INTRODUCTION TO PYTHON 1

Computer Practicum EN - WIS - LES

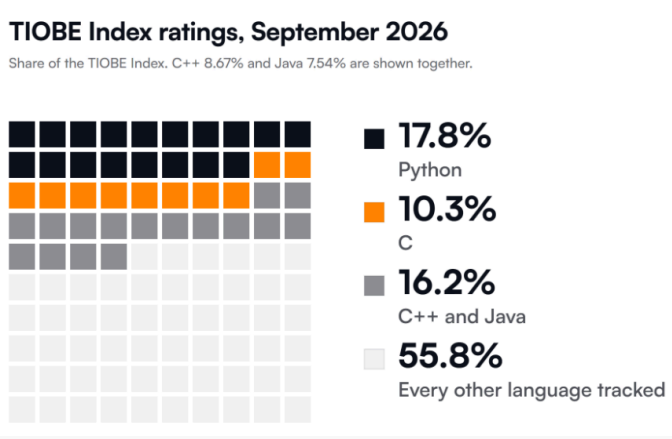
Lecturer: Niki Hrovatin

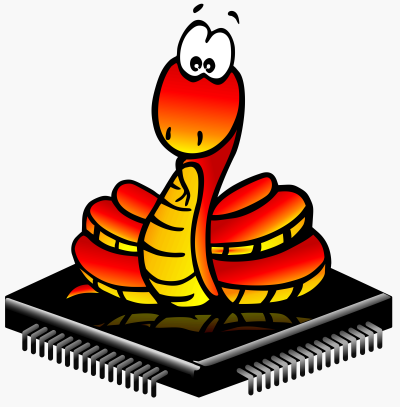
What is Python

Python is a high-level, general-purpose programming language known for its simple, readable syntax.

It is widely used for automation, scientific computing, and data analysis.

MicroPython is a lightweight implementation of Python designed to run on microcontrollers, allowing us to read sensors and control hardware.





What we will do?

- Learn essentials of programming in python by practicing with google colab
  - data types, variables and operators
  - conditional statement, loops, and lists
  - function calling
  - program management
- Implement the sensor project using micropython and Arduino

Google colab

Google Colab (Colaboratory) is a free, browser-based environment for writing and executing Python code.

- No installation required — run Python directly in your web browser.
- Interactive notebooks — combine executable code, text, and results.
- Cloud execution — code runs on Google's servers.
- Google Drive integration — save, access, and share notebooks.

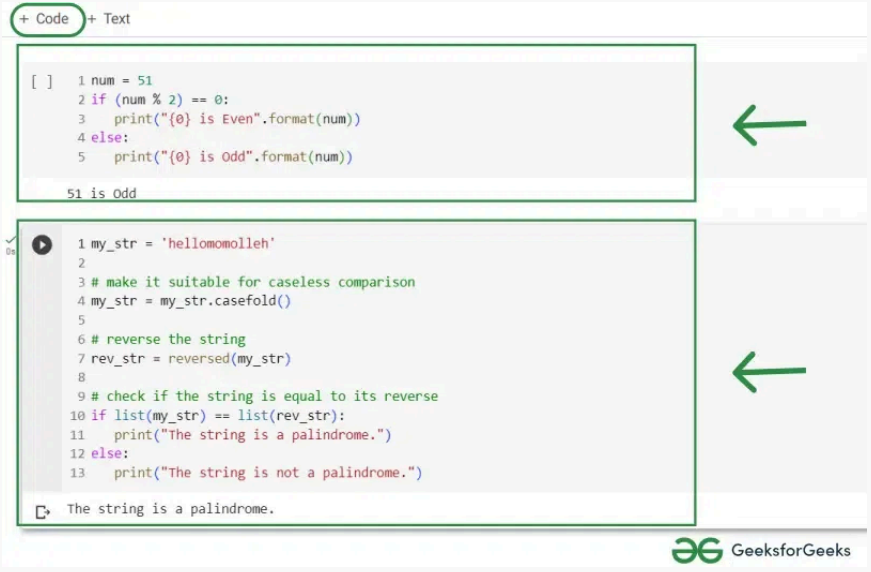


Access: https://colab.research.google.com/

Google colab - the editor

The editor is where you write your code.

