Coding in Python

Python Basics - Part 1

IDPO 2910 Group 5

April 2024

What is Python?

Did you know? Python was made by someone who was bored.
It's a language designed to be almost as understandable as English.
You will be using Python 3. Why?

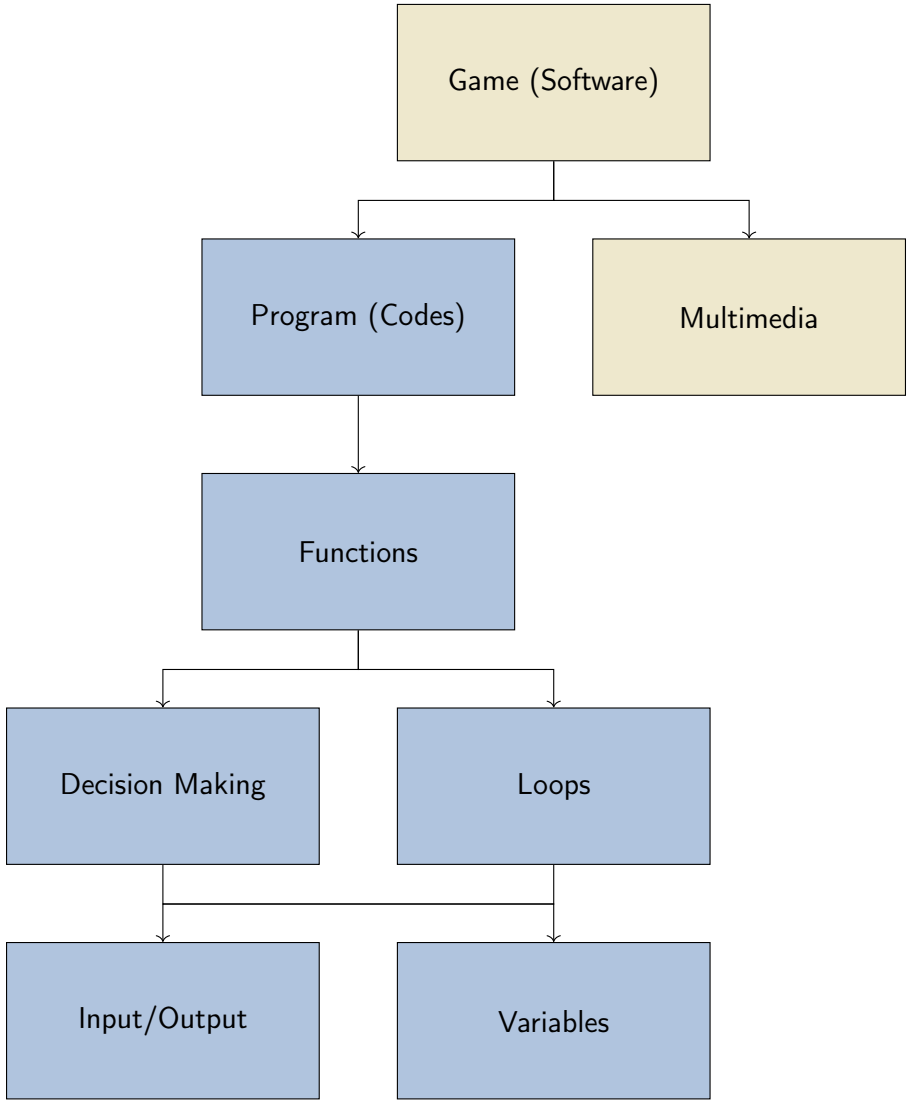
This is the logo of Python.

What is Python?

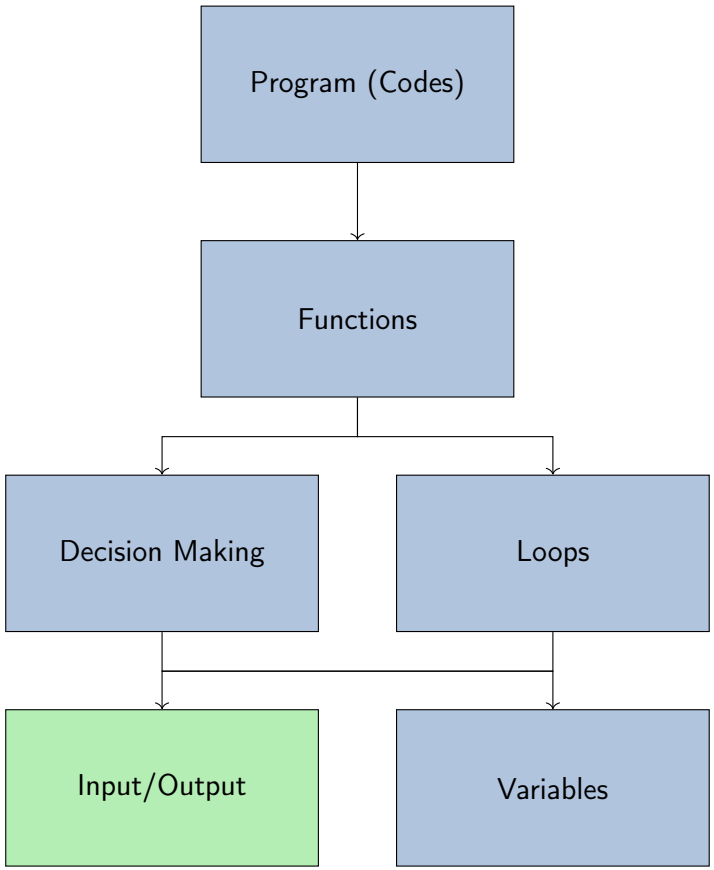
Did you know? Python was made by someone who was bored.
It's a language designed to be almost as understandable as English.
You will be using Python 3. Why? Because Python 1 are 2 are dead.

This is the logo of Python.

