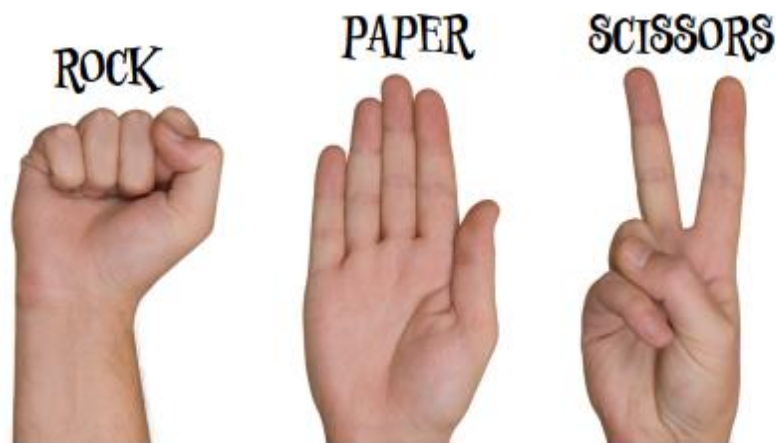# Python: Booleans & Decisions

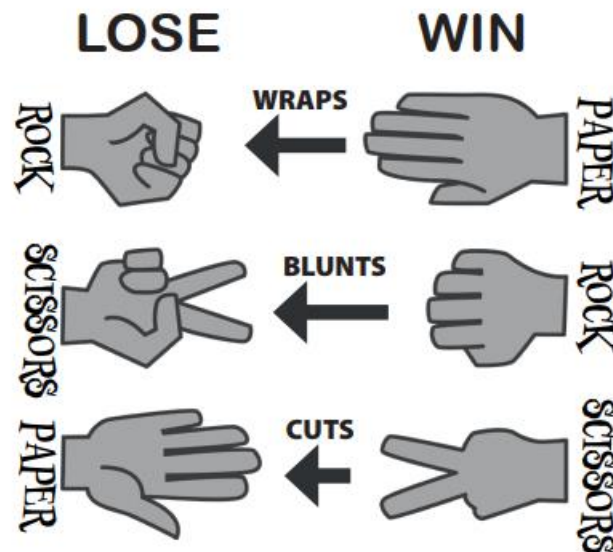# Would you like to play a game?

ROCK   PAPER   SCISSORS

There are three possible hand positions in the game Rock, Paper, Scissors.

LOSE   **WIN**

If one player chooses scissors and the other chooses rock, the rock blunts the scissors, and rock wins.

If one player chooses paper and the other chooses rock, the paper wraps the rock, and paper wins.

If both players make the same choice, then the game is a tie.

If one player chooses paper and the other chooses scissors, the scissors cut the paper, and scissors wins.

ROCK — **WRAPS** — PAPER

SCISSORS — **BLUNTS** — ROCK

PAPER — **CUTS** — SCISSORS

How you're going to play against the computer

Python 3.6.0 Shell

```
rock, paper or scissors? rock          ←——  Computer prompts you for rock, paper,
                                             or scissors and you respond with rock.

User won, I chose scissors.            ↖——  Computer determines you've won.

rock, paper or scissors? rock          ←——  Let's run it again.

We both chose rock, play again.
                                       ←——  Looks like a tie, so try again.
rock, paper or scissors? paper         ↙
                                       ←———  Computer wins this time.
Computer won, I chose scissors.
```

First, a high-level design

1 User starts the game.

    A The computer determines what its choice is going to be: rock, paper, or scissors.

2 Game play begins.

    A Get the user's choice.

    B Examine the user's choice. If it is invalid (not rock, paper, or scissors), go back to step 2A.