- It also includes a number of features to help you write and debug your code, such as code completion, syntax highlighting and error messages.



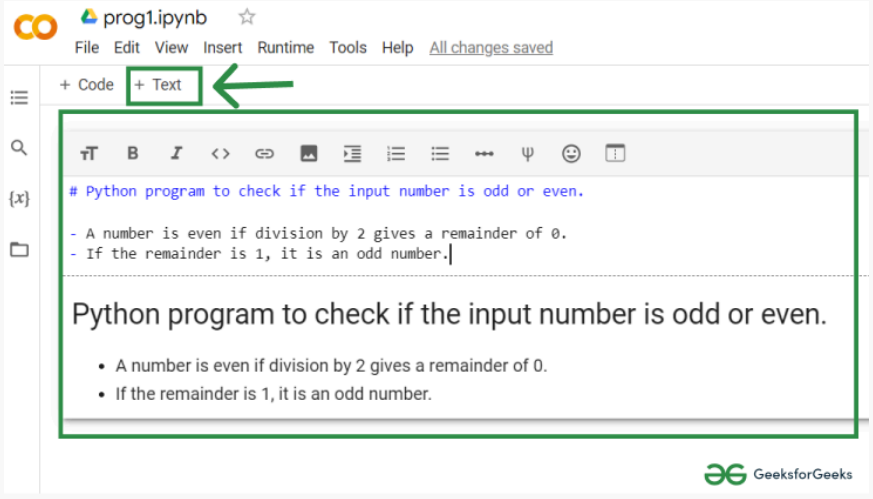
Code cells

- The area where you can able write and execute the program.
- Click the **Play icon** in the left gutter of the cell;
- Type **Cmd/Ctrl+Enter** to run the cell in place;
- Type **Shift+Enter** to run the cell and move focus to the next cell (adding one if none exists); or
- Type **Alt+Enter** to run the cell and insert a new code cell immediately below it.

