Table of Contents

1 You're going to learn programming and how to build a game with a language called Python.

> ### Did you know?
>
> Websites like YouTube and Instagram are built using Python.

Before you can start coding, you'll need to setup your computer to write Python. You'll need to have Python and a text editor installed, and know how to run a Python program.

2 To install Python, go to dojo.soy/py-setup and click on the *Download Python 3* button. There will be some other numbers after the *3*, but they change too often for me to include them. Don't worry about them.

Once the installer has downloaded, start it and click through it, accepting the default choices.

3 Now you need to get a text editor, to write your Python in. We recommend Atom, which you can download from http://atom.io, but you can use another editor if you're more familiar with it. Or just like it better.

4 Once you have both of these setup, you're ready to go. You just need to make sure that everything is working and that you know how to run a Python program. Follow these steps:

- Make a new folder for your Python Sushi Cards projects.
- Open your text editor and create a new file. Save it into the folder you just made and call it *beginner_sushi.py*.
- Open the command line on your computer (called *command prompt* on Windows and *Terminal* on Mac) and navigate to your folder using the *cd* command.
- Once you've opened your folder in the command line, you're ready to try running this blank file with this command, entered into the command line:

```
python3 beginner_sushi.py
```

If this has worked, you shouldn't see any messages when you run the command.

> ### Running a program
>
> In later cards, you'll be asked to *run* a program.
> What that means is what you've just done: getting to the folder that has the program file in it and using the *python3* command and its name to make the program do its thing!

CoderDojo

Your first Python program

I'm learning: Python

1 Time for your first bit of Python. You're going to get the computer to say hello to everyone. Type this into your code file:

```
print("Hello everyone")
```

Run this code and see what happens!

Try changing what's inside the `"` symbols, maybe by adding your name, and running it again.

2 Now add another line: Try making your code look like this:

```
print("Hello everyone")
print("The Code, it's calling to you. Just let it in.")
```

Run it again.

See how the text (called a **string**) from the second `print` is on a new line? This is because the instruction the computer gets when you tell it to `print` is:

- Read the code in the brackets and figure out the result
- Once you've figured out what it says, `print` that out on the screen.
- Put an invisible "start a new line" instruction at the end.

3 Why does the computer need to figure out what the code in those brackets says? It's because the computer can put that **string** together from parts you give it.

Try it! Use this code, but put your name in where it says "my name" (keep the *"* characters though!):

```
name = "my name"
print("Hello "+name)
print("The Code, it's calling to you. Just let it in.")
```

> **The space after "Hello"**
>
> You have to include a space before the variable or you'll just get "Hellomy name"! Python doesn't know what English looks like!

4 You made a **variable**, called `name`. This is like a box inside the computer, with a label on it. You can put anything you want in it. Then, you can use the label to get Python to go fetch the thing that's in the box and use it in your code. You created the `name` variable and stored *"[my name]"* in it. On the next line, you used the variable to stick that name into your greeting, by using the `+` symbol to attach it to the end of the **string**.

1 Getting the computer to stick your name on the end of `"Hello "` is nice, but why not just write `"Hello [my name]"`? Because with a **variable** you don't have to know what's going into it when you write the program. your can even ask the user of the program to tell you what to put into it. Update your Python program to do that:

```
name = input("What is your name?")

print("Hello "+name)

print("The Code, it's calling to you. Just let it in.")
```

Try running it. You'll need to press the "Enter" key once you've typed in your name.

2 Now, try collecting a number from your user. Notice that you can use the `+` on both sides of a variable.

Run this program, answer its questions and watch what happens.

```
name = input("What is your name?")

my_number = input("Hello "+name+", please pick a number")

print("Your number is "+my_number)
```

3 What if you want to add a number to your variable? Add a line to your program that will add one to the `my_number` variable.

```
name = input("What is your name?")

my_number = input("Hello "+name+" please pick a number")

my_number = int(my_number) + 1

print("Your number is "+str(my_number))
```

> See that you've taken a value from a variable, changed it and stored it back in the same variable all on the same line!

Now, why does the code have `int()` and `str()` around `my_number`?

It's because Python thinks of the number '1', which it uses for maths, differently to the number '1' which it writes in a sentence. Putting *int( )* around a variable tells it to treat it as an **integer** (a maths number), and putting *str( )* around it tells it to treat it as a text **string**.

Integers and **strings** are variable **types** and certain pieces of code (like `+` and `print`) only work if the variables you give them are the right type.

> ## Maths
>
> You've seen how to add here, but you can also:
>
> - Subtract using -
> - Multiply using *
> - Divide using /

CoderDojo

Comparison & if statements

I'm learning: Python

1 You can ask Python to compare one number to another number. This can be really handy (does the player have as much money as those pants cost?). You do this using special symbols:

- `a > b` asks if `a` is bigger than `b`
- `a < b` asks if `a` is smaller than `b`
- `a == b` asks if `a` is the same as `b`
- `a != b` asks if `a` is not the same size as `b`
- `a >= b` asks if `a` is bigger than, or the same size as `b`
- `a <= b` asks if `a` is smaller than, or the same size as `b`

> **==**
>
> The double-equals is used to **compare** variables, because the single equals is already used **assign** values to them.

