Introduction To Computer Programming

You will learn how to create simple computer programs using Alice and some general programming concepts

---

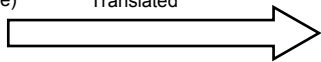# Creating Computer Programs

Machine language (binary)

Programming language
```
if (robot.wallToRight () == true)
{
  isDone = true;
  return;
}
```

Translated →

Programming Languages

•Similar to human languages there is a multitude of programming languages.

•Different languages exist to fulfill different needs.

•Examples (no where near exhaustive):
- Java: Internet-based programs
- C: often used in the development of operating systems
- C#: the language of choice when developing applications specific to Windows
- Pascal: a teaching language but often used in the development of databases (Oracle).
- Etc.
- Etc.
- Etc.

The Programming Language Used For This Class

•Alice: a programming language and environment (allows for the creation of programs) developed by Carnegie Mellon University.

•Why?
- It makes standard programming instructions easier to learn (graphical rather than text).
- It's free!

•Where?
- http://www.alice.org (main page)
- http://www.alice.org/index.php?page=downloads/download_alice2.2 (download)

•What computer type?
- Windows
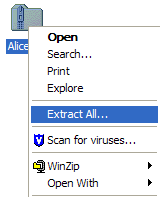- Mac

•What version?
- 2.2 (Windows, Mac), 2.0 (Linux)
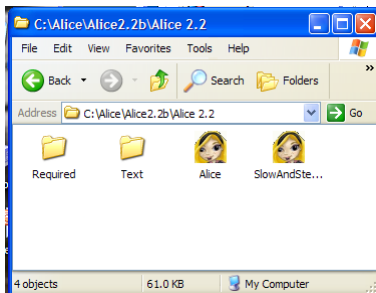
Download: What You Get

•A compressed (zip) file.
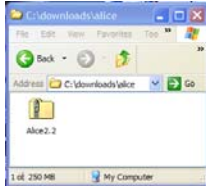

•This file needs to be uncompressed.

Starting Alice

•Alice doesn't need to be installed on your computer after it's be uncompressed.

•Just find the folder containing the executable program: "Alice" or "SlowAndSteady" (recommended)

- These programs can be found in the folder where you uncompressed the Alice files (Alice2.2b\Alice2.2)
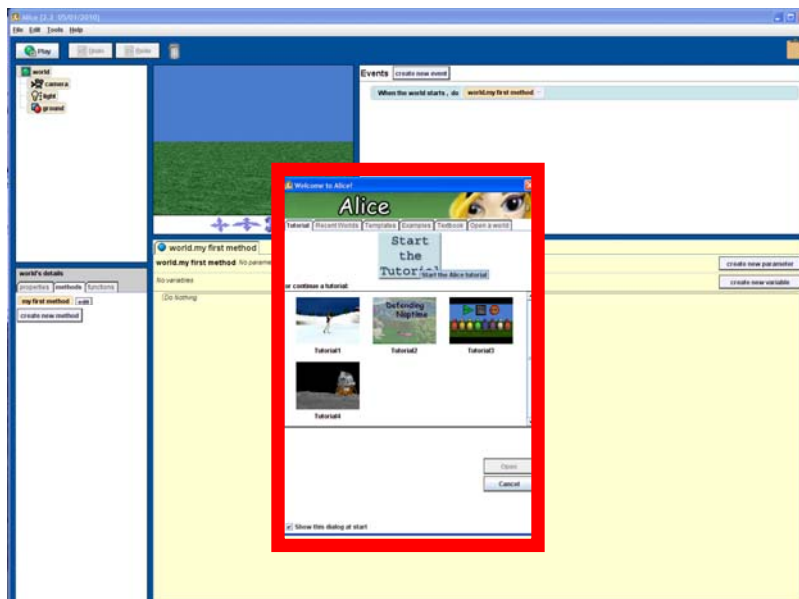
Scary Message???!!!

•If you see this message you should normally pay attention!
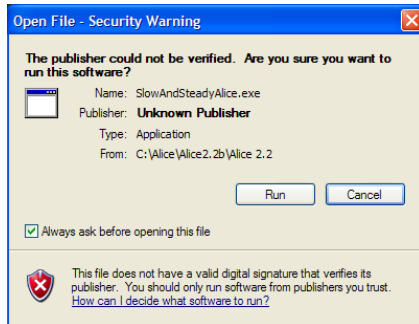
•If it's for a program that you trust then it's safe to proceed otherwise you should proceed with caution (probably select 'cancel')

Tutorials To Complete (In Tutorial)

Alice: Layout
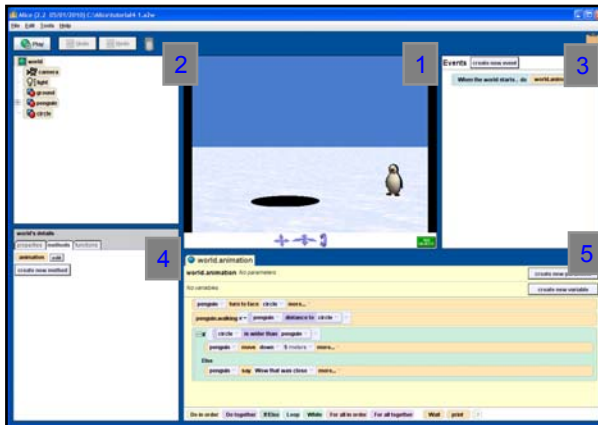
1. World
2. Object tree
3. World events
4. World details
5. Event (method) editor

---

1. **World**

- Your computer program will be written for a virtual world.