World of Game Coding



Contents


Program (Codes)
Functions
Decision Making
Loops
Input/Output
Variables

The first thing in Python - `print()` function

```
print("This is the print function.")
```

The first thing in Python - `print()` function

`print()` is a function that lets you print something, also known as text output.

```
print("Word") # This prints the word "Word".
```

Output:

```
>>> print("Word")
Word
>>> print("Haha hehe")
Haha hehe
```

input() function

We know how to output strings, what about input?

```
input("This is the input function.")
```

The first thing in Python - `print()` function

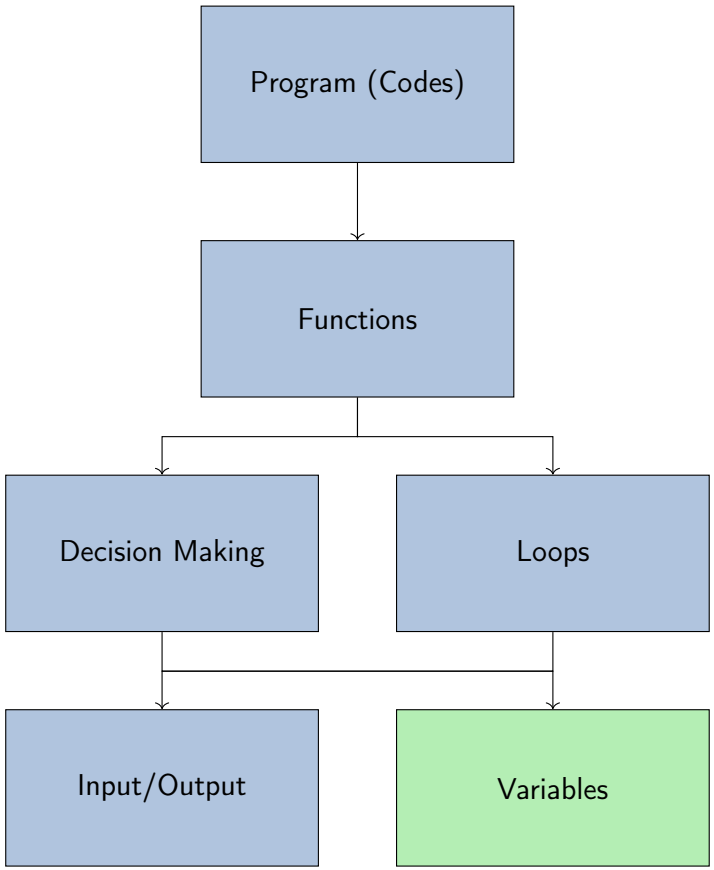
`input()` is a function that outputs a prompt and lets the user enter something.

```
>>> input("Enter a number: ")
Enter a number: 5
```

Simply inputting doesn't do anything, but we can print it.

```
>>> print(input("Enter a number: "))
Enter a number: 100
100
```

Contents


Program (Codes)
Functions
Decision Making
Loops
Input/Output
Variables

Variables

Imagine you borrow a box from the computer.

Variables

Imagine you borrow a box from the computer.

Give it a name and a value, you can now recall this value with the name!

Variables

The code usually goes:
`variable_name = data`
This means whatever `data` is, it is now stored in a variable with name `variable_name`.

Some basic variable types:

```
a = 5
```

Variables

The code usually goes:
`variable_name = data`
This means whatever `data` is, it is now stored in a variable with name `variable_name`.

Some basic variable types:

```
a = 5          # This is an integer (int) stored in a
```

Variables

The code usually goes:
`variable_name = data`
This means whatever `data` is, it is now stored in a variable with name `variable_name`.

Some basic variable types:

```
a = 5          # This is an integer (int) stored in a
b = True
```

Variables

The code usually goes:
`variable_name = data`
This means whatever `data` is, it is now stored in a variable with name `variable_name`.

Some basic variable types:

```
a = 5          # This is an integer (int) stored in a
b = True       # This is a boolean (bool) stored in b
```

Variables

The code usually goes:
`variable_name = data`
This means whatever `data` is, it is now stored in a variable with name `variable_name`.

Some basic variable types:

```
a = 5          # This is an integer (int) stored in a
b = True       # This is a boolean (bool) stored in b
c = 3.2
```

Variables

The code usually goes:
`variable_name = data`
This means whatever `data` is, it is now stored in a variable with name `variable_name`.

Some basic variable types:

```
a = 5        # This is an integer (int) stored in a
b = True     # This is a boolean (bool) stored in b
c = 3.2      # This is a float (float)  stored in c
```

Variables

The code usually goes:
`variable_name = data`
This means whatever `data` is, it is now stored in a variable with name `variable_name`.

Some basic variable types:

```
a = 5        # This is an integer (int) stored in a
b = True     # This is a boolean (bool) stored in b
c = 3.2      # This is a float (float)  stored in c
d = "abc"
```

Variables

The code usually goes:
`variable_name = data`
This means whatever `data` is, it is now stored in a variable with name `variable_name`.

Some basic variable types:

```
a = 5       # This is an integer (int) stored in a
b = True    # This is a boolean (bool) stored in b
c = 3.2     # This is a float (float)  stored in c
d = "abc"   # This is a string (str)   stored in d
```

Variables

The code usually goes:
`variable_name = data`
This means whatever `data` is, it is now stored in a variable with name `variable_name`.

Some basic variable types:

```
a = 5        # This is an integer (int) stored in a
b = True     # This is a boolean (bool) stored in b
c = 3.2      # This is a float (float)  stored in c
d = "abc"    # This is a string (str)   stored in d
e = 'abc'
```

Variables

The code usually goes:
`variable_name = data`
This means whatever `data` is, it is now stored in a variable with name `variable_name`.

Some basic variable types:

```
a = 5       # This is an integer (int) stored in a
b = True    # This is a boolean (bool) stored in b
c = 3.2     # This is a float (float)  stored in c
d = "abc"   # This is a string (str)   stored in d
e = 'abc'   # This is also a string    stored in e
```

Variables - Integers

What are integers?

Variables - Integers

What are integers?
Integers are just like what you've learnt in Maths, numbers without decimal points.