Google colab - text cells

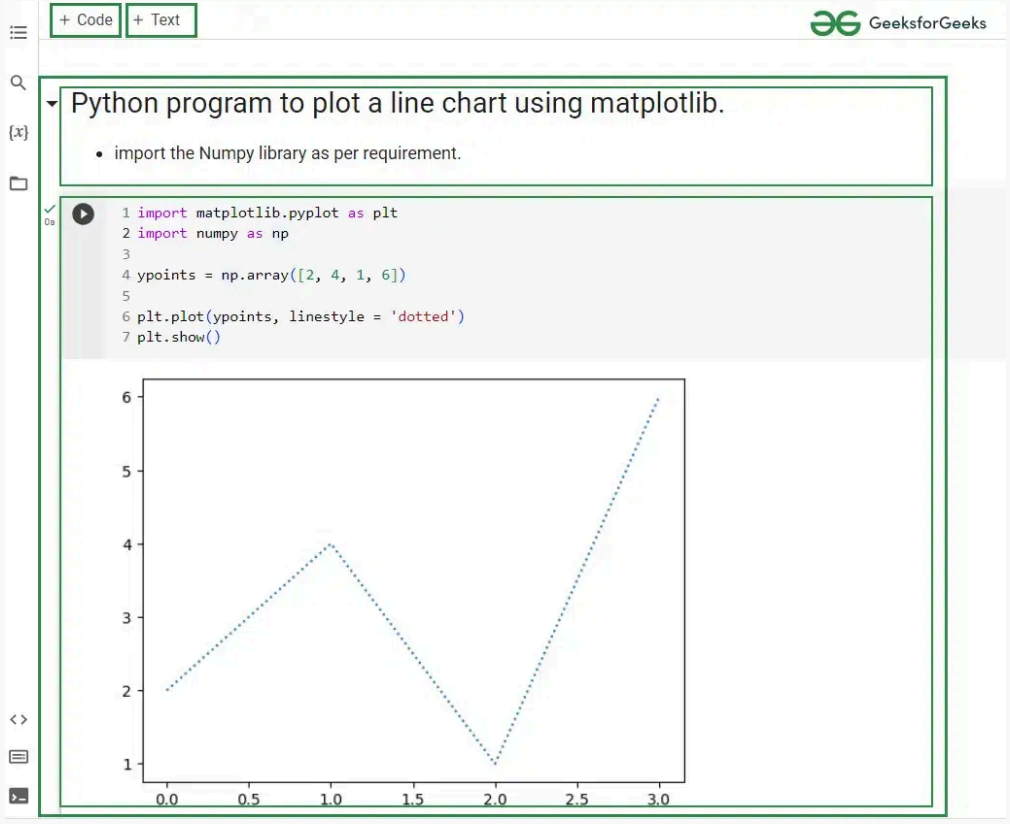
Text cells

- The area where you can able write any text.
- It supports **Markdown** syntax.
- You can also add math to text cells using **LaTeX** to be rendered by **MathJax**.



Google colab - add cells

Additionally you can add new cells by using the **+ CODE** and **+ TEXT** buttons that show when you hover between cells.



Google colab - Executing Code Cell

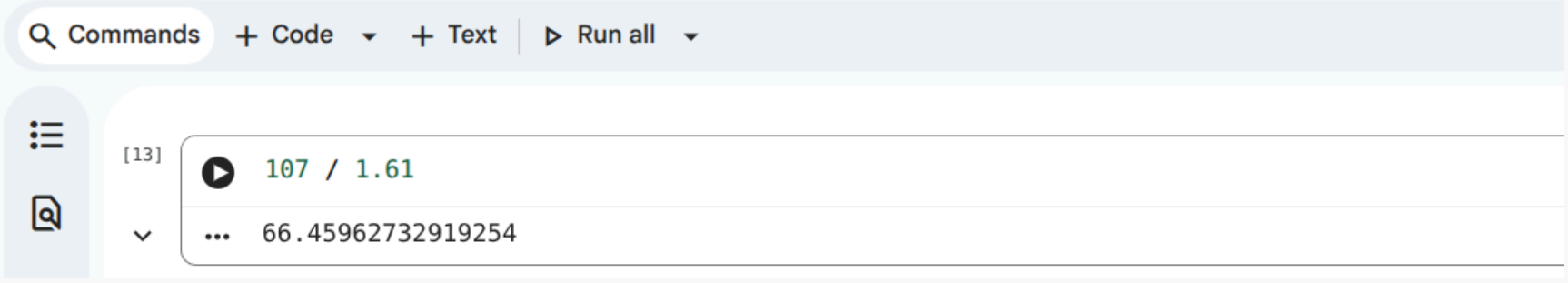
You can able to write your program within the code cell and can run the code to check your desired output.



