Expressions and Variables

Expressions

```
print(expression)
```

An expression is *evaluated* to give a *value*.

For example:

```
2 + 9 - 6
```

Evaluates to:

```
5
```

Data Types

- Integers 1, 2, 3, 42, 100, -5
- Floating points 2.5, 7.0, 3.14159
- Strings “Hello World!”, “Learn to Code”, “100”
- Booleans True / False

Numeric Expressions

We shall treat integers and floating points interchangeably for now.

Remember, we can output expressions using `print`.

Numbers on their own are expressions:

```
print(42)
```

We can do calculations:

```
print(2+2)
```

Mathematical Operations

- `+` `2 + 2 = 4`
- `-` `4 - 2 = 2`
- `*` `6 * 7 = 42`
- `/` `42 / 7 = 6.0` (← Note the floating point answer)
- `**` `2 ** 4 = 16`
- `%` `18 % 5 = 3` (Modulo - gives remainder)
- `()` `(2 + 3) * 5 = 25`

Operation Precedence

```
  2 * 4 + 3 * 3
=   8   +   9
= 17
```

```
  2 * (4 + 3) * 3
= 2 *    7    * 3
= 42
```

Exercise: Calculations

Write a program that calculates and outputs the following:

$$\frac{-6+\sqrt{6^2-4\times3\times-15}}{2\times3}$$

Hint: Square root is the same as raising to the power of 0.5.

Exercise: Calculations

Write a program that calculates and outputs the following:

$$\frac{-6+\sqrt{6^2-4\times3\times-15}}{2\times3}$$

Hint: Square root is the same as raising to the power of 0.5.

```
(-6 + (6*6 - 4*3*-15) ** 0.5) / (2*3)
(-6 + (6**2 - 4*3*-15) ** 0.5) / (2*3)

1.449489742783178
```

String Expressions

Text is represented using ‘strings’. (a ‘string of characters’)

```
“Hello World!”
“Learn to Code”

‘Hello World’
‘Learn to Code’
```

Watch out if copying from word-processor to text editor...

Simple String Operations

Concatenation is attaching strings together:

```
  "Hello" + " " + "World!"
= "Hello World!"
```

We can make strings repeat:

```
  "badger" * 3
= "badgerbadgerbadger"
```

Note that we can't use every operation! Subtracting / dividing strings doesn't really make sense!

String Formatting

Remember that Strings and Numbers (Integers / Floats) are different *data types*!

The following will not work, because we cannot mix data types. '+' has different meanings for Strings and Numbers.

```
"I am " + (10 + 3) + " years old."
```

Expressions inside quotes will not be evaluated - they will be treated as strings. So this does not give the desired result:

```
"I am " + "(10 + 3)" + " years old."
```

The solution is *string formatting*.

String Formatting

```
"I am %d years old." % (10 + 3)
```

%d is a *format specifier* - in this case, it represents an integer.

When the string expression is evaluated, format specifiers are replaced with the values listed after the %.

In the above example:
- 10 + 3 is evaluated to 13.
- Since %d represents integer formatting, the integer 13 is converted to the string “13”.
- %d is replaced with “13” in the string.

Format Specifiers

A few common format specifiers:

- %d, %i Integers
- %f Floating points
- %e Floating points in scientific notation
- %s Strings

Multiple values should be given as a comma-separated list in brackets:

```
    “My name is %s, I am %d years old, and I have %d %s.” %
(“Alex”, 13, 7, “friends”)
```

Exercise: String Formatting

```
“My name is %s, I am %d years old, and I have %d %s.” %
(“Alex”, 13, 7, “friends”)
```

Adjust the above expression to be about yourself by changing the parameters.

Adjust the expression so that ‘years’ is a parameter, rather than part of the string.

Then, adjust the parameters to display the same unit of time in months rather than years.

Exercise: String Formatting

```
“My name is %s, I am %d years old, and I have %d %s.” %
(“Alex”, 13, 7, “friends”)
```

Adjust the above expression to be about yourself by changing the parameters.

Adjust the expression so that ‘years’ is a parameter, rather than part of the string.