Variables - Integers

What are integers?
Integers are just like what you've learnt in Maths, numbers without decimal points.

```
a = 5
```

Variables - Integers

What are integers?
Integers are just like what you've learnt in Maths, numbers without decimal points.

```
a = 5          # Valid
```

Variables - Integers

What are integers?
Integers are just like what you've learnt in Maths, numbers without decimal points.

```
a = 5          # Valid
b = 12
```

Variables - Integers

What are integers?
Integers are just like what you've learnt in Maths, numbers without decimal points.

```
a = 5          # Valid
b = 12         # Valid
```

Variables - Integers

What are integers?
Integers are just like what you've learnt in Maths, numbers without decimal points.

```
a = 5          # Valid
b = 12         # Valid
c = 69420
```

Variables - Integers

What are integers?
Integers are just like what you've learnt in Maths, numbers without decimal points.

```
a = 5        # Valid
b = 12       # Valid
c = 69420    # Valid
```

Variables - Integers

What are integers?
Integers are just like what you've learnt in Maths, numbers without decimal points.

```
a = 5          # Valid
b = 12         # Valid
c = 69420      # Valid
d = -1984
```

Variables - Integers

What are integers?
Integers are just like what you've learnt in Maths, numbers without decimal points.

```
a = 5        # Valid
b = 12       # Valid
c = 69420    # Valid
d = -1984    # Valid
```

Variables - Integers

What are integers?
Integers are just like what you've learnt in Maths, numbers without decimal points.

```
a = 5          # Valid
b = 12         # Valid
c = 69420      # Valid
d = -1984      # Valid
e = 32.5
```

Variables - Integers

What are integers?
Integers are just like what you've learnt in Maths, numbers without decimal points.

```
a = 5          # Valid
b = 12         # Valid
c = 69420      # Valid
d = -1984      # Valid
e = 32.5       # This would become a float instead
```

Variables - Integers

What are integers?
Integers are just like what you've learnt in Maths, numbers without decimal points.

```
a = 5          # Valid
b = 12         # Valid
c = 69420      # Valid
d = -1984      # Valid
e = 32.5       # This would become a float instead
f = '5'
```

Variables - Integers

What are integers?
Integers are just like what you've learnt in Maths, numbers without decimal points.

```
a = 5          # Valid
b = 12         # Valid
c = 69420      # Valid
d = -1984      # Valid
e = 32.5       # This would become a float instead
f = '5'        # This would become a string instead
```

Variables - Integer Arithmetic Operations

You can do normal operations on integers:

```
a = 1 + 2
```

Variables - Integer Arithmetic Operations

You can do normal operations on integers:

```
a = 1 + 2    # a stores the integer 3
```

Variables - Integer Arithmetic Operations

You can do normal operations on integers:

```
a = 1 + 2    # a stores the integer 3
b = 80 - 52
```

Variables - Integer Arithmetic Operations

You can do normal operations on integers:

```
a = 1 + 2    # a stores the integer 3
b = 80 - 52 # b stores the integer 28
```

Variables - Integer Arithmetic Operations

You can do normal operations on integers:

```
a = 1 + 2   # a stores the integer 3
b = 80 - 52 # b stores the integer 28
c = 69 * -2
```

Variables - Integer Arithmetic Operations

You can do normal operations on integers:

```
a = 1 + 2   # a stores the integer 3
b = 80 - 52 # b stores the integer 28
c = 69 * -2 # c stores the integer -138
```

Variables - Integer Arithmetic Operations

You can do normal operations on integers:

```
a = 1 + 2   # a stores the integer 3
b = 80 - 52 # b stores the integer 28
c = 69 * -2 # c stores the integer -138
d = 6 / 4
```

Variables - Integer Arithmetic Operations

You can do normal operations on integers:

```
a = 1 + 2    # a stores the integer 3
b = 80 - 52 # b stores the integer 28
c = 69 * -2 # c stores the integer -138
d = 6 / 4    # d stores the float 1.5
```

Variables - Integer Arithmetic Operations

You can do normal operations on integers:

```
a = 1 + 2   # a stores the integer 3
b = 80 - 52 # b stores the integer 28
c = 69 * -2 # c stores the integer -138
d = 6 / 4   # d stores the float 1.5
e = 18 / 2
```

Variables - Integer Arithmetic Operations

You can do normal operations on integers:

```
a = 1 + 2   # a stores the integer 3
b = 80 - 52 # b stores the integer 28
c = 69 * -2 # c stores the integer -138
d = 6 / 4   # d stores the float 1.5
e = 18 / 2  # e stores the float 9.0
```

Variables - Integer Arithmetic Operations

You can do normal operations on integers:

```
a = 1 + 2   # a stores the integer 3
b = 80 - 52 # b stores the integer 28
c = 69 * -2 # c stores the integer -138
d = 6 / 4   # d stores the float 1.5
e = 18 / 2  # e stores the float 9.0
```

Division

Whether a number can be precisely divided or not, division returns a `float`.

Variables - Integer Arithmetic Operations

Operations with variables:

```
a = 100
b = 12
```

Variables - Integer Arithmetic Operations

Operations with variables:

```
a = 100
b = 12
c = a + b
```

Variables - Integer Arithmetic Operations

Operations with variables:

```
a = 100
b = 12
c = a + b    # c stores the integer 112
```

Variables - Integer Arithmetic Operations

Operations with variables:

```
a = 100
b = 12
c = a + b    # c stores the integer 112
d = b - a
```

Variables - Integer Arithmetic Operations

Operations with variables:

```
a = 100
b = 12
c = a + b    # c stores the integer 112
d = b - a    # d stores the integer -88
```

Variables - Integer Arithmetic Operations

Operations with variables:

```
a = 100
b = 12
c = a + b    # c stores the integer 112
d = b - a    # d stores the integer -88
e = a * -b
```

Variables - Integer Arithmetic Operations

Operations with variables:

```
a = 100
b = 12
c = a + b    # c stores the integer 112
d = b - a    # d stores the integer -88
e = a * -b   # e stores the integer -1200
```

Variables - Integer Arithmetic Operations

Operations with variables:

```
a = 100
b = 12
c = a + b    # c stores the integer 112
d = b - a    # d stores the integer -88
e = a * -b   # e stores the integer -1200
f = a / b
```

Variables - Integer Arithmetic Operations

Operations with variables:

```
a = 100
b = 12
c = a + b    # c stores the integer 112
d = b - a    # d stores the integer -88
e = a * -b   # e stores the integer -1200
f = a / b    # f stores the float 8.333333333333334
```

Variables - Integer Arithmetic Operations

Then how do we get an integer output?

