The code was originally written in a Python file and copied over to Typst.

```
print("Hello, world!", 12)

# this is a comment

# multiline string, also usable as comments, also called "doc strings"
a = """
alsdmasldas
aksdmksamd
"""
print(a)

# data types & arithmatics
a = 12
b = 42.123132123
# arithmatics operations include:
# + - * / -- pretty self-explanatory
# // -- rounds down the value
# ** -- exponentiation
print(b // a)

# accepting input & type casting
print(int(input("Enter your age: ")))
# the above code will fail if the input is invalid, for example "abc" or "17.1" will
result in an error

# conditions
# conditions evaluates into two values, True and False
# some comparators include: > (greater than), < (less than), ≥ (greater or equal), ≤
(less or equal)
# there are also compound operators which are keywords including: and, or
# besides that there are also the "not" keyword, which inverts the condition
if a ≥ 12: # True
    print("a is more than or equal to 12")

# Conditions can also be used outside of if statements, the following will print True
print(a ≥ 12)

# Some other conditions example include:
# a < 12 and b > 40 -- False, because a is *not* less than 12
# a < 12 or b > 40 -- True, because b is more than 40

# Syntactic sugar: Extra assignment operators
# You can use +=, -=, /=, *=, //=, **= as a shortcut when assigning variables
# The following:
a += 12
# is equivalent to doing "a = a + 12"
print(a)

# lists
x = [12, 23, 34]
print(x[0])
#       ^
# the 0 here is called the index

# You can use the len() function to check the length of the list
# (Note: the len() function can also be used to check the length of strings)
print(len(x))
```

```python
# Adds to the end of the list
x.append(45)
print(x)

# Delete from the list
del x[1]
# or alternatively: x.pop(1)
print(x)

# To remove an item by specifying its *value* (the above uses the index)
# x.remove(value)

# The shortcut assignment operators also work here
x[1] += 69
print(x)

# Sorting the list (from less to more)
x.sort()
print(x)

# Reversing the list
x.reverse()
print(x)

# Looping through the array using the while loop
i = 0 # i is for index
while i < len(x): # while i is less than length of the list x:
    print(x[i])
    i += 1 # Don't forget to increase the index, or else it will cause an infinite loop

# dictionaries
d = {
    "abc": 123,
}

print(d["abc"]) # accessing
d["def"] = 234 # assigning
print(d["abc"])
del d["abc"] # deleting
print(d)

# A function in programming is like a mathematical function:
# f(x) = x + 1
# will be equal to:
def f(x):
    return x + 1

# def f(x):
# define a function called "f" which accepts the parameter "x"
# the function will _return_ the value, which can be used for further calculations
# or be simply displayed

print(f(12))
#       ^^
# This is called an _argument_, arguments are the value passed in as a parameter to a
function
# so you could say "this line prints the result of the function f with 12 passed in as
the argument"
```

Extras

Return values

This section explains *why* you may want to return a value

If you have a mathematical expression as follows:

$f(x) = x + 1$

$x = 12$

$y = f(x) \times 2$

To find y, you would need to evaluate $f(x)$ (or basically $f(12)$), which will be 13, becoming $y = 13 \times 2$, 13 *is* the return value of $f(12)$

Functions return values that can be used for further processing, but function also doesn't have to return values.