Python as a calculator

Let us calculate the distance between Koper and Ljubljana in miles

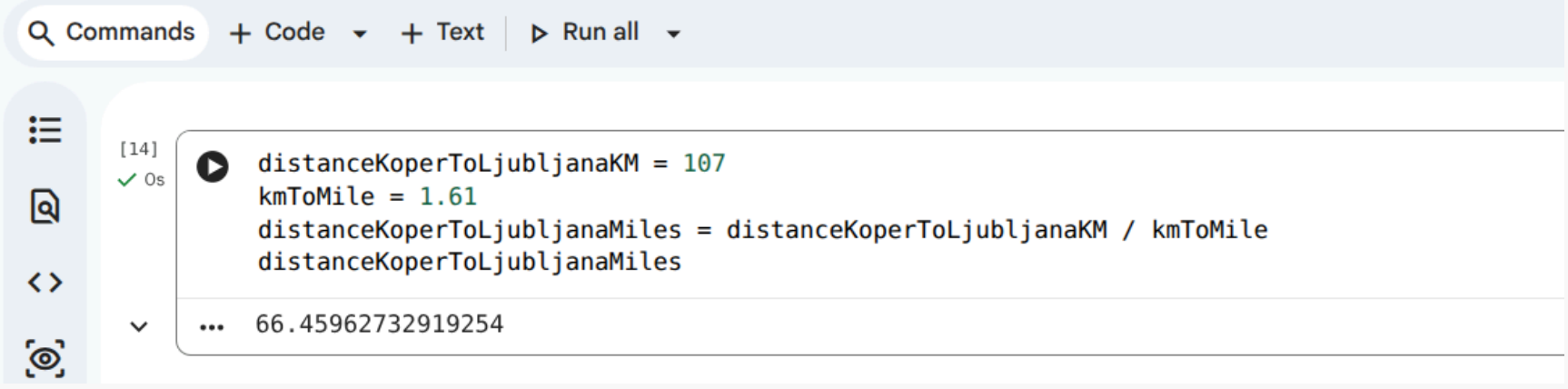
- koper - LJ = 107km



Variables

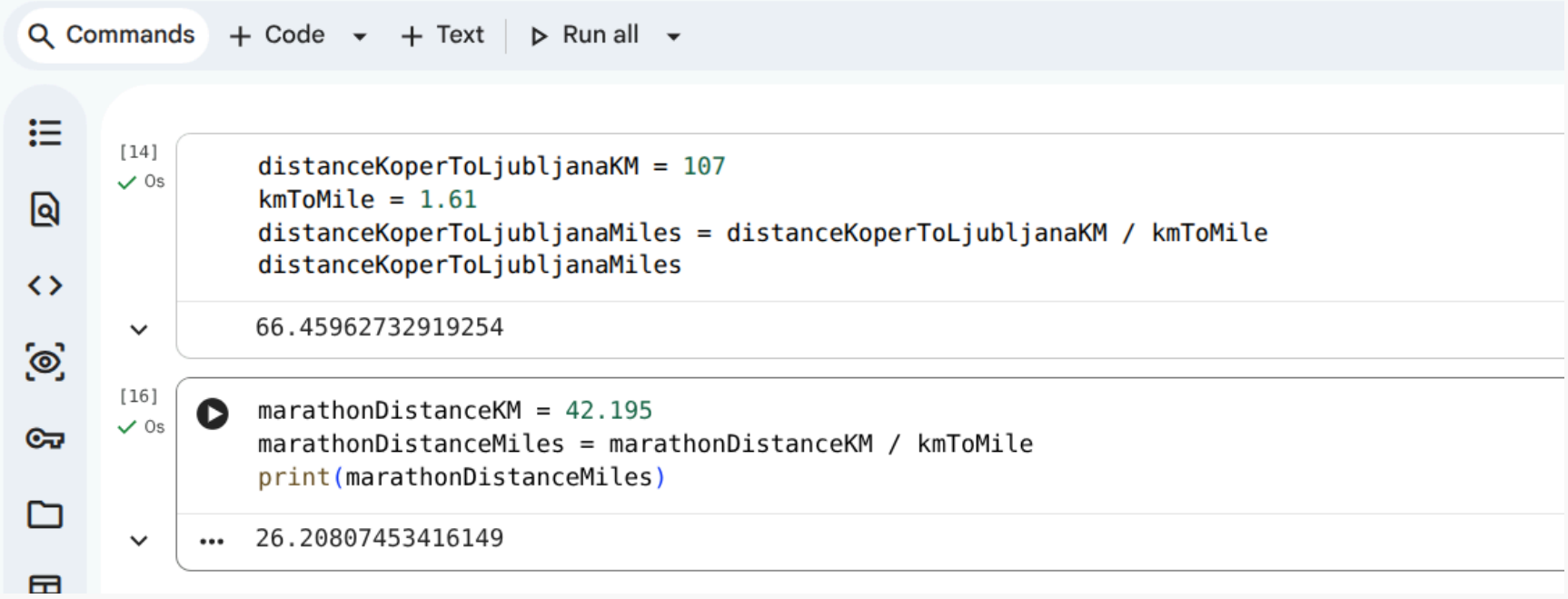
Great calculator but how can we make it store values?