Variables - Integer Arithmetic Operations

Then how do we get an integer output?

```
a = 100
b = 12
```

Variables - Integer Arithmetic Operations

Then how do we get an integer output?

```
a = 100
b = 12
c = a // b
```

Variables - Integer Arithmetic Operations

Then how do we get an integer output?

```
a = 100
b = 12
c = a // b  # c stores the integer 8
```

Variables - Integer Arithmetic Operations

Then how do we get an integer output?

```
a = 100
b = 12
c = a // b  # c stores the integer 8
            # // operator takes the closest and smaller
            # integer from the division operation
```

Variables - Integer Arithmetic Operations

Then how do we get an integer output?

```
a = 100
b = 12
c = a // b  # c stores the integer 8
            # // operator takes the closest and smaller
            # integer from the division operation
d = a % b
```

Variables - Integer Arithmetic Operations

Then how do we get an integer output?

```
a = 100
b = 12
c = a // b  # c stores the integer 8
            # // operator takes the closest and smaller
            # integer from the division operation
d = a % b   # d stores the integer 4
```

Variables - Integer Arithmetic Operations

Then how do we get an integer output?

```
a = 100
b = 12
c = a // b  # c stores the integer 8
            # // operator takes the closest and smaller
            # integer from the division operation
d = a % b   # d stores the integer 4
            # % operator takes the remainder of a
            # division operation
```

Variables - Integer Arithmetic Operations

Also, the power (exponent) operation:

```
a = 2
b = 5
```

Variables - Integer Arithmetic Operations

Also, the power (exponent) operation:

```
a = 2
b = 5
c = a ** b
```

Variables - Integer Arithmetic Operations

Also, the power (exponent) operation:

```
a = 2
b = 5
c = a ** b  # c stores the integer 32
            # ** operator means power
```

Variables - Floats

What are floats?

Variables - Floats

What are floats?
Floats are numbers with decimal point(s).

Variables - Floats

What are floats?
Floats are numbers with decimal point(s).
Arithmetic operators we learnt can be applied as well.

```
a = 0.2
```

Variables - Floats

What are floats?
Floats are numbers with decimal point(s).
Arithmetic operators we learnt can be applied as well.

```
a = 0.2      # a stores the float 0.2
```

Variables - Floats

What are floats?
Floats are numbers with decimal point(s).
Arithmetic operators we learnt can be applied as well.

```
a = 0.2      # a stores the float 0.2
b = 3.0
```

Variables - Floats

What are floats?
Floats are numbers with decimal point(s).
Arithmetic operators we learnt can be applied as well.

```
a = 0.2      # a stores the float 0.2
b = 3.0      # b stores the float 3.0
```

Variables - Floats

What are floats?
Floats are numbers with decimal point(s).
Arithmetic operators we learnt can be applied as well.

```
a = 0.2      # a stores the float 0.2
b = 3.0      # b stores the float 3.0
c = a + b
```

Variables - Floats

What are floats?
Floats are numbers with decimal point(s).
Arithmetic operators we learnt can be applied as well.

```
a = 0.2      # a stores the float 0.2
b = 3.0      # b stores the float 3.0
c = a + b    # c stores the float 3.2
```

Variables - Floats

What are floats?
Floats are numbers with decimal point(s).
Arithmetic operators we learnt can be applied as well.

```
a = 0.2      # a stores the float 0.2
b = 3.0      # b stores the float 3.0
c = a + b    # c stores the float 3.2
d = b / a
```

Variables - Floats

What are floats?
Floats are numbers with decimal point(s).
Arithmetic operators we learnt can be applied as well.

```
a = 0.2      # a stores the float 0.2
b = 3.0      # b stores the float 3.0
c = a + b    # c stores the float 3.2
d = b / a    # d stores the float 15.0
```

Variables - Floats

What are floats?
Floats are numbers with decimal point(s).
Arithmetic operators we learnt can be applied as well.

```
a = 0.2      # a stores the float 0.2
b = 3.0      # b stores the float 3.0
c = a + b    # c stores the float 3.2
d = b / a    # d stores the float 15.0
e = a ** b
```

Variables - Floats

What are floats?
Floats are numbers with decimal point(s).
Arithmetic operators we learnt can be applied as well.

```
a = 0.2      # a stores the float 0.2
b = 3.0      # b stores the float 3.0
c = a + b    # c stores the float 3.2
d = b / a    # d stores the float 15.0
e = a ** b   # e stores the float 0.008000000000000002
```

Variables - Floats

What are floats?
Floats are numbers with decimal point(s).
Arithmetic operators we learnt can be applied as well.

```
a = 0.2      # a stores the float 0.2
b = 3.0      # b stores the float 3.0
c = a + b    # c stores the float 3.2
d = b / a    # d stores the float 15.0
e = a ** b   # e stores the float 0.008000000000000002
```

Inaccuracies

Inaccuracies happen with decimal division. Be careful when dealing with floats.

Variables - Floats

What are floats?
Floats are numbers with decimal point(s).
We learnt about arithmetic operators: +, -, *, /, //, %, **
All of them, except // and % can be applied to floats.

```
a = 0.2      # a stores the float 0.2
b = 3        # b stores the integer 3
```

Variables - Floats

What are floats?
Floats are numbers with decimal point(s).
We learnt about arithmetic operators: +, -, *, /, //, %, **
All of them, except // and % can be applied to floats.

```
a = 0.2      # a stores the float 0.2
b = 3        # b stores the integer 3
c = a + b    # c stores the float 3.2
d = b / a    # d stores the float 15.0
e = a ** b   # e stores the float 0.008000000000000002
```

Variables - Floats

What are floats?
Floats are numbers with decimal point(s).
We learnt about arithmetic operators: +, -, *, /, //, %, **
All of them, except // and % can be applied to floats.

```
a = 0.2      # a stores the float 0.2
b = 3        # b stores the integer 3
c = a + b    # c stores the float 3.2
d = b / a    # d stores the float 15.0
e = a ** b   # e stores the float 0.008000000000000002
```

Arithmetic operations between `int` and `float`

Arithmetic operations between integers and floats converts the integer into a float first before operating.

Variables - Boolean values

What are boolean values?

Variables - Boolean values

What are boolean values?
There are only 2 boolean values in existence: `True` and `False`.

Variables - Boolean values

What are boolean values?

There are only 2 boolean values in existence: `True` and `False`.

```
a = True
b = False
```

Variables - Boolean values

What are boolean values?
There are only 2 boolean values in existence: `True` and `False`.

```
a = True
b = False
```

That's it.

Variables - Strings

What are strings?

Variables - Strings

What are strings?

```
a = "word"
```

Variables - Strings

What are strings?

```
a = "word"  # a stores the string "word"
```

Variables - Strings

What are strings?

```
a = "word"  # a stores the string "word"
b = 'word2'
```

Variables - Strings

What are strings?

```
a = "word"  # a stores the string "word"
b = 'word2' # b stores the string "word2"
```

Variables - Strings

What are strings?

```
a = "word"  # a stores the string "word"
b = 'word2' # b stores the string "word2"
c = '5.20'
```

Variables - Strings

What are strings?

```
a = "word"  # a stores the string "word"
b = 'word2' # b stores the string "word2"
c = '5.20'  # c stores the string "5.20"
```

Variables - Strings

What are strings?

```
a = "word"  # a stores the string "word"
b = 'word2' # b stores the string "word2"
c = '5.20'  # c stores the string "5.20"
d = 'abc"
```

Variables - Strings

What are strings?

```
a = "word"  # a stores the string "word"
b = 'word2' # b stores the string "word2"
c = '5.20'  # c stores the string "5.20"
d = 'abc"   # error
```

Variables - Strings

What are strings?

```
a = "word"  # a stores the string "word"
b = 'word2' # b stores the string "word2"
c = '5.20'  # c stores the string "5.20"
d = 'abc"   # error
```

Quotes

In Python you must use corresponding quotation marks for strings.

