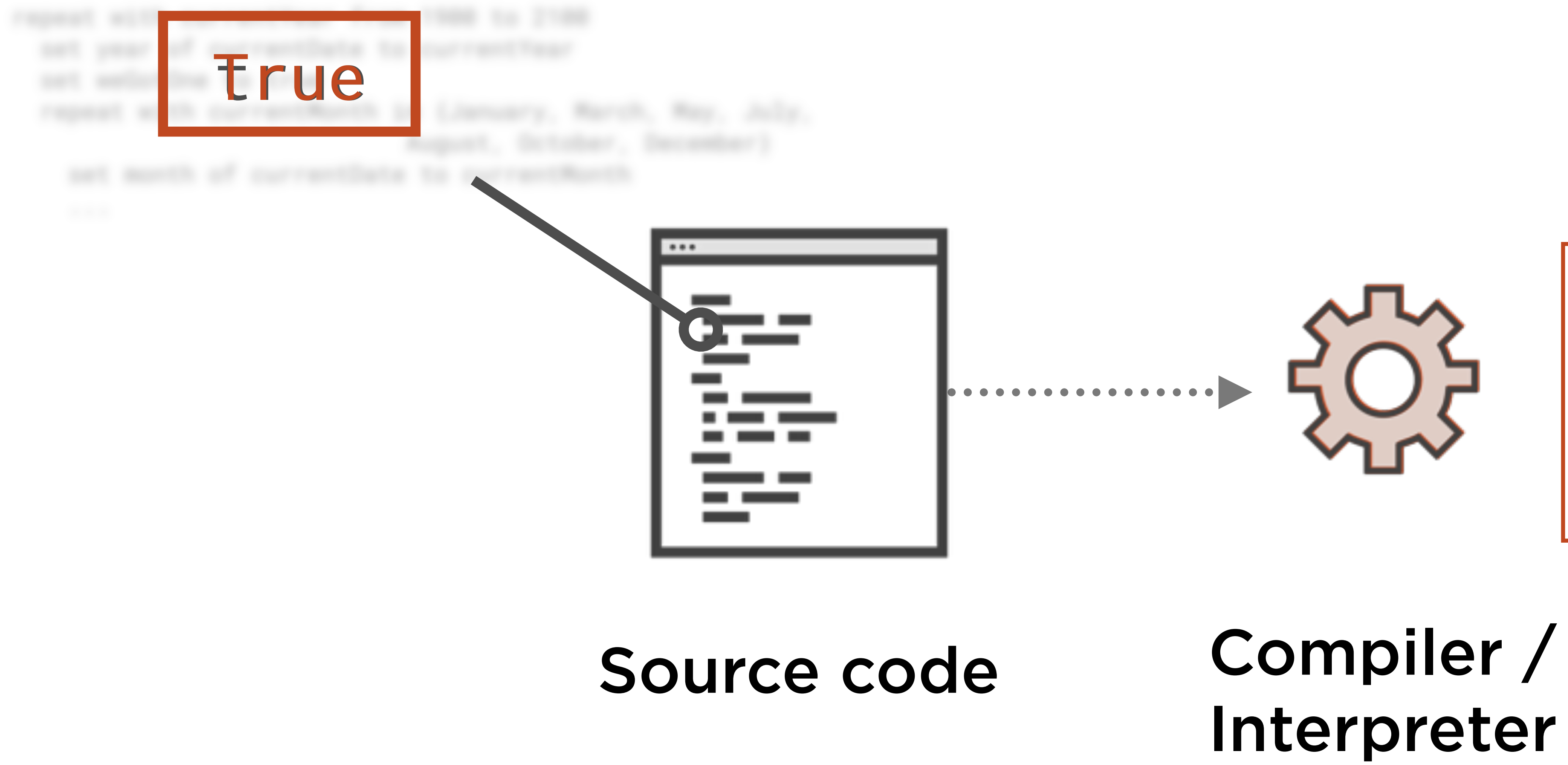
Which words belong to each language

```
repeat with currentYear from 1900 to 2100
  set year of currentDate to currentYear
  set [illegible] to true
  repeat with currentMonth in {January, March, May, July,
                               August, October, December}
    set month of currentDate to currentMonth
    ...
```

Syntax: Case Sensitivity & Capitalization

Most programming languages are case sensitive

Capitalization Matters



```
ERROR!
I don't understand
the word "True"
on line 95.
```

```
suB mAiN()
   coNst bottlesOFbeer as StrInG = " bottles of beer"
   conSt OnTheWall as string = " on the wall"
   coNST takeONEdown AS string = "take one down,"
   const passITaround as STRing = " pass it around"
```

Some languages are case-*in*sensitive

Pascal, BASIC, Ada, Fortran, SQL

Recognizing the Differences

```
import random
inclusive_range = (1, 100)

print("guess a number between %i and %i (inclusive).\n"
      % inclusive_range)

target = random.randint(*inclusive_range)
answer, i = $\boxed{None,}$ 0

while answer != target:
    i += 1
    txt = input("you guessed(%i): " % i)
    try:
        answer = int(txt)
    except $\boxed{ValueError:}$
        print("there's a problem with your input")
        continue
```

Recognizing the Differences

```
int main(){

  int number, guess;

  srand(time(NULL));
  number = 1 + (rand() % 1000);

  printf( "enter a number between 1 and 1000:");

  while( scanf( "%d", &guess ) == 1 ){
    if( number == guess ){
      printf( "you guessed well!\n" );
      break;
    }

  ...
```

next wednesday, i will drive you to the airport to fly from phoenix to london.

Next wednesday, i will drive you to the airport to fly from phoenix to london.

Next Wednesday, i will drive you to the airport to fly from phoenix to london.

Next Wednesday, i will drive you to the airport to fly from Phoenix to London.

Next Wednesday, $\overset{\textit{ich}}{\text{I}}$ will drive $\overset{\textit{Sie}}{\boxed{\text{you}}}$ to the airport to fly from Phoenix to London.