- There are different worlds that can be chosen but generally it's just the appearance that differs.

- Your view of the world can be changed using the controls.
  1. Up, down, left, right
  2. Forward, backward, left, right
  3. Tilt forward and backward

- (*JT's note: don't get to carried away with the camera controls*)
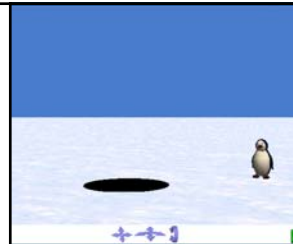
Perspective controls

2. **Object Tree**

• The world contains objects that are shown in tree form.

• Although your world will consist of three objects by default (camera, light, ground) it's the extra objects that you add to this world that will be of interest most of the time.
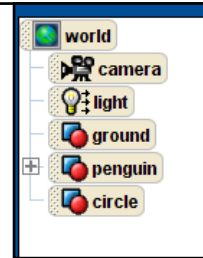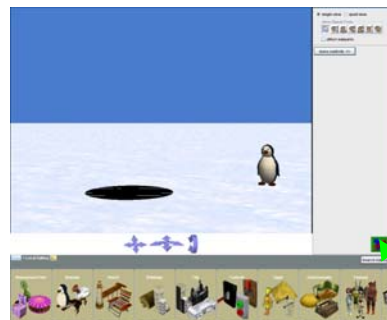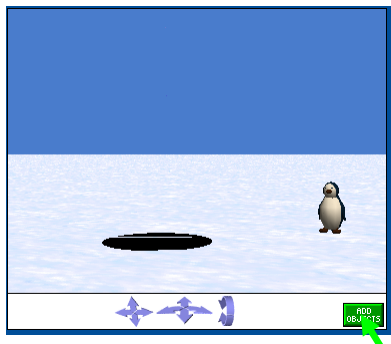
  - (In the example to the right the extra objects: penguin, circle).

Adding Objects

3. <u>World Events</u>

- •Different events can occur in the world

create new event
When the world starts
When a key is typed
When the mouse is clicked on something
While something is true
When a variable changes
Let the mouse move <objects>
Let the arrow keys move <subject>
Let the mouse move the camera
Let the mouse orient the camera
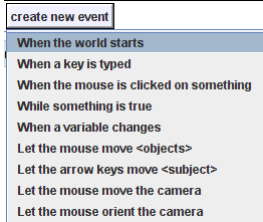
- •The event that we'll focus on is when the world starts (simulation begins running).

---

4. <u>World Details</u>

world's details
properties **methods** functions
animation edit
create new method

- •Used to see the details of the virtual world (or objects in the world).

- •Details:
  - Physical characteristics ("properties" in Alice).
  - Actions ("methods" in Alice).
  - Actions that perform an action and generates a value ("functions" in Alice).

Properties Of An Example Object: Penguin

•Properties provide information about an object

penguin's details

properties | methods | functions

create new variable

capture pose

color =
opacity = 1 (100%)
vehicle = world
skin texture = penguin.texture
fillingStyle = solid
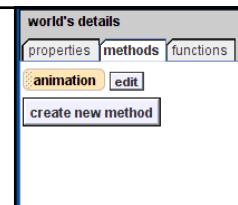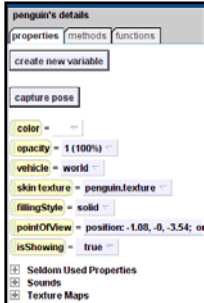pointOfView = position: -1.08, -0, -3.54; or
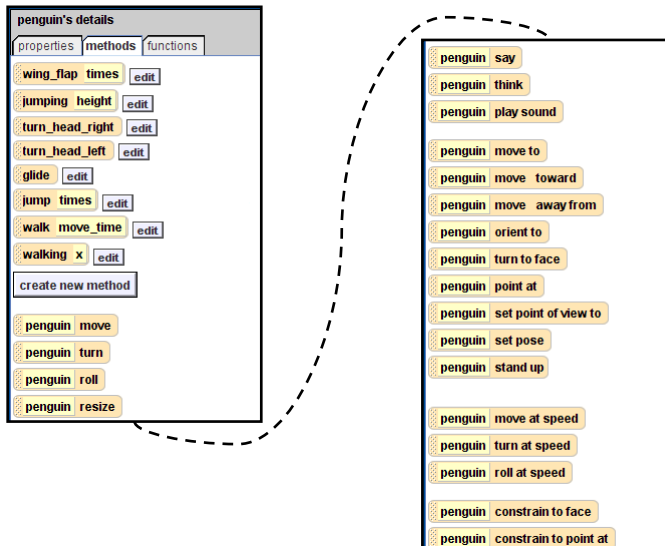isShowing = true

⊞ Seldom Used Properties
⊞ Sounds
⊞ Texture Maps

•Many (most) properties can be changed.

