Basic data types in python Introduction to Programming in Python

Numbers

In Python, there are two different types of numbers:

- An integer value is a whole number without a fractional part. In Python, this type is called `int`.
- When a fractional part is required (such as in the number 0.355), we use floating-point numbers, which are called `float` in Python.

Arithmetic

Python supports all of the basic arithmetic operations: `+`, `-` add, subtract `*`, `/` `**` multiply, divide, power

In python, expressions are written a bit differently: $\frac{a+b}{2}$ would be written as `(a + b) / 2`

Precedence is similar to algebra:

PEMDAS - Parentheses, Exponent, Multiply/Divide, Add/Subtract

Floor Division

When you divide two integers with the `/` operator, you get a floating-point value so `7 / 4` yields `1.75`

We can also perform *floor division* using the `//` operator. The `//` operator computes the quotient and discards the fractional part:

```
7 // 4
```

Evaluates to 1 because 7 divided by 4 is 1.75 with a fractional part of 0.75, which is discarded

Remainder (or Modulo Divide)

If you are interested in the remainder of dividing two integers, use the % operator (called modulus):

```
remainder = 7 % 4
```

The value of remainder will be 3, this is sometimes called modulo divide.

Mixing numeric types

If you mix integer and floating-point values in an arithmetic expression, the result is a floating-point value:

```
7 + 4.0 # Yields the floating value 11.0
```

Try it out

What are the results of the following expressions:

- `3 * 2 + 6 % 5`
- `5 // 3 + 5 % 2`
- What is the Python code that represents the formula $c = (a / b)^3$?
  1. `c = a / b ** 3`
  2. `c = (a / b) ** 3`
  3. `c = 3 ^ (a / b)`
  4. `c = (a / b) ^ 3`

Variables

A variable is a named storage location in a computer program

You 'define' a variable by telling the interpreter:

- What name you will use to refer to it
- The initial value of the variable
- You use an assignment statement to place a value into a variable:

```
year_now = int(input("Enter the current year and press RETURN: "))
```

Variable names

Variable names should describe the purpose of the variable `canVolume` or `can_volume` is better than `cv`

Use these simple rules:

- Variable names must start with a letter or the underscore (_) character
- Continue with letters (upper or lower case), digits or the underscore
- Do not use other symbols (? or %...) and spaces are not permitted
- Separate words with an underscore _ to signify word boundaries, this is sometimes called *snake_case*.
- Don't use 'reserved' Python words.

Constants

In Python a constant is a variable whose value should not be changed after it's assigned an initial value.

It is a good practice to use all caps when naming constants:

It is good style to use named constants to explain numerical values to be used in calculations:

```
age_now = int(input("Enter your current age in years: "))

if age_now > MAXIMUM_AGE:
    print("Invalid age")
```

Note: Python will let you change the value of a constant *but* you should not do it.

Arithmetic assignment operators

Python provides a shorthand notation for simple arithmetic operations followed by assignment, as in the following examples:

Example	Equivalent to
`number += extra`	`number = number + extra`
`total -= discount`	`total = total - discount`
`bonus *= 2`	`bonus = bonus * 2`
`time /= rush_factor`	`time = time / rush_factor`
`change %= 100`	`change = change % 100`
`amount *= n1 + n2`	`amount = amount * (n1 + n2)`

This shorthand should not be used at the expense of having a program that is easily readable.

Using built-in functions

A function is a collection of programming instructions that carry out a particular task.

The `print()` function can display information, but there are many other functions available in Python.

Most functions return a value. That is, when the function completes its task, it passes a value back to the point where the function was called. For example: the call `abs(-173)` returns the value `173`.

The value returned by a function can be stored in a variable:

```
distance = abs(x)
```

You can use a function call as an argument to the `print()` function

```
print(abs(-173))
```

Built-in mathematical functions

Link here for a list of all python built-in functions.

Function	Returns
`abs(x)`	The absolute value of x
`round(x)` `round(x, n)`	The floating point value of x rounded to a whole number or to n decimal places.
`max(x1, x2, x3, ....,xn)`	The largest value from the parameters.
`min(x1, x2, x3, ....,xn)`	The smallest value from the parameters.

Rounding errors

Floating point values are not exact. This is a limitation of binary values; not all floating point numbers have an exact representation. Let's try this:

```
price = 4.35
quantity = 100
total = price * quantity
# Should be 100 * 4.35 = 435.00
print(total)
```

This doesn't quite do what you might expect. That's OK, we can:

- round to the nearest integer using the `round()` function.
- or display a fixed number of digits after the decimal separator. We will see how to do that in the section on Input and Output.

Strings

String definition

Start with a simple definition:

- A string is a sequence of characters consisting of characters - letters, numbers, punctuation marks, spaces ...
- In Python, string literals are specified by enclosing a sequence of characters within a *matching pair* of either single or double quotes:

```
print("This is a string.", 'So is this.')
```

- By allowing both types of delimiters, Python makes it easy to include an apostrophe or quotation mark within a string:

```
message = 'He said "Hello"'
```

Remember to use matching pairs of quotes, single with single, double with double

String length

The number of characters in a string is called the length of the string (For example, the length of “Harry” is 5)

You can compute the length of a string using Python’s `len()` function:

```
length = len("World!") # length is 6
```

A string of length 0 is called the empty string. It contains no characters and is written as ”” or ”

Concatenation and Repetition

You can 'add' one string onto the end of another:

```
firstName = "Harry"
lastName = "Morgan"
name  = firstName + lastName  # HarryMorgan
print("my name is:", name)
```

Using "+" to concatenate strings is an example of a concept called operator overloading. The + operator performs different functions of variables of different types.

The * operator is also overloaded

```
dashes = "-" * 50
```

results in the string

"--"

Escape sequences and special characters

Sometimes you might want to print a double quote but how can you do that?

Preface the " with a "\" inside the double quoted string

```
print("He said \"Hello\"")
```

There is another way to achieve the same thing, simply enclose the string with single quotes:

```
print('He said "Hello"')
```

To print a backslash, preface the with another

```
print("C:\\Temp\\Secret.txt")
```

Input and Output

Input

You can use the `input()` function to prompt the user for a value.

```
name = input("Enter your name: ")
```

If you need a numeric value then you can *wrap* the input function in the `int()` function.

```
age_now = int(input("Enter your current age in years: "))
```

Similarly, there is also a `float()` function:

```
temperature_now = float(input("Enter the temperature: "))
```

Formatted Output

Printing floating point values can look strange:

```
Price per litre: 1.21997
```

There are three ways to control the output appearance of printed output:

- use format specifiers
- use the string `format()` method
- use f'strings (formatted string literals)

These are a bit tricky for new programmers to understand and textbooks for beginners often don't cover the topic.

My advice is: in any program, choose one style and stick to it.

I'm going to describe the `f` strings method because it's the most recent addition to the language.

Using f‘ strings with numbers

Example - print variables in some text:

```
price = 10.345
quantity = 5
total_cost = price * quantity
print(f'Quantity: {quantity:2d} unit price: {price:.4f}\
      TOTAL: {total_cost:.2f}')
```

Placeholders `{}` are used to mark where the output should go.

Output:

```
Quantity:  5 unit price: 10.3450  TOTAL: 51.73
```

Format	Meaning
`quantity:2d`	Print the integer variable `quantity` in a field 2 characters wide
`price:.4f`	Print the float `price` with 4 decimal places
`total_cost:.2f`	Print the float `total_cost` with 2 decimal places