Variables - Strings

How do I put the symbols ' and " into a string?

Variables - Strings

How do I put the symbols ' and " into a string?
For ":

Variables - Strings

How do I put the symbols ' and " into a string?
For ":

```
a = "word\"" # a stores the string "word""
```

Variables - Strings

How do I put the symbols ' and " into a string?
For ":

```
a = "word\"" # a stores the string "word""
b = 'word"'  # b stores the same string as a
```

Variables - Strings

How do I put the symbols ' and " into a string?
For ":

```
a = "word\"" # a stores the string "word""
b = 'word"'  # b stores the same string as a
```

Same goes for single quotes ':

```
a = 'word\'' # a stores the string "word'"
b = "word'"  # b stores the same string as a
```

Variables - Strings

There are additional symbols in strings.

```
a = "word\n" # \n represents the newline character
b = "word\t" # \t represents the tab character
```

Variables - Strings

```
a = "haha"
b = "hehe"
c = a + b
```

Variables - Strings

```
a = "haha"
b = "hehe"
c = a + b     # c stores the string "hahahehe"
```

Variables - Strings

```
a = "haha"
b = "hehe"
c = a + b     # c stores the string "hahahehe"
```

Concatenation of strings

You can concatenate (add) strings together with the addition symbol.

Type conversion

You can convert between types with their type names in Python.

Data Type	Command
Integer	`int()`
Float	`float()`
String	`str()`
Boolean	`bool()`

Focus

We will focus on `int()` today as it will be needed in your game.

Conversion with `int()`

`int()` tries to convert a variable into an integer.

```
a = 10          # int
print(int(a))
```

Conversion with `int()`

`int()` tries to convert a variable into an integer.

```
a = 10          # int
print(int(a)) # 10
```

Conversion with `int()`

`int()` tries to convert a variable into an integer.

```
a = 10          # int
print(int(a)) # 10
                # Nothing occurs
```

Conversion with `int()`

`int()` tries to convert a variable into an integer.

```
a = 10          # int
print(int(a)) # 10
                # Nothing occurs

b = 3.7         # float
print(int(b))
```

Conversion with `int()`

`int()` tries to convert a variable into an integer.

```
a = 10          # int
print(int(a)) # 10
                # Nothing occurs

b = 3.7         # float
print(int(b)) # 3
```

Conversion with `int()`

`int()` tries to convert a variable into an integer.

```
a = 10          # int
print(int(a)) # 10
                # Nothing occurs

b = 3.7         # float
print(int(b)) # 3
                # Discards values to the right of
                # the decimal point
```

Conversion with `int()`

`int()` tries to convert a variable into an integer.

```
a = 10          # int
print(int(a)) # 10
                # Nothing occurs

b = 3.7         # float
print(int(b)) # 3
                # Discards values to the right of
                # the decimal point

c = True        # boolean
print(int(c))
```

Conversion with int()

int() tries to convert a variable into an integer.

```
a = 10          # int
print(int(a)) # 10
                # Nothing occurs

b = 3.7         # float
print(int(b)) # 3
                # Discards values to the right of
                # the decimal point

c = True        # boolean
print(int(c)) # 1
```

Conversion with int()

int() tries to convert a variable into an integer.

```
a = 10          # int
print(int(a)) # 10
                # Nothing occurs

b = 3.7         # float
print(int(b)) # 3
                # Discards values to the right of
                # the decimal point

c = True        # boolean
print(int(c)) # 1

d = False       # boolean
print(int(d))
```

Conversion with int()

int() tries to convert a variable into an integer.

```
a = 10          # int
print(int(a)) # 10
                # Nothing occurs

b = 3.7         # float
print(int(b)) # 3
                # Discards values to the right of
                # the decimal point

c = True        # boolean
print(int(c)) # 1

d = False       # boolean
print(int(d)) # 0
```

Conversion with int()

int() tries to convert a variable into an integer.

```
a = 10          # int
print(int(a)) # 10
                # Nothing occurs

b = 3.7         # float
print(int(b)) # 3
                # Discards values to the right of
                # the decimal point

c = True        # boolean
print(int(c)) # 1

d = False       # boolean
print(int(d)) # 0
                # For boolean: 0 if False, True otherwise
```

Conversion with int()

```
i = "123abc"  # string
print(int(i))
```

Conversion with int()

```
i = "123abc"  # string
print(int(i)) # Error
```

Conversion with int()

```
i = "123abc"  # string
print(int(i)) # Error

j = "123"     # string with ONLY numbers
print(int(j))
```

Conversion with int()

```
i = "123abc"  # string
print(int(i)) # Error

j = "123"     # string with ONLY numbers
print(int(j)) # 123
```

Conversion with int()

```
i = "123abc"  # string
print(int(i)) # Error

j = "123"     # string with ONLY numbers
print(int(j)) # 123
              # Only integers in strings would be
              # successfully converted
```

Conversion with int()

```
i = "123abc"  # string
print(int(i)) # Error

j = "123"     # string with ONLY numbers
print(int(j)) # 123
              # Only integers in strings would be
              # successfully converted

k = "123.123" # string with ONLY numbers, but with
              # a number that represents a float
print(int(k))
```