Methods Of An Example Object: Penguin

•Methods are the actions that an object is capable of carrying out.

penguin's details

properties | methods | functions

wing_flap times edit
jumping height edit
turn_head_right edit
turn_head_left edit
glide edit
jump times edit
walk move_time edit
walking x edit

create new method

penguin move
penguin turn
penguin roll
penguin resize

penguin say
penguin think
penguin play sound

penguin move to
penguin move toward
penguin move away from
penguin orient to
penguin turn to face
penguin point at
penguin set point of view to
penguin set pose
penguin stand up

penguin move at speed
penguin turn at speed
penguin roll at speed

penguin constrain to face
penguin constrain to point at

Functions Of An Example Object: Penguin

•Functions are typically asking a question (logical statement).

penguin's details
properties | methods | functions

⊟ proximity
penguin is within threshold of object
penguin is at least threshold away from object
penguin distance to
penguin distance to the left of
penguin distance to the right of
penguin distance above
penguin distance below
penguin distance in front of
penguin distance behind
⊟ size
penguin 's width
penguin 's height
penguin 's depth
penguin is smaller than
penguin is larger than
penguin is narrower than
penguin is wider than
penguin is shorter than
penguin is taller than

⊟ spatial relation
penguin is to the left of
penguin is to the right of
penguin is above
penguin is below
penguin is in front of
penguin is behind
⊟ point of view
penguin 's point of view
penguin 's position
penguin 's quaternion
⊟ other
penguin 's current pose
penguin 's part named key
penguin 's variable named variableName of type valueClass

James Tam

Some Useful Functions Of The World

•Logical operations ("Boolean logic", "math")

•Generating random numbers

•Prompting the user of the program for information ("ask user")

🌐 world
📷 camera
💡 light
ground

world's details
properties | methods | functions

create new function

⊟ boolean logic
not a
both a and b
either a or b , or both
⊟ math
a == b
a != b
a > b
a >= b
a < b
a <= b
⊟ random
choose true probabilityOfTrue of
random number
⊟ string
a joined with b
what as a string
⊟ ask user
ask user for a number
ask user for yes or no
ask user for a string

5. Event (Method) Editor

- Recall: a computer program consists of a series of instructions
  - (e.g., save document, print document, spell check etc.)
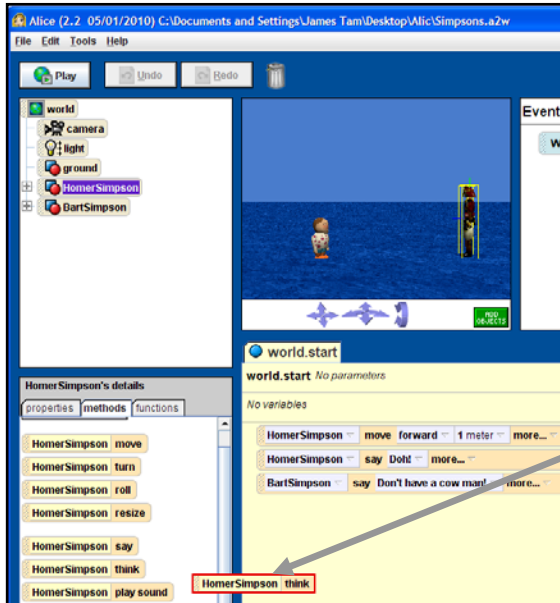- This is where you use the methods, functions and the other parts of Alice to 'program' (create) those instructions.
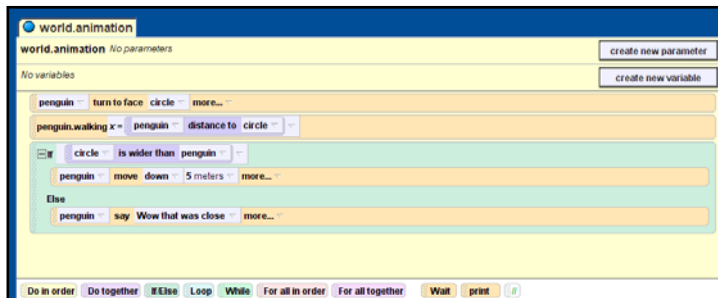
Programs Are Written In A Narrative Story Form

- Basic instructions are provided in a linear fashion (one after another).
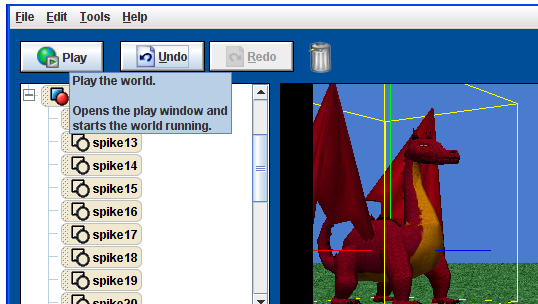
Instructions Are Given To The Program Visually

Dragging an instruction from the available list into the editor.