Python

```
...

if currentScore > highScore:
    return True
  else:
    return False
```

Swift

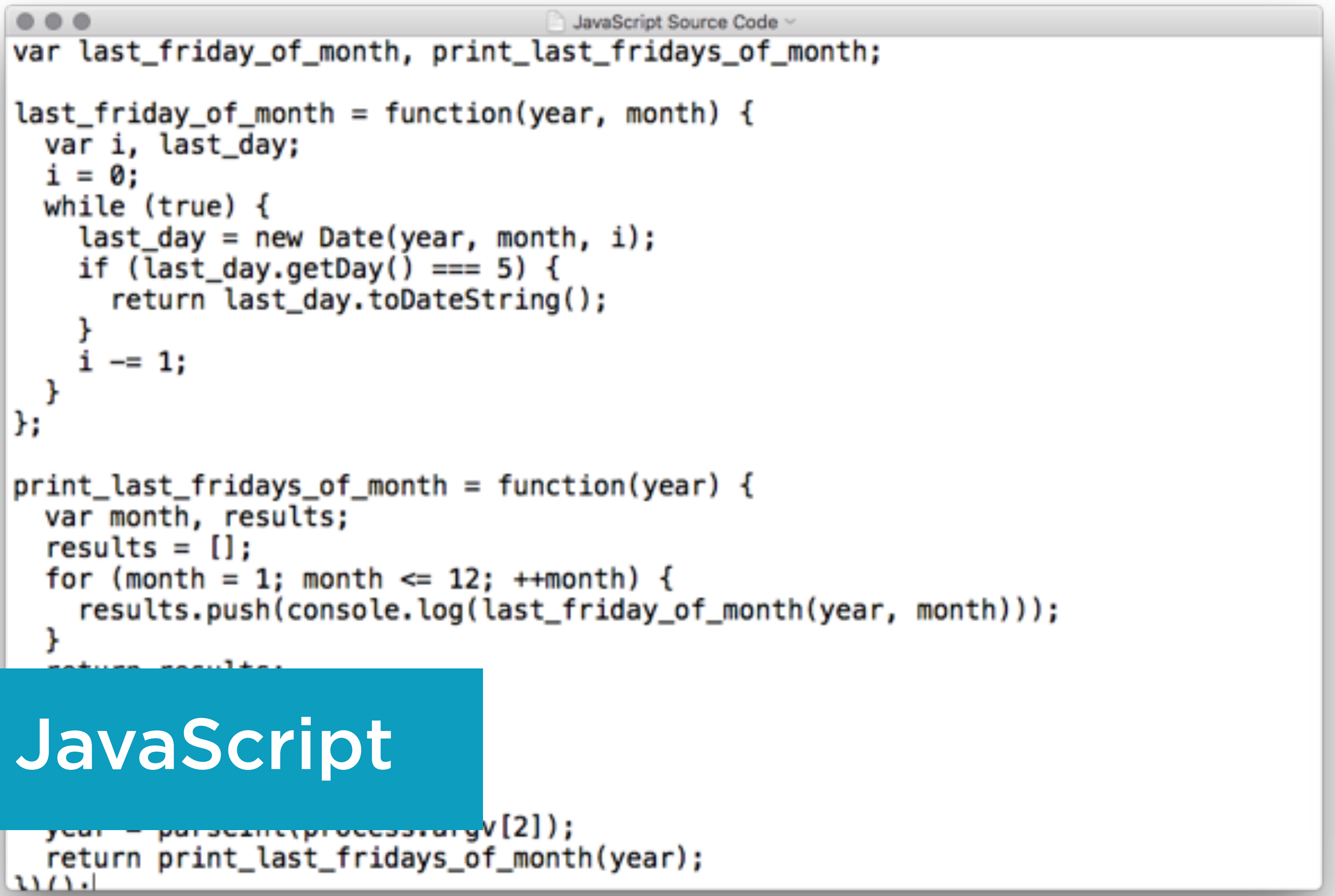
```
...

if currentScore > highScore {
    return true
} else {
    return false
}
```

Python

```
...

if currentScore > highScore:
    return True
  else:
    return False
```

Swift

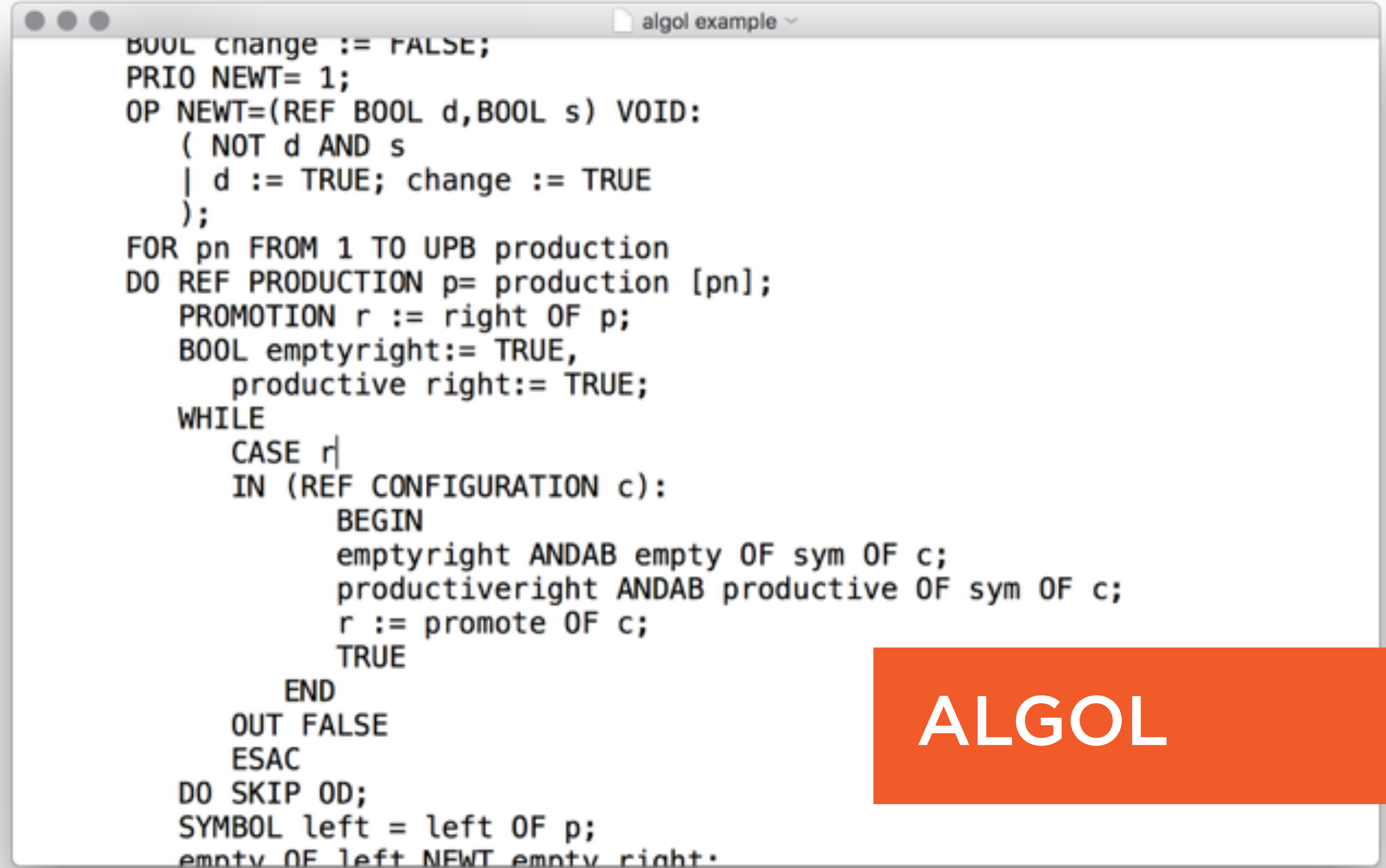
```
...

if currentScore > highScore {
    return true
} else {
    return false
}
```



```
var last_friday_of_month, print_last_fridays_of_month;

last_friday_of_month = function(year, month) {
  var i, last_day;
  i = 0;
  while (true) {
    last_day = new Date(year, month, i);
    if (last_day.getDay() === 5) {
      return last_day.toDateString();
    }
    i -= 1;
  }
};

print_last_fridays_of_month = function(year) {
  var month, results;
  results = [];
  for (month = 1; month <= 12; ++month) {
    results.push(console.log(last_friday_of_month(year, month)));
  }
  return results;
```