- Do this by defining variables
- Can later be called by the variable name
- Variable names are case sensitive and unique



Variables

We can now reuse the variable **kmToMile** in the next block without having to define it again!



Types

Variables actually have a type, which defines the way it is stored.

The basic types are:

Type	Declaration	Example	Usage
Integer	int	`x = 124`	Numbers without decimal point
Float	float	`x = 124.56`	Numbers with decimcal point
String	str	`x = "Hello world"`	Used for text
Boolean	bool	`x = True` or `x = False`	Used for conditional statements
NoneType	None	`x = None`	Whenever you want an empty variable

Why to care about the type



Failure!!!

- Coding is all about trial and error.
- Don't be afraid of it.
- Error messages aren't scary, they are useful.

Why to care about the type



Important lesson to remember! We can't do arithmetic operations on variables of different types. Therefore make sure that you are always aware of your variables types!

You can find the type of a variable using type(). For example type type(x).

Why to care about the type

[17] 0s

```
x = 10
y = "20"
x + y
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call las
/tmp/ipykernel_1578/1207366883.py in <cell line: 0>()
      1 x = 10
      2 y = "20"
----> 3 x + y

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Next steps: Explain error

[20] 0s

```
print(type(x))
print(type(y))
```

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

Casting types

Luckily Python offers us a way of converting variables to different types!
Casting – the operation of converting a variable to a different type



Similar methods exist for other data types: **int(), float(), str()**

Quiz!

What will be the result?



Quiz!

What will be the result?



result: `1020`

Arithmetic operations

Similar to actual Mathematics.

Order of precedence is the same as in Mathematics.

We can also use parenthesis ()

Symbol	Task Performed	Example	Result
+	Addition	4 + 3	7
-	Subtraction	4 - 3	1
/	Division	7 / 2	3.5
%	Mod	7 % 2	1
*	Multiplication	4 * 3	12
//	Floor division	7 // 2	3
**	Power of	7 ** 2	49

Order precedence example

```
16 ** 2 / 4
```

```
64.0
```

What about this? -- type it!

```
[ ]  5 + 20 * 2 / 8 + 1
```

Exercise

Write a code cell that:

1. Stores the temperature in a variable called `temperature_c`.
2. Converts the temperature to Fahrenheit using the formula:
   - $F = C \times \frac{9}{5} + 32$
3. Prints the result.

Exercise

A sensor records three temperature measurements:

- 22.4
- 23.1
- 23.8

Write a code cell that:

1. Stores each measurement in a separate variable
2. Calculates the average temperature.
3. Prints the calculated average.

Comparison operators

- Return Boolean values
  - (e.g. True or False)
- Used extensively for conditional statements

Operator	Output
x == y	True if x and y have the same value
x != y	True if x and y don't have the same value
x < y	True if x is less than y
x > y	True if x is more than y
x <= y	True if x is less than or equal to y
x >= y	True if x is more than or equal to y

Comparison operators

```
[24]
✓ 0s
x = 5     # assign the value 5 to the variable x
x == 5    # check if value of x is 5

...  True
```

Very important!!!: note that == is not the same as =

- == is checking equality
- = is assigning a value to a variable

By the way, we use # to give comments in code

- Comments is additional text that is not executed as computer code