Running Your Program

• This is when the instructions in your program are executed.
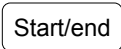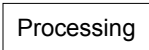
When Your Program Starts

- The instructions in a particular method e.g., "my first method" are executed.
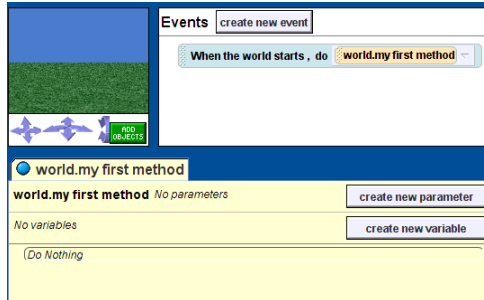
- Although actual programs contain many methods, for the purposes of this class you only have to put all your instructions into this one method.

- Rename to a sensible name e.g., "start", "main"

Flowcharts

- It's a graphical notation that's used to specify how a computer program will execute.

- Symbols:
  - Terminal (for the start and end of the program). Start/end

  - Processing (for a step in the program) Processing

- Arrows are used to connect the symbols (show the order and direction of execution).

Start/end

Instruction

End
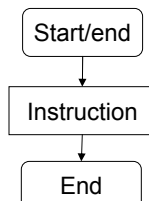
Example Program: Flowchart

world.start			
world.start *No parameters*			
No variables			
HomerSimpson ▽	move forward ▽ 1 meter ▽	more... ▽	
HomerSimpson ▽	say Doh! ▽ more... ▽		
BartSimpson ▽	say Don't have a cow man! ▽ more... ▽		

Start

↓

Homer: move forward 1 meter

↓

Homer: "Doh!"

↓

Bart: "Don't have a cow man!"

↓

End
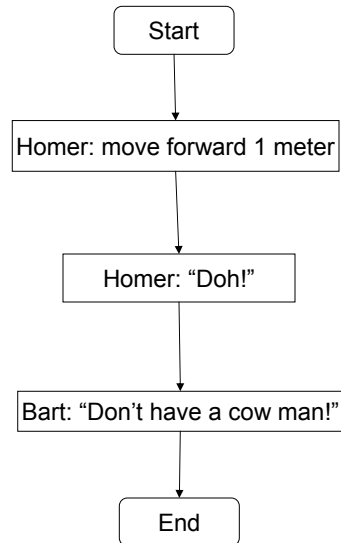
Having Instructions Execute Simultaneously

• By default instructions in Alice will execute one after another.

1	HomerSimpson ▽	move forward ▽ 1 meter ▽	more... ▽
2	HomerSimpson ▽	say Doh! ▽ more... ▽	
3	BartSimpson ▽	say Don't have a cow man! ▽	more... ▽

• Sometimes you may want to have two instructions execute at the same time e.g. characters wave to each other simultaneously.

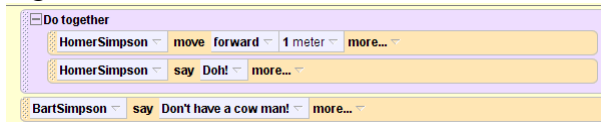• Simultaneous actions can be specified through the "do together" instruction.

Do in order Do together If/Else Loop While For all in order For all together Wait
print //

Having Programs Execute Simultaneously (2)

•The instructions that execute at the same time must be grouped together with the "do together" instruction.

Variables

•Are used by a program to temporarily store information

•Many types of information that can be stored by a variable in Alice.
  - For this class you will be mostly concerned with: numbers, Booleans (true or false), string (series of characters).
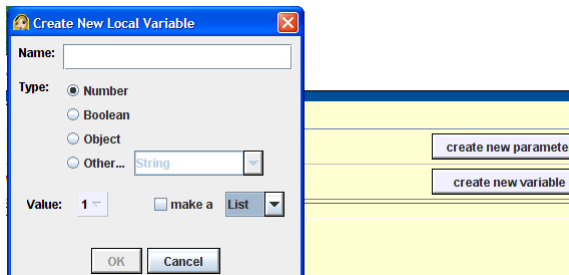
Example Use Of A Variable

Distance is a variable with an initial value of 1.

The no. of meters forward that the troll moves is determined by the current value stored in distance (1).

`123 distance = 1 ▽`

`troll ▽  move  forward ▽  distance meters ▽  more... ▽`

`distance ▽  set value to  5 ▽ more... ▽`

`dragon ▽  move  forward ▽  distance meters ▽  more... ▽`

Variable distance now stores the number 5.

The no. of meters forward that the dragon moves is determined by the current value stored in distance (which is now 5).