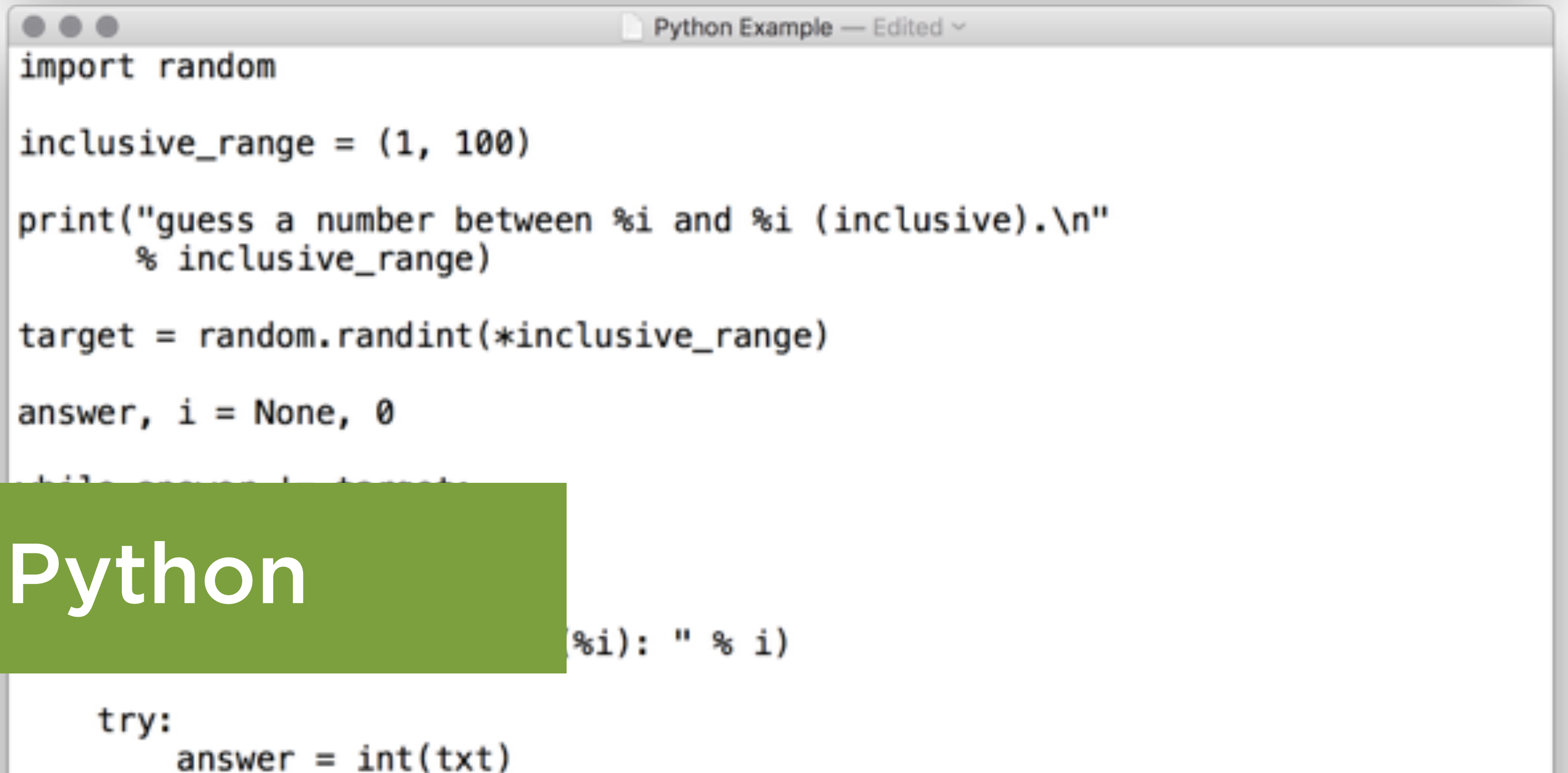
```
  year = parseInt(process.argv[2]);
  return print_last_fridays_of_month(year);
})();
```

JavaScript



```
        BOOL change := FALSE;
        PRIO NEWT= 1;
        OP NEWT=(REF BOOL d,BOOL s) VOID:
           ( NOT d AND s
           | d := TRUE; change := TRUE
           );
        FOR pn FROM 1 TO UPB production
        DO REF PRODUCTION p= production [pn];
           PROMOTION r := right OF p;
           BOOL emptyright:= TRUE,
              productive right:= TRUE;
           WHILE
              CASE r
              IN (REF CONFIGURATION c):
                    BEGIN
                    emptyright ANDAB empty OF sym OF c;
                    productiveright ANDAB productive OF sym OF c;
                    r := promote OF c;
                    TRUE
                 END
              OUT FALSE
              ESAC
           DO SKIP OD;
           SYMBOL left = left OF p;
           empty OF left NEWT empty right;
```