2 You use these comparisons is inside `if` statements: code that should only run if a condition (in the brackets) is true. In this case, **printing** some text.

```
if(my_number > 100):
    print("That's a big number!")
```

> ### Indentation
>
> The `print` is **indented**. That means that four spaces have been put before it. Python needs these spaces to understand your program.

3 Now, put that little bit of code together with your program from the last card. Change the program so it looks like this:

```
name = input("What is your name?")
my_number = input("Hello "+name+" please pick a number")
my_number = int(my_number)
print("Your number is "+str(my_number))

if(my_number > 100):
    print("That's a big number!")
```

Now run it and try entering different numbers, above and below 100 to see what happens. What would happen if you entered 100 exactly?

4 You can also combine conditions, using `and` and `or`. So this let's you write code like:

```
if(my_number >= 20 and my_number < 30):

    print("That number is in the twenties!")
```

Or, for example:

```
if(food == "Cake" or food == "Chocolate" or food == "Pie"):

    print("Sounds tasty!")
```

CoderDojo

Fancy if statements

I'm learning: Python

1 What if you want to check if the user's number is big enough, and tell them if it's not? Say if it's bigger than 100. Then, either congratulate the user on giving a number that's big enough, or tell them where they went wrong. Try this:

```
name = input("What is your name?")
my_number = input("Hello "+name+" please pick a number that's bigger than 100")
my_number = int(my_number)
print("Your number is "+str(my_number))

if(my_number > 100):
    print("That's a big number!")
else:
    print("That number is too small!")
```

Here the `else` works like an `if` statement where the condition is "the thing in the if isn't true"

2 What if you want to tell the user if they're close? Say if they've picked a number over 90? Then you use an `elif`. That's `else` and `if` stuck together, because it happens only if the thing in the `if` statement isn't true and if the thing in the `elif` statement's brackets is true. So here's what you add to get the program to tell the user they're close:

```
elif(my_number > 90):
    print("Almost there!")
```

And here's what it looks like with the rest of the program. Notice that the `elif` has to come between the `if` and the `else`.

```
name = input("What is your name?")
my_number = input("Hello "+name+" please pick a number that's bigger than 100")
my_number = int(my_number)
print("Your number is "+str(my_number))

if(my_number > 100):
    print("That's a big number!")
elif(my_number > 90):
    print("Almost there!")
else:
    print("That number is too small!")
```

1 You can ask the user to pick a number now, check if it's the right size and, if it isn't, tell them it's not. What if you wanted it to keep going until you got an answer that was the right size? You could write `if` statements inside `if` statements, but what if the user still doesn't give you the right size of number?

You need a way to ask the question over and over until you get the right kind of answer. The way to do this in computer programming is called a **loop**. You're going to use one called the `while` loop.

2 A `while` loop is a bit like an `if` statement: it has code inside it that only runs if the condition in brackets is true. The difference is that a `while` loop runs over and over, until its condition is false. You have to make sure that there is always a way out of your `while` loops, or they'll run forever! It looks like this:

```
while(my_number < 100):
    my_number = input("Hello "+name+" please pick a number that's bigger than 100")
    my_number = int(my_number)
```

3 Now add a `while` loop to your program.

```
name = input("What is your name?")
my_number = 0

# Loop as long as "my_number" is less than 100
while(my_number < 100):
    # Ask the user for a number
    my_number = input("Hello "+name+" please pick a number that's bigger than 100")
    # Convert the user's answer from a string to an integer
    my_number = int(my_number)
    print("Your number is "+str(my_number))
    # Check if the number is bigger than 100
    if(my_number > 100):
        print("That's a big number!").
    elif(my_number > 90):
        print("Almost there! Try again!")
    else:
        print("That number is too small! Please try again!")
    # If my_number is smaller than 100 at this point, loop again
```

Comments

These are notes for programmers (or you later) that the computer will ignore. In Python, they start with # and last to the end of the line.

CoderDojo

Let's make a game!

I'm learning: Python

1 So now you've learned about:

- `print` statements: Talking to our user
- variables: A way to get our program to remember and update values
- strings: Pieces of text
- `input`: How to get information from our user
- maths: How to do maths with a number
- integers: Numbers for doing maths with
- `if` statements: Do something based on a condition
- `while` loops: Keep doing something until a condition isn't true

2 Try using these to make this game:

- There is a number (an integer), between 1 and 9, that the program picks in secret
- The player has 5 guesses to pick the number
- The game teaches the player the rules
- The player is told after each guess whether the number is lower, higher, or right, and how many guesses they have left
- If the player gets their guess right, they get a special winning message
- If the player gets their 5th guess wrong, the game is over and they lose

3 You can play an example of the game at dojo.soy/py-dice.

4 You're missing just one thing to be able to write this game: A way to get a random number between 1 and 9. The code to do this is a little beyond what you've covered, so just treat this next bit as a piece of magic. It will be explained in later Sushi Cards. Put this as the **first line** in your program:

```
from random import randint as dice
```

Now, anywhere you want to use a random number between 1 and 9, just use *dice(1,9)*. For example:

```
secret_number = dice(1,9)
```

5 Try to make the game now! Remember to use previous cards. If you're stuck, or when you're done,.you can check my answer at dojo.soy/py-guess. Don't worry if yours looks very different, as long as it works. Good luck!

> ## What did you think?
>
> You've finished this series of Sushi Cards. I'd love to know what you thought of them. If you have a few minutes, please let me know at dojo.soy/py-beginner