Variable Naming Conventions

- Should be meaningful
  - E.g., 'x', 'y' vs. 'age', 'distance'

- Variable names should generally be all lower case.

- For variable names composed of multiple words separate each word by capitalizing the first letter of each word (save for the first word) or by using an underscore. (Be consistent!)
  - netIncome
  - gross_pay

Program Documentation

- English statements inserted into a computer program.

- They are not written in a programming language.

- They are for the reader of the program and won't be executed by the computer.

Creating Program Documentation

Create comments for the documentation graphically.

What Should Be In The Program Documentation

•What does the program do e.g., tax program.

•What are it's capabilities e.g., it calculates personal or small business tax.

•What are it's limitations e.g., it only follows Canadian tax laws and cannot be used in the US. In Canada it doesn't calculate taxes for organizations with a yearly gross earnings over $1 billion.

•Author

•What is the version of the program

What Should Be In The Program Documentation (2)

•If you don't use numbers for the different versions of your program then consider using dates (tie this with program features i.e., list the features of version 'x' of the program).
  - How does the program work.
  - This is often a description in English (or another high-level) language that describes the way in which the program operates.
  - The purpose of this description is to help the reader quickly understand how the program works.
  - Typically used to describe things that are not immediately self evident from the program code.

•For an example of a completely documented program see "The Simpsons" example program.

Mathematical Expressions

- Most any arbitrary mathematical expression can be created in Alice.
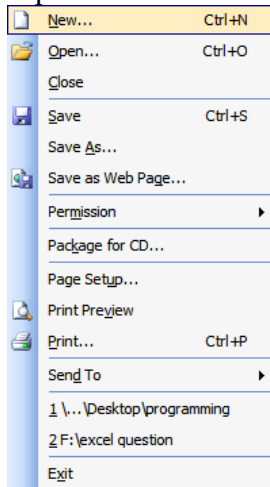- Expressions are created graphically.
- Example (from tutorial 3-3).
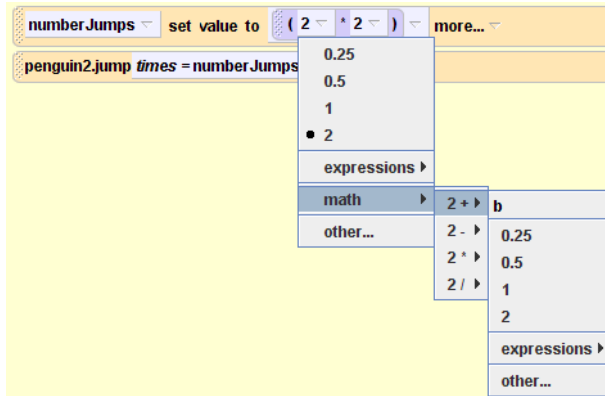
Common Mathematical Operations

Mathematical operation	Symbol used in Alice
Addition	+
Subtraction	-
Multiplication	*
Division	/

Choosing Among Alternative Course Of Action

• Frequently this is needed when the program must react to user input or the actions of the user.

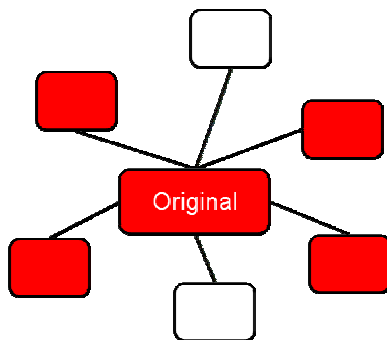

New: series of instructions

Open: series of instructions

Save: series of instructions

Etc.

# Choosing Among Alternative Course Of Action

• This can also be needed for other events e.g., the computer program determining the effects of randomly occurring events.


Original

Branching/Decision Making

- It's the mechanism used in many programming languages used to execute different instructions when alternatives are faced.

- At the branch (where the decision must be made) a logical statement determines which alternative occurs.
  - Another way of looking at it: a question is asked which has a true or false answer.
  - The answer to the question determines which branch executes.

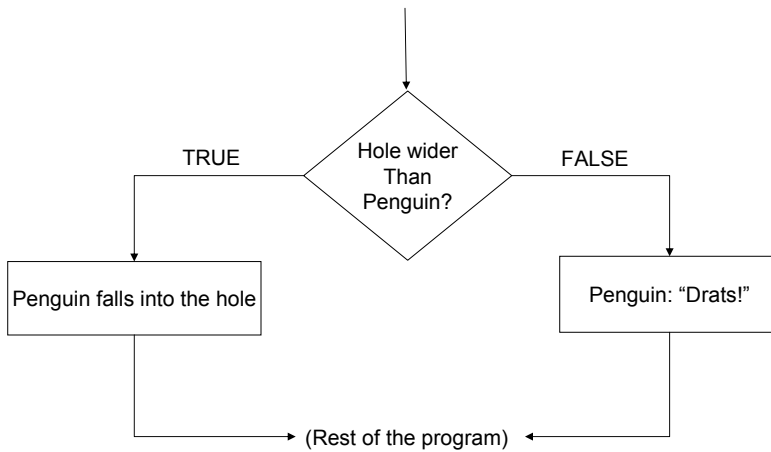- The flowchart symbol used to represent a branch/decision is a diamond.