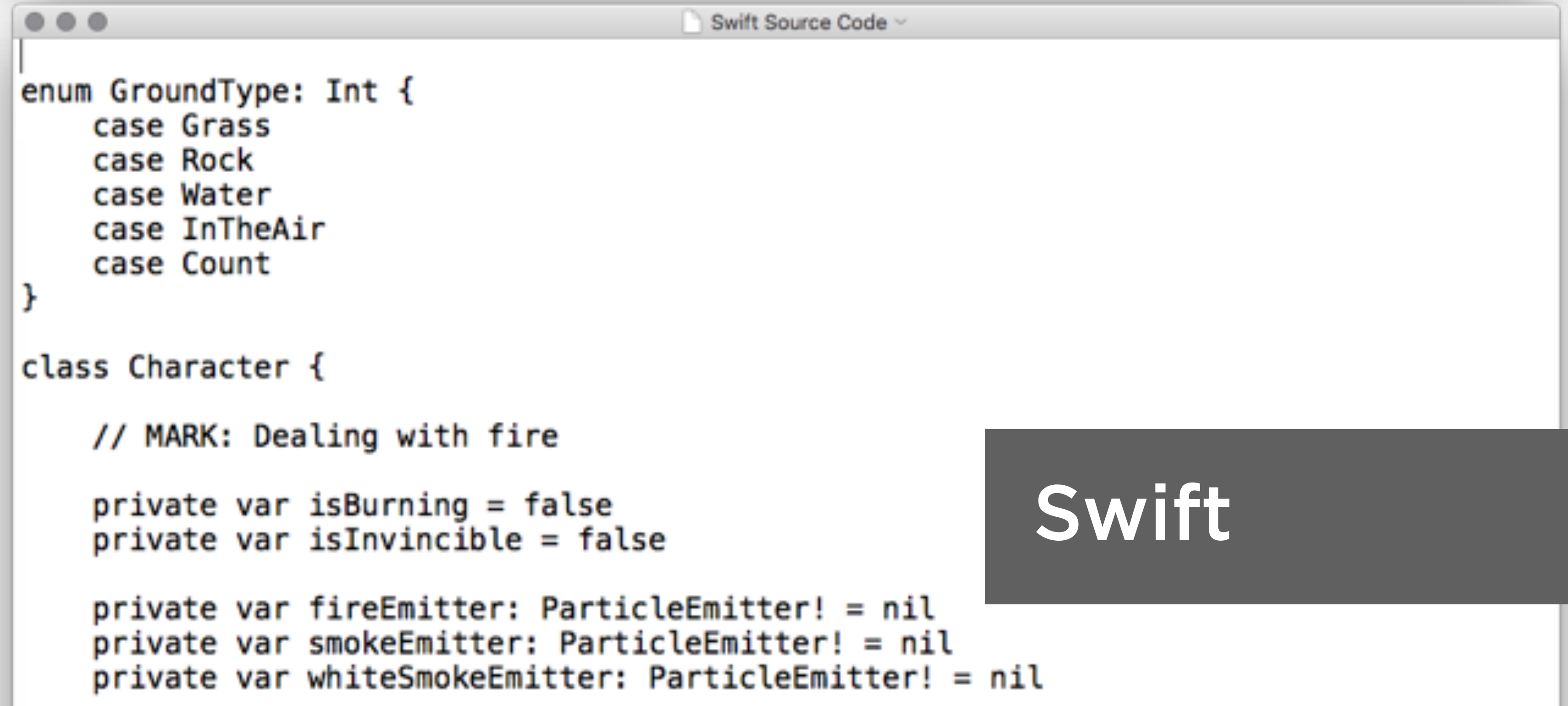
ALGOL



```
import random

inclusive_range = (1, 100)

print("guess a number between %i and %i (inclusive).\n"
      % inclusive_range)

target = random.randint(*inclusive_range)

answer, i = None, 0

while answer != target:
```

```
%i): " % i)

    try:
        answer = int(txt)
```

Python



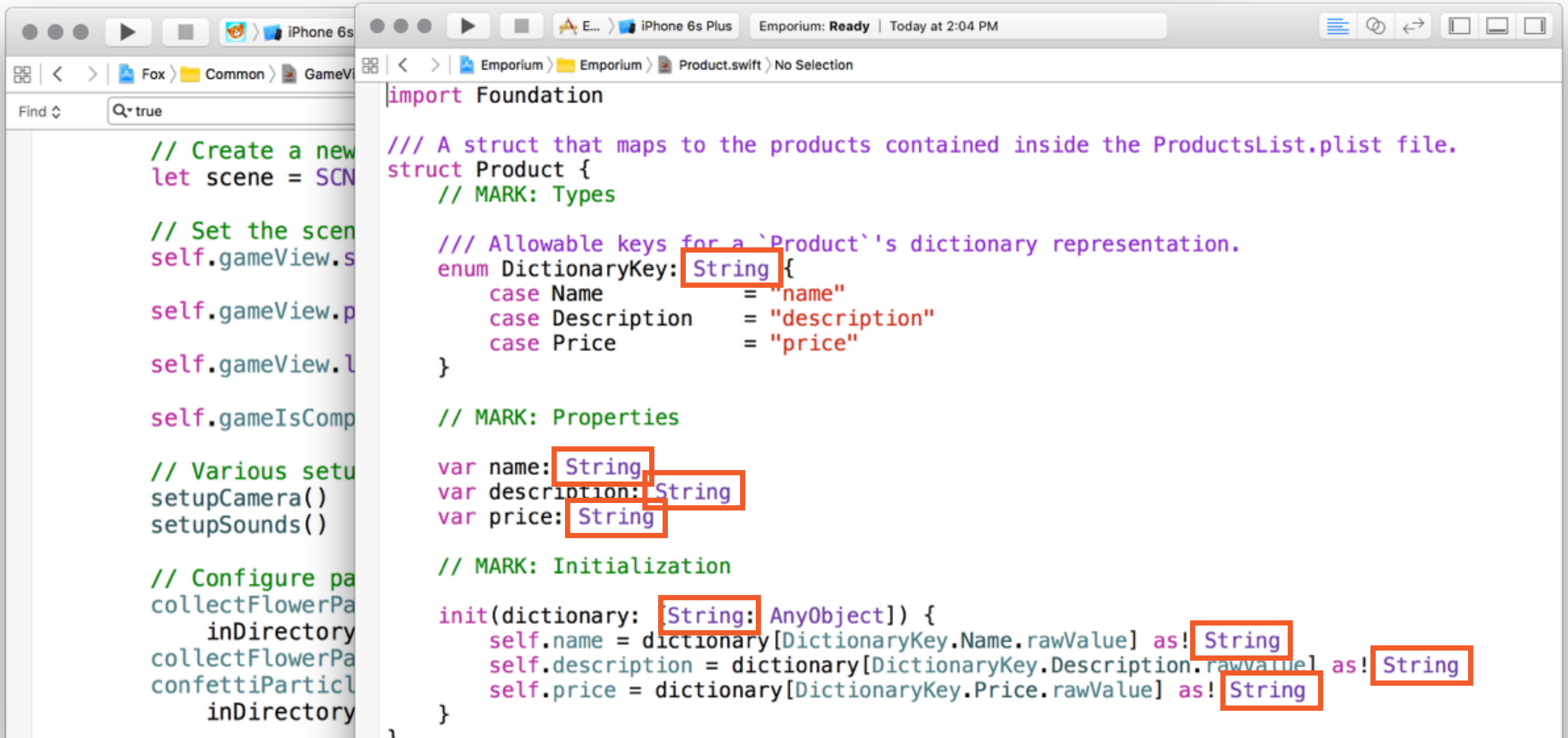
```
enum GroundType: Int {
    case Grass
    case Rock
    case Water
    case InTheAir
    case Count
}

class Character {

    // MARK: Dealing with fire

    private var isBurning = false
    private var isInvincible = false

    private var fireEmitter: ParticleEmitter! = nil
    private var smokeEmitter: ParticleEmitter! = nil
    private var whiteSmokeEmitter: ParticleEmitter! = nil
```