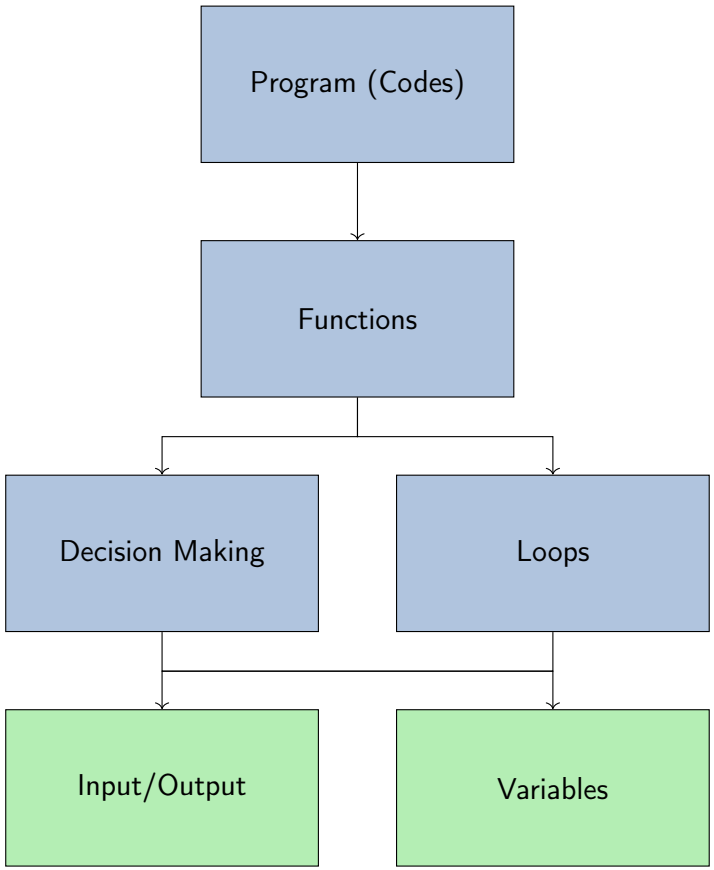
Conversion with int()

```
i = "123abc"  # string
print(int(i)) # Error

j = "123"     # string with ONLY numbers
print(int(j)) # 123
              # Only integers in strings would be
              # successfully converted

k = "123.123" # string with ONLY numbers, but with
              # a number that represents a float
print(int(k)) # Error
```

Contents


Program (Codes)
Functions
Decision Making
Loops
Input/Output
Variables

Variables in output - Revisiting the `print()` function

How do we print variables?

```
a = 5
print(a)
```

Variables in output - Revisiting the `print()` function

How do we print variables?

```
a = 5
print(a)          # 5
```

Variables in output - Revisiting the `print()` function

How do we print variables?

```
a = 5
print(a)          # 5
b = "haha"
print(b)
```

Variables in output - Revisiting the `print()` function

How do we print variables?

```
a = 5
print(a)          # 5
b = "haha"
print(b)          # haha
```

Variables in output - Revisiting the `print()` function

How do we print variables?

```
a = 5
print(a)          # 5
b = "haha"
print(b)          # haha
print(a + 2)
```

Variables in output - Revisiting the `print()` function

How do we print variables?

```
a = 5
print(a)          # 5
b = "haha"
print(b)          # haha
print(a + 2)      # 7
```

Variables in output - Revisiting the `print()` function

How do we print variables?

```
a = 5
print(a)          # 5
b = "haha"
print(b)          # haha
print(a + 2)      # 7
print(b + "a")
```

Variables in output - Revisiting the `print()` function

How do we print variables?

```
a = 5
print(a)          # 5
b = "haha"
print(b)          # haha
print(a + 2)      # 7
print(b + "a") # hahaa
```

Variables in output - Revisiting the `print()` function

How do we print variables?

```
a = 5
print(a)          # 5
b = "haha"
print(b)          # haha
print(a + 2)      # 7
print(b + "a") # hahaa
print(a, b)
```

Variables in output - Revisiting the `print()` function

How do we print variables?

```
a = 5
print(a)          # 5
b = "haha"
print(b)          # haha
print(a + 2)      # 7
print(b + "a")    # hahaa
print(a, b)       # 5 haha
```

Variables in output - Revisiting the `print()` function

How do we print variables?

```
a = 5
print(a)          # 5
b = "haha"
print(b)          # haha
print(a + 2)      # 7
print(b + "a") # hahaa
print(a, b)       # 5 haha
print(b, b)
```

Variables in output - Revisiting the `print()` function

How do we print variables?

```
a = 5
print(a)          # 5
b = "haha"
print(b)          # haha
print(a + 2)      # 7
print(b + "a")    # hahaa
print(a, b)       # 5 haha
print(b, b)       # haha haha
```

Variables in output - Revisiting the `print()` function

How do we print variables?

```
a = 5
print(a)          # 5
b = "haha"
print(b)          # haha
print(a + 2)      # 7
print(b + "a")    # hahaa
print(a, b)       # 5 haha
print(b, b)       # haha haha
```

The comma

Using `,` in `print()` would add a space in between the 2 items.

Variables in output - Revisiting the `print()` function

How do we print variables?

```
a = 5
print(a)          # 5
b = "haha"
print(b)          # haha
```

Variables in output - Revisiting the `print()` function

How do we print variables?

```
a = 5
print(a)          # 5
b = "haha"
print(b)          # haha
print(a + "5")
```

Variables in output - Revisiting the `print()` function

How do we print variables?

```
a = 5
print(a)          # 5
b = "haha"
print(b)          # haha
print(a + "5") # error
```

Variables in output - Revisiting the `print()` function

How do we print variables?

```
a = 5
print(a)          # 5
b = "haha"
print(b)          # haha
print(a + "5") # error
print(b + 2)
```

Variables in output - Revisiting the `print()` function

How do we print variables?

```
a = 5
print(a)          # 5
b = "haha"
print(b)          # haha
print(a + "5") # error
print(b + 2)    # error
```

Variables in output - Revisiting the `print()` function

How do we print variables?

```
a = 5
print(a)          # 5
b = "haha"
print(b)          # haha
print(a + "5") # error
print(b + 2)    # error
print(a + b)
```

Variables in output - Revisiting the `print()` function

How do we print variables?

```
a = 5
print(a)          # 5
b = "haha"
print(b)          # haha
print(a + "5") # error
print(b + 2)    # error
print(a + b)    # error
```

Variables in output - Revisiting the `print()` function

How do we print variables?

```
a = 5
print(a)          # 5
b = "haha"
print(b)          # haha
print(a + "5") # error
print(b + 2)    # error
print(a + b)    # error
```

Addition

You cannot use addition to print things of incompatible types.

Variables in output - Revisiting the `print()` function

How do we print variables?

```
a = 5
b = 32
c = 32.0
print(a * b)     # 160
print(a * c)     # 160.0
```

Variables in output - Revisiting the `print()` function

How do we print variables?

```
a = 5
b = 32
c = 32.0
print(a * b)     # 160
print(a * c)     # 160.0
```

Takeaway

`print()` function evaluates the expression inside the brackets first before actually printing.