Question
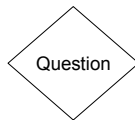
Branching/Decision Making (2)
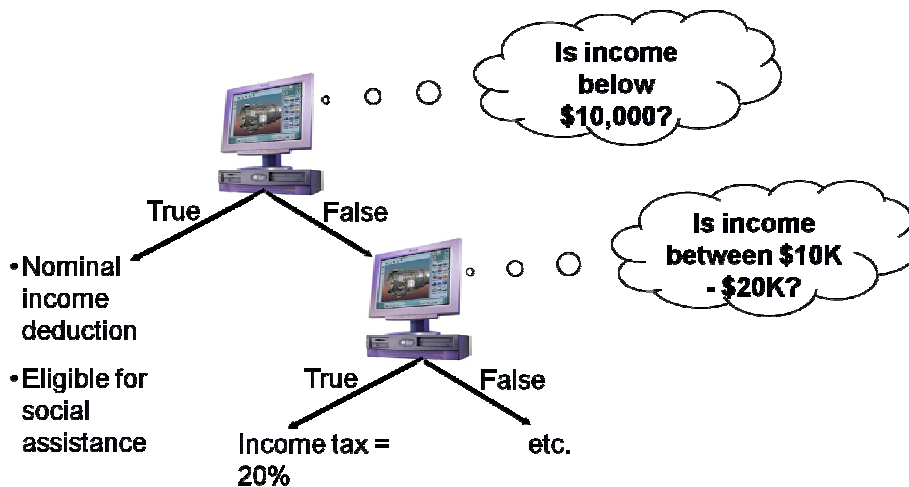
Example Of Decision
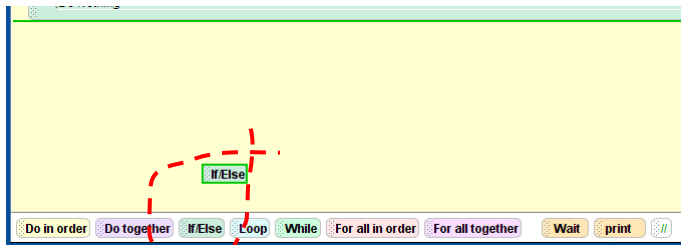
•From Tutorial 4-1:

•Does the penguin fall into the hole?

```
                              │
                              ▼
              ┌─────────────────────────────┐
   TRUE       │      Hole wider             │      FALSE
◄─────────────┤         Than                ├─────────────►
              │       Penguin?              │
              └─────────────────────────────┘
   │                                             │
   ▼                                             ▼
┌──────────────────────────┐        ┌──────────────────────────┐
│ Penguin falls into the hole │     │   Penguin: "Drats!"      │
└──────────────────────────┘        └──────────────────────────┘
   │                                             │
   │              ┌───────────────────┐          │
   └─────────────►│ (Rest of the program) │◄─────┘
                  └───────────────────┘
```
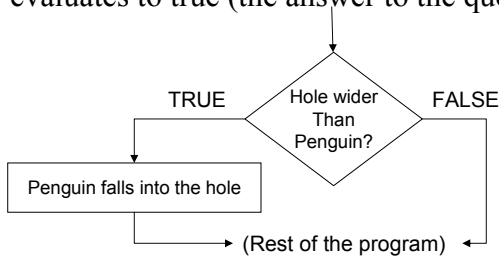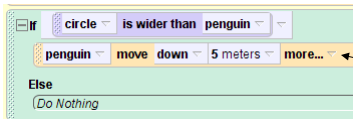

Decision Making In Alice

•Add an If/Else instruction

One Action: Do Something When True

• If you want the program to react when the logical statement evaluates to true (the answer to the question is true).


```
TRUE          Hole wider        FALSE
                Than
              Penguin?

Penguin falls into the hole

            (Rest of the program)
```

• Just use the 'if' and leave the 'else' portion empty.
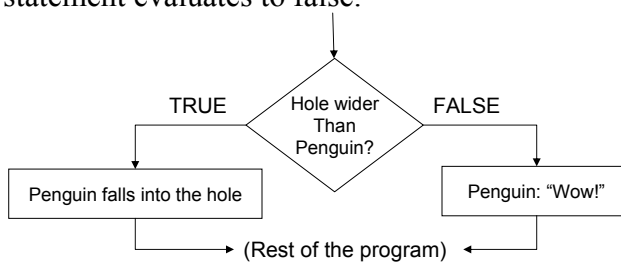

Indented part after the 'if': what executes if the statement is true

Two Actions: Do Something For Both True/False Cases

• You want the program to do something when the logical statement evaluates to true and something different when the statement evaluates to false.



```
TRUE          Hole wider        FALSE
                Than
              Penguin?

Penguin falls into the hole          Penguin: "Wow!"

            (Rest of the program)
```

• Add instructions for both the 'if' and the 'else' portion of the instruction.
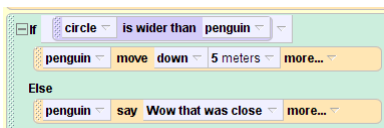
Logical Operators

- Are used in conjunction with logical statements and relate or compare things.