    If it is the same as the computer's, set the winner to a tie and move on to step 3.
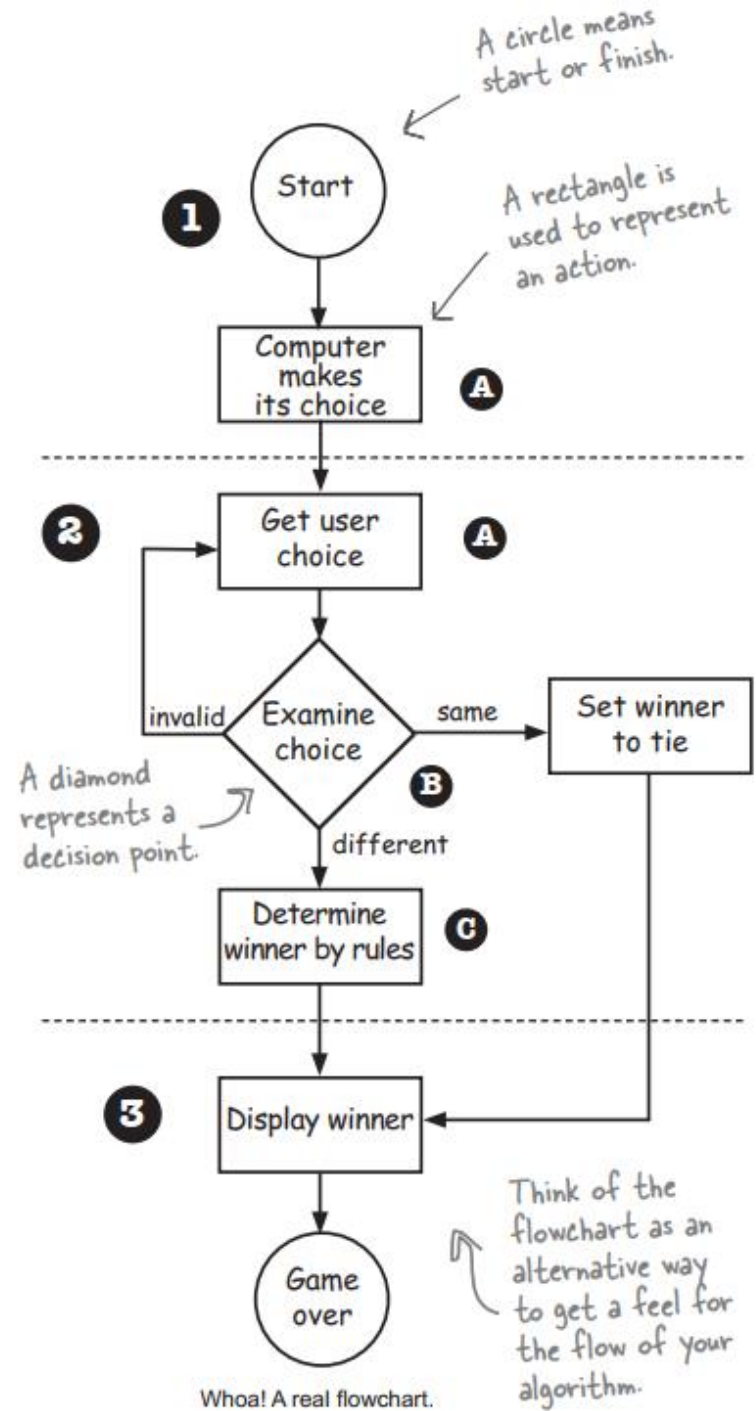
    C Determine who wins by the rules of the game.

3 Game finishes.

Tell the user who won along with what the computer's choice was.

First, a high-level design

A circle means start or finish.

A rectangle is used to represent an action.

A diamond represents a decision point.

Think of the flowchart as an alternative way to get a feel for the flow of your algorithm.

Whoa! A real flowchart.

The computer's choice

How to generate a random number

Use the import keyword first.

Follow that by the name of the module you want to use—in this case, the random module.

import random

Typically we place import statements at the top of the code file so that you can easily keep track of all the modules you're importing.

Remember, a module is just another file with Python code in it.

We start with the
name of the module—
in this case random.

Then we add a period
(otherwise known to
coders as a dot).

Then comes the
function name, randint.

So randint will return 0, ,1 or 2.

random.randint(0,2)

We pass randint
two numbers....

...this is a range, so randint
will give us random integers
between 0 and 2.