More on `print()` function

In Python, the `print()` function automatically adds a new line after execution. We, however, can stop that.
The `end=` tag allows us to define the character added when `print()` is executed.

```
print(5, end="")
print(4)
print("a", end="abc")
print("d", end=" ")
print("e")
```

More on `print()` function

In Python, the `print()` function automatically adds a new line after execution. We, however, can stop that.
The `end=` tag allows us to define the character added when `print()` is executed.

```
print(5, end="")
print(4)
print("a", end="abc")
print("d", end=" ")
print("e")
# What is the output?
```

More on `print()` function

In Python, the `print()` function automatically adds a new line after execution. We, however, can stop that.
The `end=` tag allows us to define the character added when `print()` is executed.

```
print(5, end="")
print(4)
print("a", end="abc")
print("d", end=" ")
print("e")
# What is the output?
# Output: 54
#         aabcd e
```

End of line

Remember to include a new line `\n` in the last line of a printed string. Else it may mess up the future outputs from other lines of the code or the computer terminal.

More on `print()` function

We mentioned that whenever `,` is used in `print()`, the items would be separated by a space.
This can actually be changed using the `sep=` tag.

More on `print()` function

We mentioned that whenever `,` is used in `print()`, the items would be separated by a space.
This can actually be changed using the `sep=` tag.

```
>>> print("100", 100, end="\n3\n")
```

More on `print()` function

We mentioned that whenever `,` is used in `print()`, the items would be separated by a space.
This can actually be changed using the `sep=` tag.

```
>>> print("100", 100, end="\n3\n")
>>> 100 100
    3
```

More on `print()` function

We mentioned that whenever `,` is used in `print()`, the items would be separated by a space.
This can actually be changed using the `sep=` tag.

```
>>> print("100", 100, end="\n3\n")
>>> 100 100
    3
>>> print("100", 100, sep="a", end="\n3\n")
```

More on `print()` function

We mentioned that whenever `,` is used in `print()`, the items would be separated by a space.
This can actually be changed using the `sep=` tag.

```
>>> print("100", 100, end="\n3\n")
>>> 100 100
    3
>>> print("100", 100, sep="a", end="\n3\n")
>>> 100a100
    3
```

More on `print()` function

Another example:

```
>>> a = 5
>>> b = 10
>>> print(a, b, a + b, end="20\n")
```

More on `print()` function

Another example:

```
>>> a = 5
>>> b = 10
>>> print(a, b, a + b, end="20\n")
>>> 5 10 1520
```

More on `print()` function

Another example:

```
>>> a = 5
>>> b = 10
>>> print(a, b, a + b, end="20\n")
>>> 5 10 1520
>>> print(a, b, a + b, sep="", end="20\n")
```

More on `print()` function

Another example:

```
>>> a = 5
>>> b = 10
>>> print(a, b, a + b, end="20\n")
>>> 5 10 1520
>>> print(a, b, a + b, sep="", end="20\n")
>>> 5101520
>>> print(a, b, a + b, end="20\n", sep="")
```

More on `print()` function

Another example:

```
>>> a = 5
>>> b = 10
>>> print(a, b, a + b, end="20\n")
>>> 5 10 1520
>>> print(a, b, a + b, sep="", end="20\n")
>>> 5101520
>>> print(a, b, a + b, end="20\n", sep="")
>>> 5101520
```

More on `print()` function

Another example:

```
>>> a = 5
>>> b = 10
>>> print(a, b, a + b, end="20\n")
>>> 5 10 1520
>>> print(a, b, a + b, sep="", end="20\n")
>>> 5101520
>>> print(a, b, a + b, end="20\n", sep="")
>>> 5101520
```

Command Parameters

As long as you mark `sep` and `end` clearly **and** after the things you want to print, the ordering doesn't matter!

Converting the type of an input

How do we convert the data type of variables?

```
>>> number = input("Enter your number: ")
Enter your number: 50
>>> print(number)
```

Converting the type of an input

How do we convert the data type of variables?

```
>>> number = input("Enter your number: ")
Enter your number: 50
>>> print(number)
50
```

Converting the type of an input

How do we convert the data type of variables?

```
>>> number = input("Enter your number: ")
Enter your number: 50
>>> print(number)
50
>>> print(number + 1000)
```

Converting the type of an input

How do we convert the data type of variables?

```
>>> number = input("Enter your number: ")
Enter your number: 50
>>> print(number)
50
>>> print(number + 1000) # Error occurs. Why?
```

Converting the type of an input

How do we convert the data type of variables?

```
>>> number = input("Enter your number: ")
Enter your number: 50
>>> print(number)
50
>>> print(number + 1000) # Error occurs. Why?
```

Explanation

number is a string type while 1000 is an integer.

Converting the type of an input

How do we convert the data type of variables?

```
>>> number = input("Enter your number: ")
Enter your number: 50
>>> print(number)
```

Converting the type of an input

How do we convert the data type of variables?

```
>>> number = input("Enter your number: ")
Enter your number: 50
>>> print(number)
50
```

Converting the type of an input

How do we convert the data type of variables?

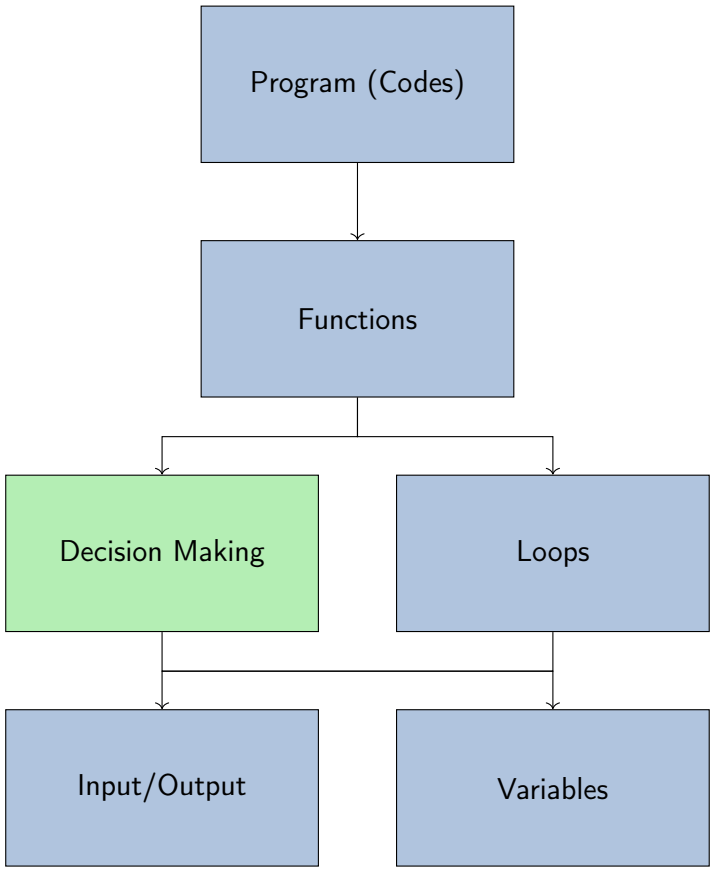
```
>>> number = input("Enter your number: ")
Enter your number: 50
>>> print(number)
50
>>> print(int(number) + 1000)
```