- The comparison will result in a true or false result.
  - E.g., temperature is above freezing (greater than zero).

- Accessing the logical operators in Alice.
 - Look under the functions of the 'world' object.
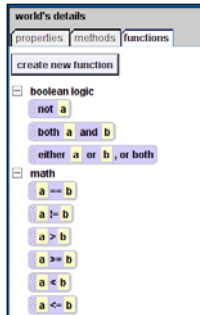
Types Of Logical Operators

Desired comparison	Operator (Alice)	Example
Equality	==	If age == 0
Not equal to (inequality)	!=	If age != 0
Greater than	>	If age > 19
Greater than, equal to	>=	If age >= 18
Less than	<	If age < 30
Less than, equal to	<=	If age <= 65

Multiple Branches: Independent

•If a number of questions must by asked by a program and the answer to one question has no impact on the answer to the other questions.

  - That means that all questions will be asked regardless of whether any questions answered true or false.
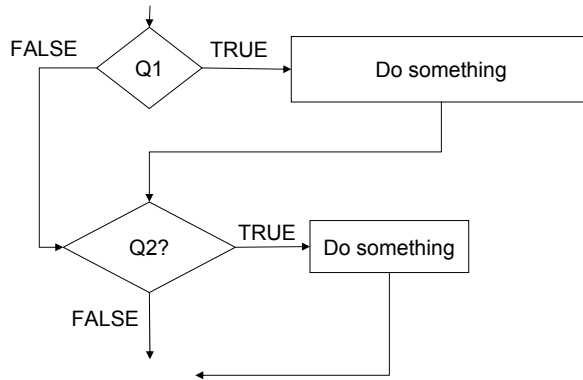
Multiple Branches: Independent (2)

•Multiple if/else statements can be used in succession.

•Example program: multiple_branches

Indented: executes if first 'if' is true
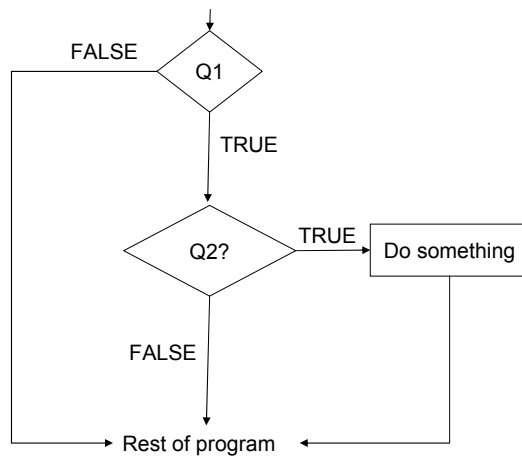
Not indented: execution is independent of the first 'if'

Nested Branches

•Sometimes the answer to one question will determine whether
or not other questions will be asked.
- The answer to one question (true or false) has an effect on whether another
  question is asked.
- The asking of the second question is dependent on the answer to the first
  question.
- (Alternatively: the second question is nested inside the first question).
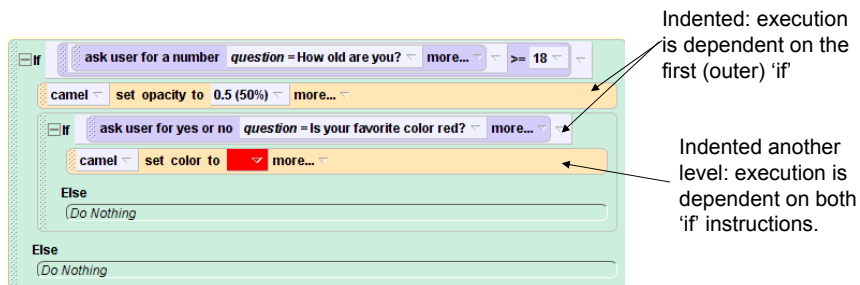
Nested Branches: Flowchart

Nested Branches: Alice

Indented: execution is dependent on the first (outer) 'if'

Indented another level: execution is dependent on both 'if' instructions.