Swift

Next Wednesday, I will drive you to the airport to fly from Phoenix to London.

Next Wednesday, I will drive you to the airport to fly from Phoenix to London.

Emulate What You See in a Language



```
// Create a new
let scene = SCN

// Set the scen
self.gameView.s

self.gameView.p

self.gameView.l

self.gameIsComp

// Various setu
setupCamera()
setupSounds()

// Configure pa
collectFlowerPa
    inDirectory
collectFlowerPa
confettiParticl
    inDirectory
```

```
import Foundation

/// A struct that maps to the products contained inside the ProductsList.plist file.
struct Product {
    // MARK: Types

    /// Allowable keys for a `Product`'s dictionary representation.
    enum DictionaryKey: String {
        case Name          = "name"
        case Description   = "description"
        case Price         = "price"
    }

    // MARK: Properties

    var name: String
    var description: String
    var price: String

    // MARK: Initialization

    init(dictionary: [String: AnyObject]) {
        self.name = dictionary[DictionaryKey.Name.rawValue] as! String
        self.description = dictionary[DictionaryKey.Description.rawValue] as! String
        self.price = dictionary[DictionaryKey.Price.rawValue] as! String
    }
}
```

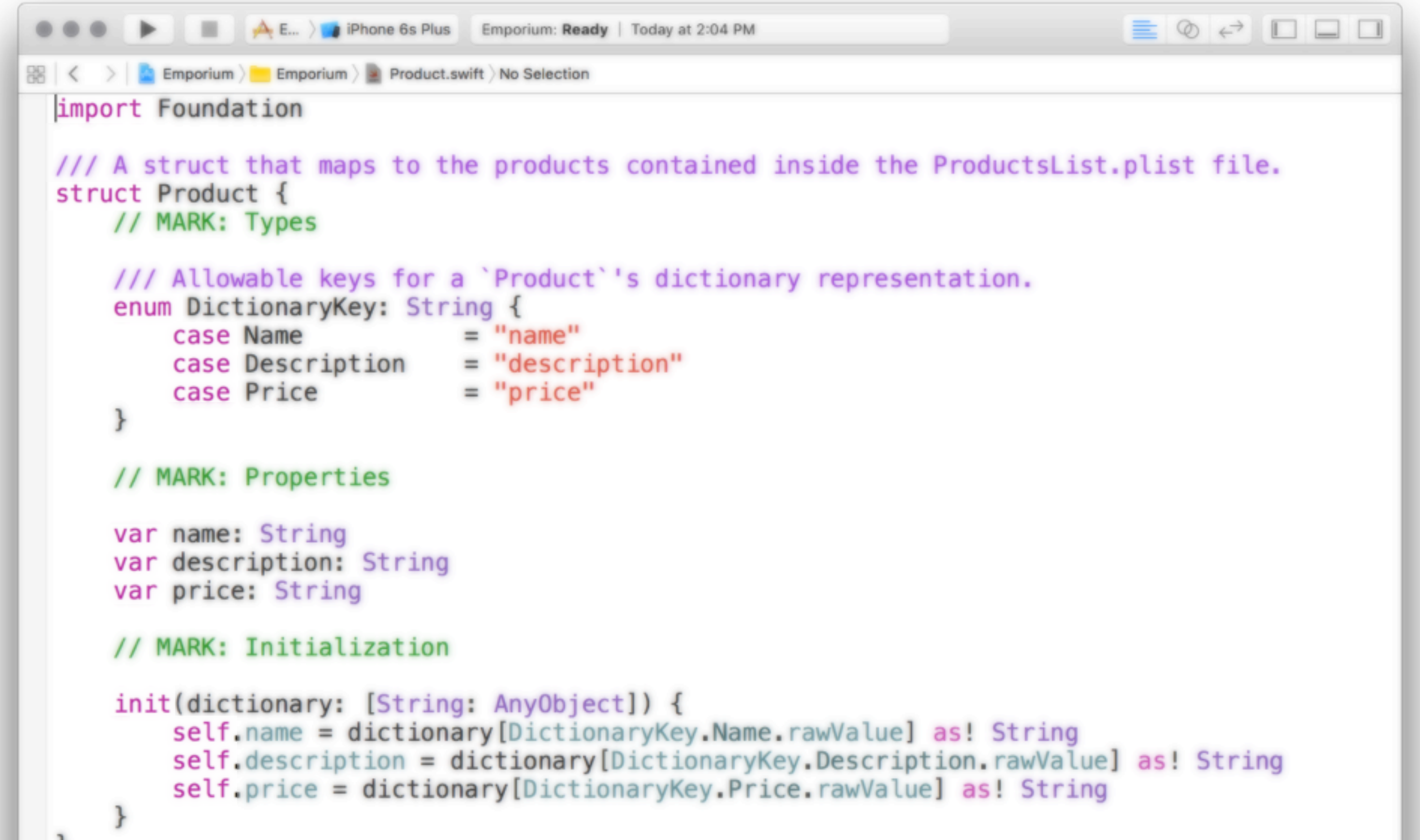
“Hello, $\boxed{\text{, i }}$am a programmer.”