Converting the type of an input

How do we convert the data type of variables?

```
>>> number = input("Enter your number: ")
Enter your number: 50
>>> print(number)
50
>>> print(int(number) + 1000) # 1050
```

Contents


Program (Codes)
Functions
Decision Making
Loops
Input/Output
Variables

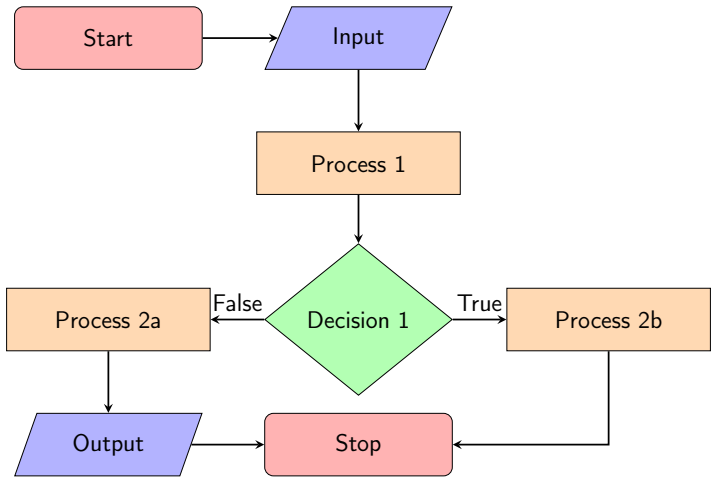
Decision Making

What is decision making?

Decision Making

What is decision making?

We use condition(s) to decide whether some code should be run.



The if clause

```
a = 5 # a stores the integer 5
if (a == 5):
    print("a stores 5.") # This line is activated

b = 10 # b stores the integer 10
if (b == 5)
    print("b stores 5.") # This line is not activated
```

The if clause

If the condition is true, then the code under it is run.

The == operator

```
a = 5 # a stores the integer 5
if (a == 5):
    print("a stores 5.") # This line is activated

b = 10 # b stores the integer 10
if (b == 5)
    print("b stores 5.") # This line is not activated
```

The == operator

The operator == is used to compare 2 values. If the values on the both sides are the same, then it becomes `True`. It becomes `False` otherwise.

The if-else clause

```
a = 5 # a stores the integer 5
if (a == 5):
    print("a stores 5.") # This line is activated
else:
    print("a does not store 5.")

b = 10 # b stores the integer 10
if (b == 5)
    print("b stores 5.")
else:
    print("b does not store 5.") # This line is activated
```

The else statement

Code under the `else` statement is executed when the condition in `if` is not true.

The if-else clause

```
a = 5 # a stores the integer 5
if (a == 5):
␣␣␣␣print("a stores 5.") # This line is activated
else:
␣␣␣␣print("a does not store 5.")

b = 10 # b stores the integer 10
if (b == 5)
␣␣␣␣print("b stores 5.")
else:
␣␣␣␣print("b does not store 5.") # This line is activated
```

Indentation in Python

Indentation decides whether the code is under the if/else statements.
It does not have to be 4 spaces, but they have to be **consistent**.

The if-elif-else clause

```
a = 5 # a stores the integer 5
if (a == 5):
    print("a stores 5.") # This line is activated
elif (a == 10):
    print("a stores 10.")
else:
    print("a does not store 5 or 10.")
```

The elif statement

The elif (stands for else-if) statement is a secondary if statement that is run if the previous if/elif condition(s) are not true.

The if-elif-else clause

```
a = 15 # a stores the integer 15
if (a == 5):
    print("a stores 5.")
elif (a == 10):
    print("a stores 10.")
elif (a == 15):
    print("a stores 15") # This line is activated
else:
    print("a does not store 5, 10 or 15.")
```

Stacking the elif statement

The elif statement can be stacked on top of one another.

Comparison Operators

We've learnt that == means "equal to". What are some other operators?

Operator	Meaning
==	equal to
>	larger than
>=	larger than or equal to
<	smaller than
<=	smaller than or equal to
!=	not equal to

Decision Making and Comparison Operators

```
a = 10 # a stores the integer 10
if (a > 5)
    print("a is larger than 5")

if (a >= 10)
    print("a is larger than or equal to 10")
```

In this example, both `print()` statements are activated.

Decision Making and Comparison Operators

```
a = 10 # a stores the integer 10
if (a > 5)
    print("a is larger than 5")
elif (a >= 10)
    print("a is larger than or equal to 10") # Not run
```

Decision Making and Comparison Operators

```
a = 10 # a stores the integer 10
if (a > 5)
    print("a is larger than 5")
elif (a >= 10)
    print("a is larger than or equal to 10") # Not run
```

In this example, only the first `print()` statements are activated.

`if` vs `elif`

If a condition is fulfilled, any `elif` clauses afterwards will not be considered.

Summary

There are 4 basic variable types: `int`, `bool`, `float` and `str`.
Some basic and commonly-used operators: `+`, `-`, `*`, `/`, `//`, `%`, `**`.

To convert between types, you can simply surround the target with brackets, and call the type.
`int -> int()`; `bool -> bool()`; `float -> float()`; `str -> str()`.

Summary

The `print()` statement:

```
print(*objects, sep=' ', end='\n', file=None, flush=False)
```

where
`*objects` refer to the things you want to print,
`sep` refers to the separation between each object,
(i.e. when commas are used.)
`end` refers to the string to end the print statement with.
The other arguments can be ignored as they are rarely used.

The `input()` statement:

```
input(prompt)
```

where `prompt` is quite literally what it means. It prints the output then returns the value inputted as a string.

The end

Written in LaTeX

Last updated: 13 Mar 2024