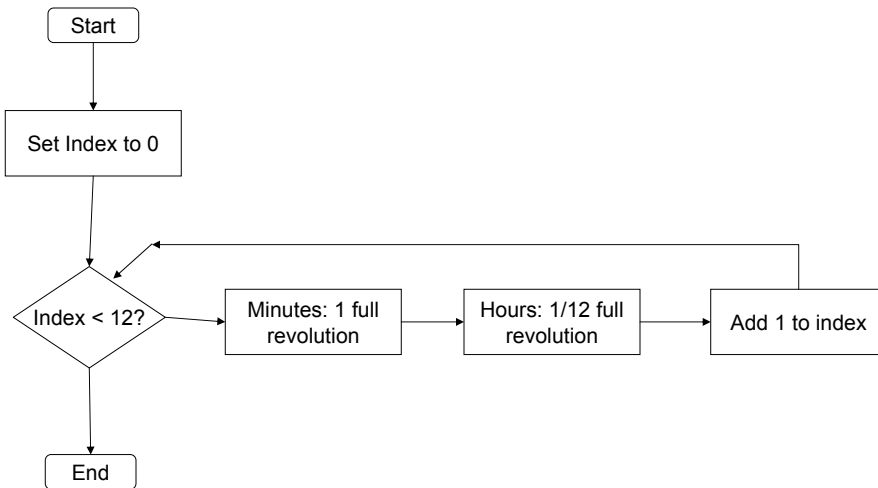
Loops

- Loops are used to repeat a part of the program (or potentially the entire program itself).

- They are similar to branches where a logical statement is employed.

- In the case of a loop if the logical statement evaluates to true then a part of the program will repeat (loop).

- The repetition will continue until the logical statement evaluates to false.
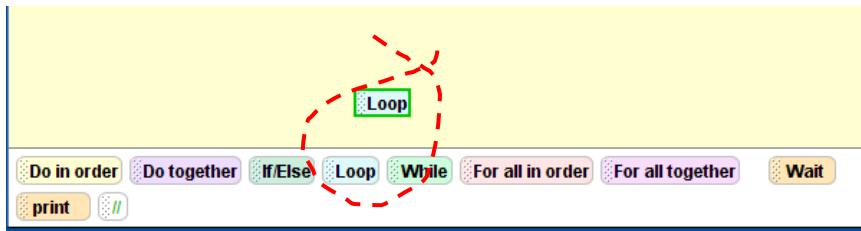
Loop Example: Flowchart

Start

↓

Set Index to 0

↓

Index < 12? → Minutes: 1 full revolution → Hours: 1/12 full revolution → Add 1 to index

↓

End

Loop Example: Alice

⊟ Loop **12** times ▽ **times** [show complicated version]

⊟ **Do together**

clock.minute ▽ **roll** **left** ▽ **1** revolution ▽ **more...** ▽

clock.hour ▽ **roll** **left** ▽ **0.08** revolutions ▽ **more...** ▽

Adding A Loop In Alice

Loop

| Do in order | Do together | If/Else | Loop | While | For all in order | For all together | Wait |

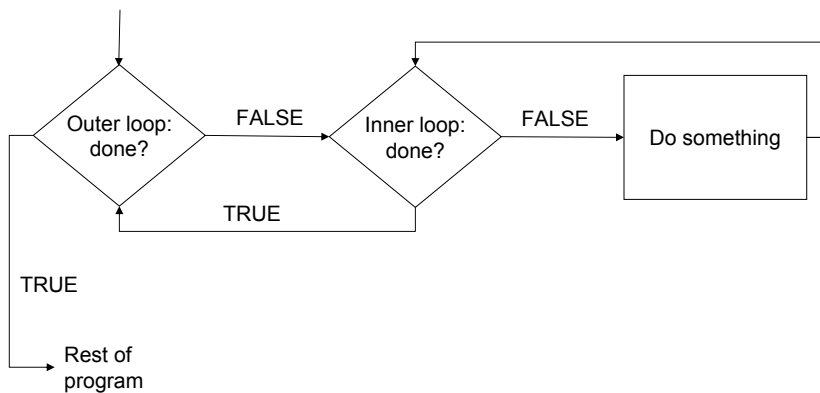| print | // |

Nesting: Loops And Branches

•Loops can be nested within branches (and branches can be nested within loops).

•Or one loop can be nested within another loop.

Outer loop: done? —FALSE→ Inner loop: done? —FALSE→ Do something

TRUE (from inner loop back)

TRUE

Rest of program

Example Of Loop Nesting

• A program that controls a car odometer

Innermost loop: counts
from beginning to end (0 –
9)

Next innermost loop: after
the innermost loop is done
increase this value by 1

Programming Type Questions: Exam

• Program writing
  - Given a description of what a program should do, you will be required to draw up a flowchart that shows the instructions that fulfill the requirements for that program.

• Tracing programs
  - Given some statements from an Alice program or a flowchart you should be able to determine what the program does when it executes e.g., how many times does it run, what output or effects does it produce given some inputs.

You Should Now Know

- How a computer program is written in a programming language must be translated before it can be executed
- How to create, save and run programs using Alice
- How to add new objects to the Alice world
- Alice objects have properties, methods and functions
- Common functions in the world object
- How to specify the instructions of a computer program visually using flowcharts
- The method for getting instructions in Alice to execute at the same time
- What is a variable and how/why are they used in Alice programs
- Good naming conventions for variables
- How to document a computer program and why is it important
- How to create a mathematical in Alice

You Should Now Know (2)

- How and why are branches used in computer programs
- How to create or trace a nested branch
- How and why are loops used in computer programs
- How to create or trace nested loops
- Commonly used logical operators