We're going to dive into all the specifics
of this notation later in the book, but
for now, just take it all in.

See for yourself. Get into the Python Shell, import the random module, and then generate some random integers with `random.randint(0.2)`.

Keep calling randint to see what you get!

```
Python 3.6.0 Shell
>>> import random
>>> random.randint(0, 2)
```

What will you get?

How to use the random number

Remember to import the random module.

We added an extra line (also known as "whitespace") just to separate the imported module from the actual code we're writing. That should help with readability as our code grows.

```
import random

random_choice = random.randint(0,2)
```

And we can't forget to assign our random number to a variable so we can actually make use of it later in code.

Here's where we generate the random number to represent the computer's choice.

A Test Drive

Just to get things rolling, go ahead and get this code into a file called *rock.py*, save your code, and choose the **Run > Run Module** menu item to make sure everything's working.

Here's the code so far:

Add this new code to your file.

```python
import random
```

We added an extra line just to provide some output.

```python
random_choice = random.randint(0, 2)
print('The computer chooses', random_choice)
```

Here's what we got. You might want to try it a few times to see the choices are random.

```
Python 3.6.0 Shell

The computer chooses 2
>>>
```

Sharpen your pencil

Assume `random_choice` is already set to 0, 1, or 2 and write some pseudocode to set the variable `computer_choice` to "rock", "paper", or "scissors" based on `random_choice`'s value.

You don't know how to code this in Python yet, but remember pseudocode uses English-like language. Don't overthink it.

True? Or False?

Here's an expression comparing the values of two variables that hold numbers.

We call this a relational operator; in this case it's the greater than operator, which is True if the first operand is greater than the second, and False if not.

We're going to look at other relational operators like "less than," "equal to," and so on in a bit.

bank_balance > ferrari_cost

You can read this as "is the bank_balance greater than the ferrari_cost?"

The result of this expression is either True or False depending on whether the bank balance is greater than the cost of the ferrari.

Again, this is the expression.

```
decision = bank_balance > ferrari_cost
print(decision)
```

False

decision

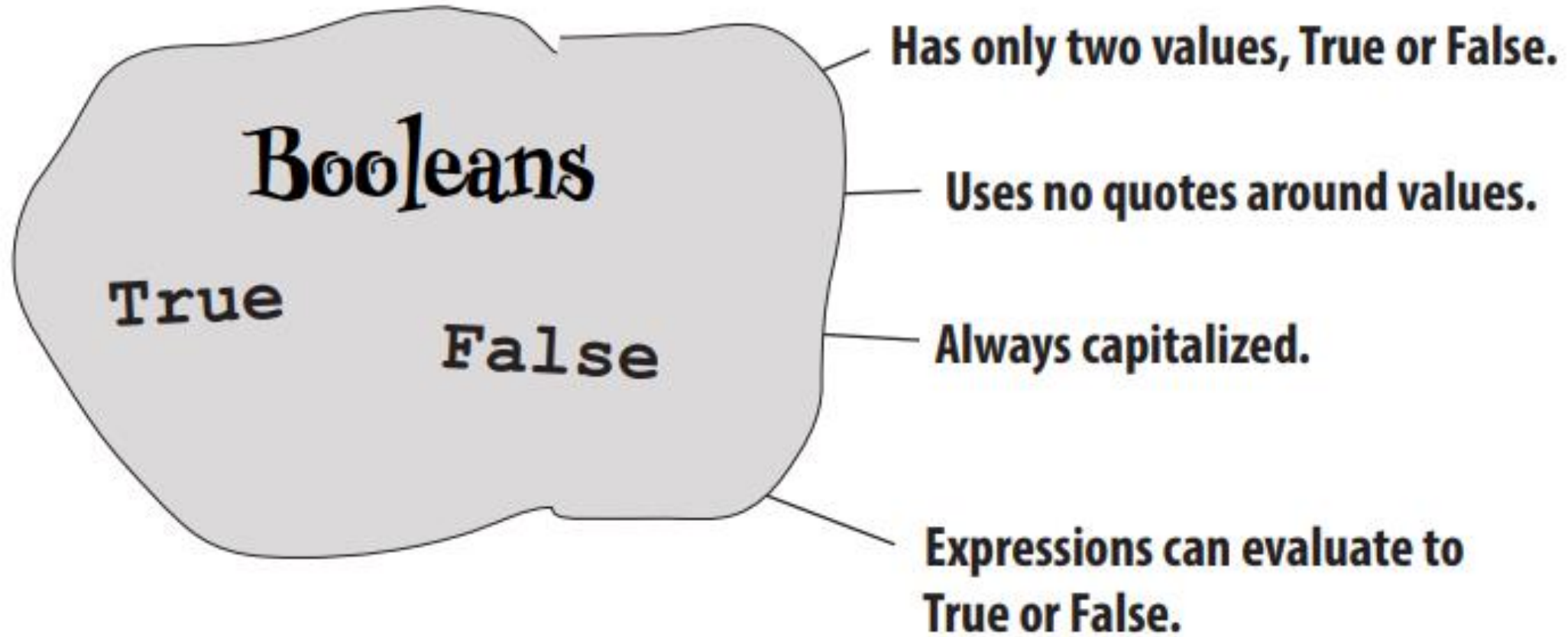The decision variable holds the (True or False) value of the comparison.