```
[25]
✓ 0s
x > 7

...  False
```

Logical operators

Allows us to extend the conditional logic

- Will be used later on

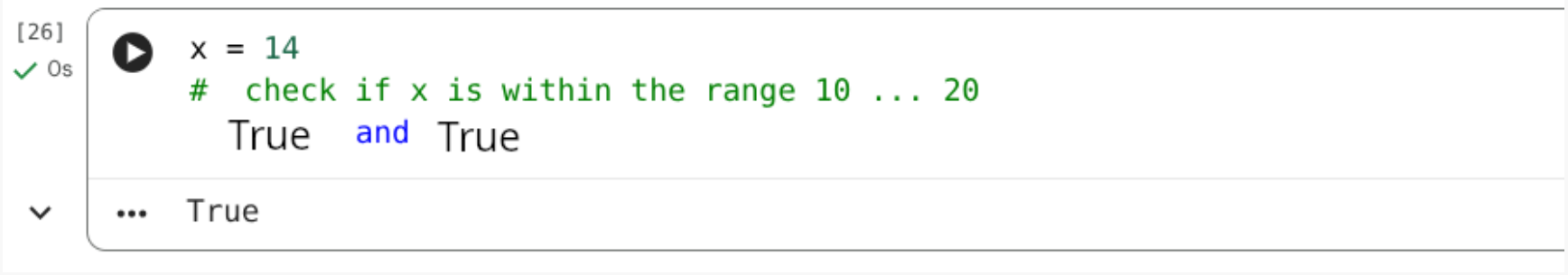
Operation	Result
x or y	True if at least on is True
x and y	True only if both are True
not x	True only if x is False

a	not a
False	True
True	False

a	b	a and b	a or b
False	False	False	False
False	True	False	True
True	False	False	True
True	True	True	True

Truth-table definitions of `bool` *operations*

Logical operators



Logical operators

[26]
✓ 0s

```
x = 14
#  check if x is within the range 10 ... 20
  True  and  True
```

```
True
```

[26]
✓ 0s

```
x = 14
#  check if x is within the range 10 ... 20
(x > 10) and (x < 20)
```

```
True
```

Exercise - Temperature Warning

A temperature sensor measures 35°C.

Write a Python program that:

1. Stores the temperature in a variable.
2. Checks whether the temperature is below 10°C or above 30°C.
3. Prints `True` if the temperature is outside this range, otherwise `False`.

Exercise - Temperature Warning

A sensor measures the following environmental conditions:

- Temperature: 24°C
- Relative humidity: 55%

Write a Python program that:

1. Stores both measurements in separate variables.
2. Checks whether both conditions are satisfied:
    - Temperature is between 20°C and 26°C (inclusive).
    - Humidity is between 30% and 60% (inclusive).
3. Prints `True` only if both conditions are satisfied, otherwise `False`.

Strings

For using text in programs!

- use the quote character to declare them "I'm a String"

Can be concatenated!!



Mixing up strings and numbers

Often we would need to mix up numbers and strings. It is best to keep numbers as numbers (i.e. int or float) and cast them to strings whenever we need them as a string.

- just use str(variable-name)

```
x = 6
x = ( x * 5345 ) // 63
print("The answer to life, the universe and everything is "+ str(x))
```

```
The answer to life, the universe and everything is 42
```

Printing

When writing Arduino scripts, your outcomes aren't printed on the terminal.

- Thus, you must print them yourself with the print() function.
- Beware to not mix up the different type of variables!

```
y = "Python is powerful"
z = " and verstaile!"
print(y + z + " And the answer to life is: " + str(x))
```

```
Python is powerful and verstaile! And the answer to life is: 42
```

Quiz!

Do you see anything wrong with this block?

```
str1 = "Choose "
str2 = 5
str3 = " cards"
print(str1 + str2 + str3)
```

Quiz!

Do you see anything wrong with this block?

```
str1 = "Choose "
str2 = 5
str3 = " cards"
print(str1 + str2 + str3)
```

mixing types!

Quiz!

Do you see anything wrong with this block?

```
str1 = "Choose "
str2 = 5
str3 = " cards"
print(str1 + str2 + str3)
```

mixing types!

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipykernel_1578/396908741.py in <cell line: 0>()
      2 str2 = 5
      3 str3 = " cards"
----> 4 print(str1 + str2 + str3)

TypeError: can only concatenate str (not "int") to str
```

Next steps: Explain error

Printing

Success!!!



If we comma separate statements in a print function we can have different variables printing!