“Hello, I am a programmer.”

I plan to travel next week.

Next Wednesday, I will drive you to the airport to fly from Phoenix to London.

Example: Swift



```swift
import Foundation

/// A struct that maps to the products contained inside the ProductsList.plist file.
struct Product {
    // MARK: Types

    /// Allowable keys for a `Product`'s dictionary representation.
    enum DictionaryKey: String {
        case Name           = "name"
        case Description    = "description"
        case Price          = "price"
    }

    // MARK: Properties

    var name: String
    var description: String
    var price: String

    // MARK: Initialization

    init(dictionary: [String: AnyObject]) {
        self.name = dictionary[DictionaryKey.Name.rawValue] as! String
        self.description = dictionary[DictionaryKey.Description.rawValue] as! String
        self.price = dictionary[DictionaryKey.Price.rawValue] as! String
    }
```

Display…

Add…

Play…

Syntax: Writing Statements

Most programming languages are case sensitive

"Change the color of **the background** to **light blue**"

"Add 99 to **the current score**"

"Print **this document** to **the default printer**"

"Move **the spaceship graphic** by **5 pixels** to the **right**"

Each Statement Must Be Complete

What is being done, and what is it being done to?

"Change the color of **the background** to **light blue**"

"Add 99 to **the current score**"

"Print **this document** to **the default printer**"

"Move **the spaceship graphic** by **5 pixels** to the **right**"

Example Statements

0

score

Statement Examples

COBOL syntax:

```
ADD 99 to score.
```

AppleScript syntax:

```
set score to score + 99
```

Swift, Ruby, Python (and others):

```
score = score + 99
```

C, C++, PHP, Java (and others):

```
score = score + 99;
```

```
score IS NOW EQUAL TO score + 99
```

Ending Statements

```
statement one;
statement two;

statement three might be long and be split across
   multiple lines to make it
   easier to read;

statement four;
statement five;
```

```
static unsigned int offset;
static unsigned int ino = 721;
static time_t default_mtime;

struct file_handler {
 const char *type;
 int (*handler)(const char *line);
};

static void push_string(const char *name)
{
 unsigned int name_len = strlen(name) + 1;

 fputs(name, stdout);
 putchar(0);
 offset += name_len;
}

static void push_pad (void)
{
 while (offset & 3) {
  putchar(0);
  offset++;
```

Ending Statements

COBOL syntax:

```
ADD 99 to score.
```

AppleScript syntax:

```
set score to score + 99
```

Swift, Ruby, Python (and others):

```
score = score + 99
```

C, C++, PHP, Java (and others):

```
score = score + 99;
```

```
enum GroundType: Int {
    case Grass
    case Rock
    case Water
    case InTheAir
    case Count
}

class Character {

    // MARK: Dealing with fire

    private var isBurning = false
    private var isInvincible = false

    private var fireEmitter: ParticleEmitter! = nil
    private var smokeEmitter: ParticleEmitter! = nil
    private var whiteSmokeEmitter: ParticleEmitter! = nil

    func haltFire() {
        if isBurning {
            isBurning = false

            // stop fire and smoke
```

Ending Statements

COBOL syntax:

```
ADD 99 to score.
```

AppleScript syntax:

```
set score to score + 99
```

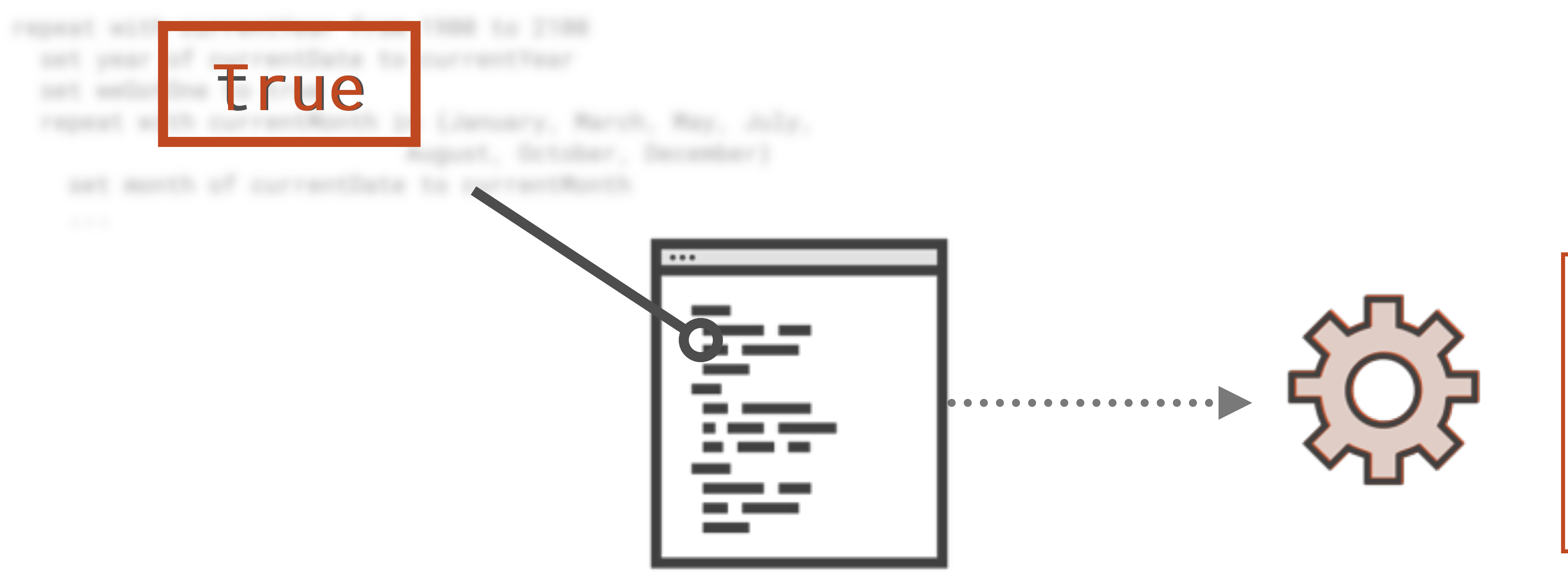
Swift, Ruby, Python (and others):

```
score = score + 99
```