Here's the output.

Python 3.6.0 Shell

```
False

>>>
```

No Ferrari today.

Introducing the Boolean type

Booleans

True

False

Has only two values, True or False.

Uses no quotes around values.

Always capitalized.

Expressions can evaluate to True or False.

Sharpen your pencil

Get out your pencil and put some Boolean expressions through their paces. For each expression below, compute its value and write in your answer. Be sure to check your answers at the end of the chapter. Remember, Boolean expressions always evaluate to either `True` or `False`.

This tests if the first value is greater than the second. You can also use >= to test if the first value is greater than or equal to the second.

your_level > 5

When `your_level` is 2, what does this evaluate to? _____

When `your_level` is 5, what does this evaluate to? _____

When `your_level` is 7, what does this evaluate to? _____

The == operator tests if two values are equal to each other. It's True if they are and False if not.

color == "orange"

Is this expression `True` or `False` when color has the value "pink"? _____

Or has the value "orange"? _____

The != operator tests if two values are NOT equal to each other.

color != "orange"

Is this expression `True` or `False` when `color` has the value "pink"? _____

Making decisions

```
if bank_balance >= ferrari_cost:
    print('Why not?')
    print('Go ahead, buy it')
else:
    print('Sorry')
    print('Try again next week')
```

You'll find anytime we have a colon in Python, it is followed by an indented set of statements.

We add an else keyword.

Next we have a colon.

Then we have one or more statements that will be executed if the condition is False.

Notice that all the statements we want executed when the conditional is False are indented four spaces too.

Decisions and more decisions

Start with your first condition, using an if keyword.

```python
if number_of_scoops == 0:

    print("You didn't want any ice cream?")

    print('We have lots of flavors.')

elif number_of_scoops == 1:

    print('A single scoop for you, coming up.')

elif number_of_scoops == 2:

    print('Oh, two scoops for you!')

elif number_of_scoops >= 3:

    print("Wow, that's a lot of scoops!")

else:

    print("I'm sorry I can't give you negative scoops.")
```

Follow that with an elif keyword and a second condition.

And then add any number of other elifs with their own conditions.

Remember, for each if, elif, and else, we can supply as many statements to execute as we like.

And finally, you can supply a final else, which acts as a catch-all if all previous conditions fail.

Note that only the code of the first True condition will be executed, or if no conditions are True, the else's code will be executed.

```
import random

random_choice = random.randint(0,2)
print('The computer chooses', random_choice)


if random_choice == 0:
    computer_choice = 'rock'
elif random_choice == 1:
    computer_choice = 'paper'
else:
    computer_choice = 'scissors'

print('The computer chooses', computer_choice)
```

We don't need this anymore, so you can delete it.

Add this new code to your file rock.py.

Check to see if random_choice is 0 and if so, set the computer's choice to the string "rock".