C, C++, PHP, Java (and others):

```
score = score + 99;
```

Capitalization Matters



```
ERROR!
I don't understand
the word "True"
on line 95.
```

Source code

Compiler / Interpreter

```
suB mAiN()
   coNst bottlesOFbeer as StrInG = " bottles of beer"
   conSt OnTheWall as string = " on the wall"
   coNST takeONEdown AS string = "take one down,"
   const passITaround as STRing = " pass it around"
```

Some languages are case-*in*sensitive

Pascal, BASIC, Ada, Fortran, SQL

Recognizing the Differences

```
int main(){

  int number, guess;

  srand(time(NULL));
  number = 1 + (rand() % 1000);

  printf( "enter a number between 1 and 1000:");

  while( scanf( "%d", &guess ) == 1 ){
    if( number == guess ){
      printf( "you guessed well!\n" );
      break;
    }

  ...
```

```
Display "Hello" on the screen
Add 99 to the score
Play "fanfare" sound effect
Move spaceship 5 pixels to the right
```

Syntax: Using Pseudocode

Writing human-readable statements before we write computer-readable statements

On game completion:

Display "congratulations!" message

Play fanfare

Reset game

Save score

Reset game

```
If the game is finished:
    Display the "congratulations!" message in the center of the screen
    Play the "fanfare.mp3" sound effect
    Save the score
    Reset all game pieces
```

Pseudocode

C programmer

```
function gameComplete {
    print("congratulations");
    playSound("fanfare.mp3");
    (etc.)
}
```

VB programmer

```
Sub OnGameComplete
    Print "congratulations"
    Play fanfare
    (etc.)
End Sub
```

On game completion:

Display "congratulations!" message

Play fanfare ← ?

Save score

Reset game

to do: research this! mp3? wav file?
what's the best way?

Syntax: Using Whitespace

COBOL

```
PROCEDURE DIVISION USING L-Input-Date-DT
                   RETURNING L-Output-Day-NUM.
000-Main SECTION.

    EVALUATE RETURN-CODE
    WHEN 7
        IF TEST-DAY-YYYYDDD(L-Input-Date-DT) > 0
            MOVE 0 TO L-Output-Day-NUM
            GOBACK
        END-IF
        MOVE DATE-OF-INTEGER(INTEGER-OF-DAY(L-Input-Date-DT))
            TO WS-Input-Date-DT
    WHEN 8
        IF TEST-DATE-YYYYMMDD(L-Input-Date-DT) > 0
            MOVE 0 TO L-Output-Day-NUM
            GOBACK
        END-IF
```

COBOL

```
PROCEDURE DIVISION USING L-Input-Date-DT
*******************RETURNING L-Output-Day-NUM.
000-Main SECTION.
***********************************************************
****EVALUATE RETURN-CODE
****WHEN 7
********IF TEST-DAY-YYYYDDD(L-Input-Date-DT) > 0
************MOVE 0 TO L-Output-Day-NUM
************GOBACK
********END-IF
********MOVE DATE-OF-INTEGER(INTEGER-OF-DAY(L-Input-Date-DT))
************TO WS-Input-Date-DT
****WHEN 8
********IF TEST-DATE-YYYYMMDD(L-Input-Date-DT) > 0
************MOVE 0 TO L-Output-Day-NUM
************GOBACK
********END-IF
```

Python

```
def sing(b, end):
    print(b or 'no more','bottle'+('s' if b-1 else ''), end)

for i in range(99, 0, -1):
    sing(i, 'of beer on the wall,')
    sing(i, 'of beer,')
    print('take one down, pass it around,')
    sing(i-1, 'of beer on the wall.\n')
```

Python

```
def sing(b, end):
****print(b or 'no more','bottle'+('s' if b-1 else ''), end)
***********************************************************
for i in range(99, 0, -1):
****sing(i, 'of beer on the wall,')
****sing(i, 'of beer,')
****print('take one down, pass it around,')
****sing(i-1, 'of beer on the wall.\n')
***********************************************************
***********************************************************
```

PHP

```php
function last_friday_of_month($year, $month) {

    $day = 0;

    while(True) {

       $last_day = mktime(0, 0, 0, $month+1, $day, $year);

       if (date("w", $last_day) == 5) {
          return date("Y-m-d", $last_day);
       }

       $day -= 1;
    }

}
```

PHP

```
function last_friday_of_month($year, $month) {
***********************************************************
****$day = 0;
***********************************************************
****while(True) {
***********************************************************
*******$last_day = mktime(0, 0, 0, $month+1, $day, $year);
***********************************************************
*******if (date("w", $last_day) == 5) {
***********return date("Y-m-d", $last_day);
*******}
***********************************************************
*******$day -= 1;
****}
***********************************************************
}
***********************************************************
```

PHP

```
function last_friday_of_month($year, $month) {
***********************************************************
****$day = 0;
***********************************************************
****while(True) {
***********************************************************
*******$last_day = mktime(0, 0, 0, $month+1, $day, $year);
***********************************************************
*******if*(date(**w", $last_day) == 5) {
************return*date("Y-m-d", $last_day);
*******}***********
***********************************************************
*******$day -= 1;
****}
***********************************************************
}
***********************************************************
```

PHP

```
function last_friday_of_month($year, $month) {
************************************************************
****$day = 0;
************************************************************
****while(True) {
************************************************************
*******$last_day = mktime(0, 0, 0, $month+1, $day, $year);
************************************************************
*************************if (date("w", $last_day) == 5) {
*****************************return date("Y-m-d", $last_day);
*************************}
************************************************************
*******$day += 1;*******************************************
****}*******************************************************
************************************************************
}
************************************************************
```

PHP

```
function last_friday_of_month($year, $month) {

    $day = 0;

    while(True) {

       $last_day = mktime(0, 0, 0, $month+1, $day, $year);

                         if (date("w", $last_day) == 5) {
                             return date("Y-m-d", $last_day);
                         }




       $day -= 1;
    }
```