Otherwise, check to see if random_choice is 1 and if so, set the computer's choice to the string "paper".

Otherwise, the only choice left is "scissors".

And just to test things, let's print out computer_choice.

We're still here.

Start

Computer makes its choice
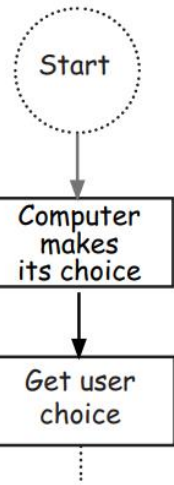
Get user choice

Getting the user's choice

```
import random

random_choice = random.randint(0, 2)

if random_choice == 0:
    computer_choice = 'rock'
elif random_choice == 1:
    computer_choice = 'paper'
else:
    computer_choice = 'scissors'
```

We're assigning the string returned from the input function to the variable user_choice.

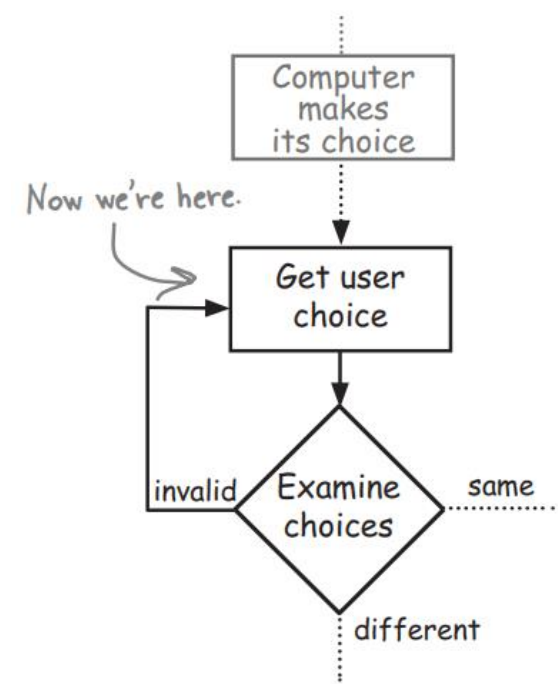~~print('The computer chooses', computer_choice)~~

We don't need this debugging print statement anymore.

We're using the input function again, and prompting for the user's choice in the game.

```
user_choice = input('rock, paper or scissors? ')
print('You chose', user_choice, 'and the computer chose', computer_choice)
```

Let's add a print statement just to keep track of things as we're coding this.

Computer makes its choice

Now we're here.

Get user choice

invalid / Examine choices \ same

different

First create a list of choices, which is just a list of strings.

Dont' be confused by the square brackets; we'll be learning about lists in the next chapter.

```python
choices = ['rock', 'paper', 'scissors']
computer_choice = random.choice(choices)
```

Then we pass our list to the choice function, which will randomly choose one item for us.

Taking a look at the user's choice

Get user choice

invalid   Examine   same   Set winner
           choices   to tie

Now we're
here.   different

☐ The user and the computer made the same choice, and we have a tie.

☐ The user and the computer made different choices, and we need to determine who won

☐ The user entered an invalid choice, and needs to enter another choice.
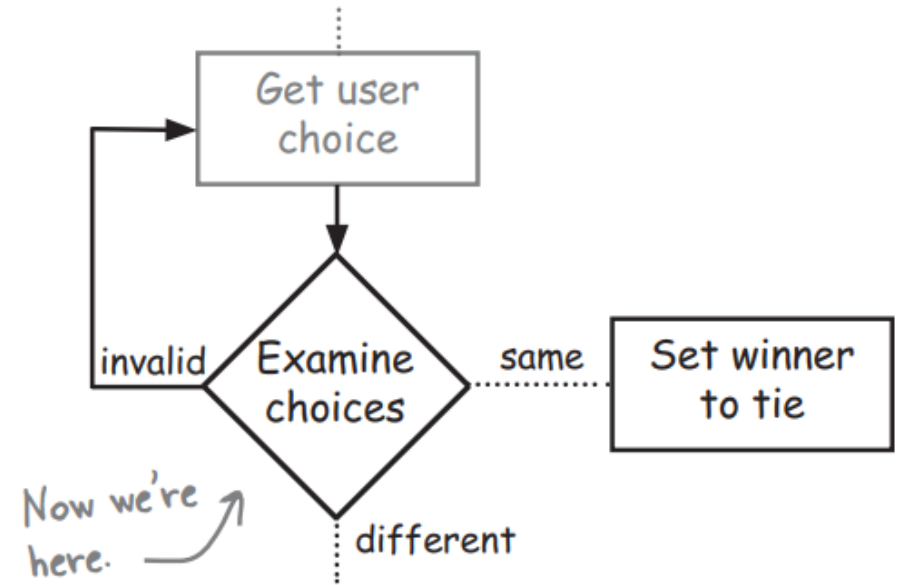
Sharpen your pencil

Your turn again. Based on our plan on the previous page, finish the code fragment below. Your code should determine if there is a tie, and if so, set the `winner` variable to `'Tie'`. After you've completed this exercise, we'll get this code into the *rock.py* file in the next step.

```
if _____ == _____:
    winner = _____
```

Adding the code to detect a tie

```python
import random

winner = ''

random_choice = random.randint(0,2)

if random_choice == 0:
    computer_choice = 'rock'
elif random_choice == 1:
    computer_choice = 'paper'
else:
    computer_choice = 'scissors'

user_choice = input('rock, paper or scissors? ')
print('You chose', user_choice, 'and the computer chose', computer_choice)

if computer_choice == user_choice:
    winner = 'Tie'
```

How to implement the game logic

`computer_choice == 'paper'`

A simple Boolean expression that we're familiar with at this point, which asks if the computer's choice is paper

And if the user chose rock:

`user_choice == 'rock'`

And another expression asking if the user's choice is rock

Here's our first condition.

Here's our second condition.

computer_choice == 'paper' and user_choice == 'rock'

This entire phrase is a Boolean expression and will evaluate to either True or False.

Placing an and operator between them means this entire expression will be True if and only if both conditions are True.

Now the if statement's conditional expression is the entire combined Boolean expression.

```python
if computer_choice == 'paper' and user_choice == 'rock':
    winner = 'Computer'
```

So this code handles one of the three cases where the computer wins.

If the expression is True, then we execute the if's code block.

More about Boolean operators

You've got the money for the Ferrari if you have enough money in the bank balance...

OR, you have a loan that is equal to the cost of the Ferrari.

```
if bank_balance > ferrari_cost or loan == ferrari_cost:
    print('Buy it!')
```

Note that only one of these conditions needs to be True to get the Ferrari, but both can be True as well. If both are False, then you'll have to wait for the Ferrari.

Here we've put a not in front of a Boolean expression.

First we evaluate this relational operator to True or False, and then the not operator is applied, evaluting to the opposite Boolean value.

```
if not bank_balance < ferrari_cost:
    print('Buy it!')
```

You can read this as "if the bank account is NOT less than the Ferrari cost," then buy it.

Sharpen your pencil

Take out your pencil and put some more Boolean expressions through their paces. For each expression below, compute its value and write in your answer.

`age > 5 and age < 10`

When age is 6, what does this evaluate to? _____

When age is 11, what does this evaluate to? _____

When age is 5, what does this evaluate to? _____

`age > 5 or age == 3`

Notice we added parens here, which makes this more readable.

When age is 6, what does this evaluate to? _____

When age is 2, what does this evaluate to? _____

When age is 3, what does this evaluate to? _____

`not (age > 5)`

When age is 6, what does this evaluate to? _____

When age is 2, what does this evaluate to? _____

```python
if      computer_choice == user_choice      :
        winner      =      'Tie'

elif    computer_choice == 'paper'      and      user_choice == 'rock'      :
        winner = 'Computer'

else    :
        winner == 'User'
```

You're lucky they didn't knock off the first test to see if the computer won, and the very last part where the user wins.