Change the col
or of the background to light blue

Add99tothecurrentscore

Syntax: Whitespace

The language must be able to recognize each element of the statement

Change the   color of the   background to   light blue

Add 99 to the current score

Syntax: Whitespace

With most languages, any additional spaces, blank lines or tabs are ignored

```
enum GroundType: Int {
    case Grass
    case Rock
    case Water
    case InTheAir
    case Count
}

class Character {

    // MARK: Dealing with fire

    private var isBurning = false
    private var isInvincible = false

    private var fireEmitter: ParticleEmitter! = nil
    private var smokeEmitter: ParticleEmitter! = nil
    private var whiteSmokeEmitter: ParticleEmitter! = nil

    func haltFire() {
        if isBurning {
            isBurning = false

            // stop fire and smoke
```

```
enum GroundType: Int {
    case Grass
    case Rock
    case Water
    case InTheAir
    case Count
}

class Character {

    // MARK: Dealing with fire

    private var isBurning = false
    private var isInvincible = false

    private var fireEmitter: ParticleEmitter! = nil
    private var smokeEmitter: ParticleEmitter! = nil
    private var whiteSmokeEmitter: ParticleEmitter! = nil

    func haltFire() {
        if isBurning {
            isBurning = false

            // stop fire and smoke
```

```
enum GroundType: Int {
    case Grass
    case Rock
    case Water
    case InTheAir
    case Count
}

class Character {

    // MARK: Dealing with fire

    private var isBurning = false
    private var isInvincible = false

    private var fireEmitter: ParticleEmitter! = nil
    private var smokeEmitter: ParticleEmitter! = nil
    private var whiteSmokeEmitter: ParticleEmitter! = nil

    func haltFire() {
        if isBurning {
            isBurning = false

            // stop fire and smoke
```

In Python, Indentation is "Syntactically Significant"

```
if playerScore > highScore:
    print("Congratulations, you have the high score!")
    highScore = playerScore
print("Thank you for playing")
```

Swift

```
if playerScore > highScore { opening curly brace - block begins here
    print("Congratulations, you have the high score!")
    highScore = playerScore
    print("Thank you for playing")
} closing curly brace - block ends here
```

Syntax: Adding Comments

```
// Play the live audio stream
func playAudio() {
    if isBurning {

        // check we have internet connection
        fireEmitter()
        if something > 500 {
            self.smokeEmitter.birthRate = 0
            print("something")
        }

        // is anything else playing audio right now??
        something.particleSystem.birthRate = 0
        someFunction()

        // reset volume
        whiteSmokeEmitter.particleSystem.birthRate = 1.0

        // start audio playing
        walkSpeed = 1.0
    }
}
```

`// this is a comment`	◀ C-style languages
`#  this is a comment`	◀ Ruby, Python
`'  this is a comment`	◀ Visual Basic
`-- this is a comment`	◀ Ada
`REM this is a comment`	◀ BASIC

```
// Play the live audio stream
func playAudio() {
    if isBurning {

        // check we have internet connection
        fireEmitter()
        if something > 500 {
            self.smokeEmitter.birthRate = 0
            print("something")
        }

        // is anything else playing audio right now??
        something.particleSystem.birthRate = 0
        someFunction()

        // reset volume
        whiteSmokeEmitter.particleSystem.birthRate = 1.0

        // start audio playing
        walkSpeed = 1.0
    }
}
```

Not All Code Needs Commented

```
// display a message that says "Thanks for playing!"
print("Thanks for playing!")
```

Multiline / Block Comments

```
/*
   everything from this point on is a comment.
   as many lines as you need.

   blank lines if you want them.

   it's all treated as a comment.
   until we get to
   the closing asterisk-and-forward-slash.
*/
```

Comments as Reminders

```
// TODO: display custom graphic here
fireEmitter()
if something > 500 {
    self.smokeEmitter.birthRate = 0
    print("something")
}

// FIXME: this does not display correctly on right-to-left languages
something.particleSystem.birthRate = 0
someFunction()

// HACK: this is a really slow workaround! find a better way!
whiteSmokeEmitter.particleSystem.birthRate = 1.0
```

Commenting Out Code

```
print("Thanks for playing!")
//playSound("fanfare.mp3")
player.saveScore()
game.resetPieces()
```

Syntax: Keywords

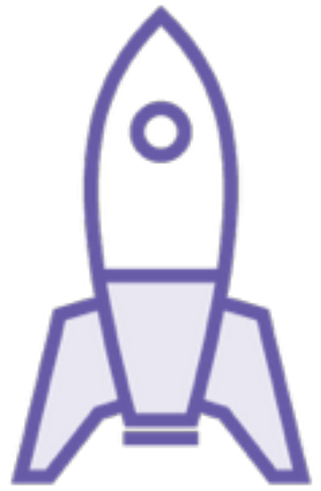
What We'll Always Get

	Python	COBOL	AppleScript	Java
Change a value	`score = score + 10`	`ADD 10 TO score`	`set score to score + 10`	`score = score + 10;`

Ask a question	`if score > 100:`
	`    ...`
		`IF score GT 100 THEN`
		`    ...`
			`if score > 100 then`
			`    ...`
				`if (score > 100) {`
				`    ...`

Display a message	`print("Hello")`	`DISPLAY "Hello"`	`display dialog "Hello"`	`System.out.println("Hello");`
Control the program	`break`, `continue`, `return`, *etc.*	`PERFORM`, `NEXT`, `STOP RUN`, *etc.*	`tell`, `continue`, `exit`, *etc.*	`break`, `continue`, `return`, *etc.*

What We Might Hope For



```
spaceship.launch()
spaceship.rotate(180)
```

```
spaceship.explode()
```

What We Might Hope For

`officelocations`

Java Keywords

abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

```
func playAudio() {
    if isBurning {

        fireEmitter()
        if something > 500 {
            self.smokeEmitter.birthRate = 0
            spaceship.explode()
        }

        something.particleSystem.birthRate = 0
        someFunction()

        whiteSmokeEmitter.particleSystem.birthRate = 1.0

        walkSpeed = 1.0
    }
}
```

```
        fireEmitter()
        if something > 500 {
            marketing.generateMonthlyReport()
            print("something")
        }

        something.particleSystem.birthRate = 0
        someFunction()

        whiteSmokeEmitter.particleSystem.birthRate = 1.0

        walkSpeed = 1.0
    }
}
```

Example: Swift

keyword

```
// ...

if currentScore > highScore {
    print("Congratulations, you have the high score!")
} else {
    print("Try again - better luck next time.")
}

// ...
```