Magnets available:

`:` `elif` `if` `and` `and` `and` `or` `elif` `and` `and` `or` `if` `:` `elif` `if` `:` `=`

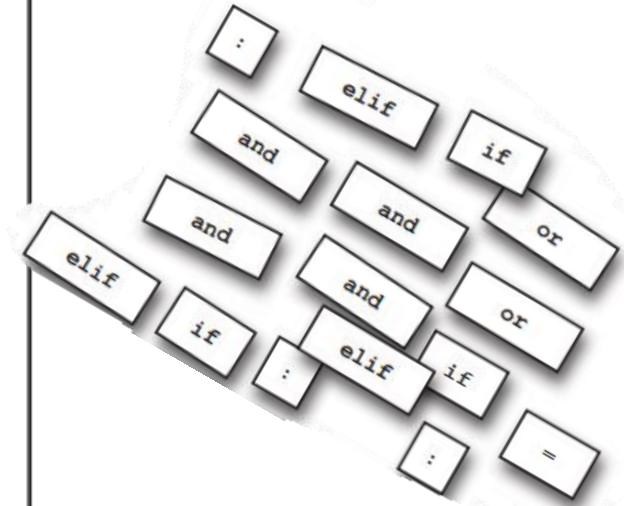`winner = 'Computer'` `winner = 'Computer'` `computer_choice == 'rock'`
`user_choice == 'paper'`
`user_choice == 'scissors'` `computer_choice == 'scissors'` `winner = 'User'`

Place your magnets here.

```python
import random

winner = ''

random_choice = random.randint(0,2)

if random_choice == 0:
    computer_choice = 'rock'
elif random_choice == 1:
    computer_choice = 'paper'
else:
    computer_choice = 'scissors'

user_choice = input('rock, paper or scissors? ')

if computer_choice == user_choice:
    winner = 'Tie'
elif computer_choice == 'paper' and user_choice == 'rock':
    winner = 'Computer'
elif computer_choice == 'rock' and user_choice == 'scissors':
    winner = 'Computer'
elif computer_choice == 'scissors' and user_choice == 'paper':
    winner = 'Computer'
else:
    winner = 'User'

print('The', winner, 'wins!')
```
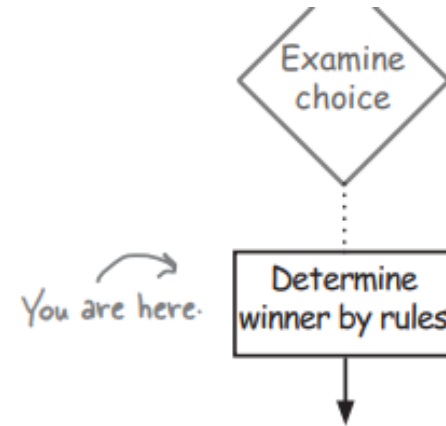
Examine choice

You are here. Determine winner by rules

Here's the code for the game logic. Go ahead and enter this code.

Here we're doing some setup by importing the random module and setting up the winner variable.

```python
import random

winner = ''
```

The computer randomly chooses rock, paper, scissors by generating a random number from 0 to 2 and then mapping that to a corresponding string.

```python
random_choice = random.randint(0,2)

if random_choice == 0:
    computer_choice = 'rock'
elif random_choice == 1:
    computer_choice = 'paper'
else:
    computer_choice = 'scissors'
```

Get the user's choice with a simple input statement.

```python
user_choice = input('rock, paper or scissors? ')
```

Here's our game logic, which checks to see if the computer wins (or not), and makes the appropriate change to the winner variable.

```python
if computer_choice == user_choice:
    winner = 'Tie'
elif computer_choice == 'paper' and user_choice == 'rock':
    winner = 'Computer'
elif computer_choice == 'rock' and user_choice == 'scissors':
    winner = 'Computer'
elif computer_choice == 'scissors' and user_choice == 'paper':
    winner = 'Computer'
else:
    winner = 'User'
```

Here we announce the game was a tie, or the winner along with the computer's choice.

```python
if winner == 'Tie':
    print('We both chose', computer_choice + ', play again.')
else:
    print(winner, 'won, I chose', computer_choice + '.')
```