Then, adjust the parameters to display the same unit of time in months rather than years.

```
“My name is %s, I am %d %s old, and I have %d %s.” %
(“Alex”, 13*12, “months”, 7, “friends”)
```

But why do we need expressions?

So far, we have been hard-coding these expressions.

We could have calculated the results ahead of time!

So why do expressions exist?

Variables!

We want to be able to store the results of our calculations somewhere for later use.

We want to be able to remember information a user has given us.

We might want to use these values in later calculations.

Variables

A *variable* is a piece of memory that we can give a label and set aside for storing a value.

We can define as many as there is space for!

```
variablename = expression
```

```
x = 10 * 8 + 5
age = 32
firstname = “Alice”
```

Valid variables names can contain letters, numbers, and underscores, but cannot start with a number.

Using Variables

We must define variables before we use them.

```
print(x)  # Error!
x = 10
```

We can change the values of variables after we have defined them using the same syntax.

```
x = 10
print(x)  # Outputs 10
x = 4
print(x)  # Outputs 4
```

Using Variables

It is possible to reassign a variable with a new data-type, but this is usually bad practice.

```
x = 10          # x is an Integer
print(x)        # Outputs 10
x = “Ten”       # Reassign x to be a String
print(x)        # Outputs Ten
```

In some other languages, this is impossible.

Exercise: Variables

```
“My name is %s, I am %d years old, and I have %d %s.” %
(“Alex”, 13, 7, “friends”)
```

Create appropriately-named variables for each of the four parameters, and print the string using those variables rather than hard-coded parameters.

Now add a second `print` statement that outputs the same information in a different order. e.g. “I am a 13 year old with 7 friends, called Alex.”

Try changing the values in the variables and re-running the program - observe how the output changes.

User Input

Inputs need to be stored in variables so they can be accessed later.

We do this using `input`, which acts as follows:

- A message is shown to the user, asking for input.
- The program waits for the user to enter some input and press enter.
- The *value* that was input is returned.

We can treat this as an expression, and therefore assign it to a variable.

```
animal = input("Please enter your favourite animal: ")
```

Exercise: User Input

Write a program that asks the user for their name, then outputs a greeting using it.

Exercise: User Input

Write a program that asks the user for their name, then outputs a greeting using it.

- ```
  name = input("What is your name? ")
  print("Hello, %s." % name)
  ```
- ```
  name = input("What is your name? ")
  print("Hello, " + name + ".")
  ```
- ```
  print("Hello, %s." % input("What is your name? "))
  ```

Which of these is better?
  ```

Exercise: Calculations with User Input

Write a program that asks for two numbers, then prints their sum.

What could go wrong here?

Exercise: Calculations with User Input

Write a program that asks for two numbers, then prints their sum.

What could go wrong here?

```
a = input(“Enter a number: “)
b = input(“And another: “)
print(a + b)
```

a and b are strings, so they are concatenated! Inputting 2 and 3 will output 23.

Converting Data Types

We can convert from one data type to another using the following functions:

- `int()`
- `float()`
- `str()`

These will give an error if there is no sensible conversion!

```
int(“hello”)  # Error!
```

Exercise: Calculations with User Input

Write a program that asks for two numbers, then prints their sum, *taking account of type conversion*.

Broken version:

```
a = input("Enter a number: ")
b = input("And another: ")
print(a + b)
```

Exercise: Calculations with User Input

```
# Method 1
a = int(input("Enter a number: "))
b = int(input("And another: "))
print(a + b)

# Method 2
a = input("Enter a number: ")
b = input("And another: ")
print(int(a) + int(b))
```

Exercise: Circle Equations

Write a program that takes as input the *radius* of a circle, and outputs both the *circumference* and the *area* of that circle.

Bonus points for outputting a formated string rather than a list of numbers!

Maths!

$\pi = 3.14159$

circumference = $2\pi r$

area = πr^2

Exercise: Circle Equations

```
r = float(input("Radius: "))
pi = 3.14159
circ = 2 * pi * r
area = pi * r * r
print("Circumference: %f" % circ)
print("Area: %f" % area)
```

Summary

- Output to terminal
- Expressions - calculations and string building
- Variables - and their use in expressions
- User input from terminal

How do we make decisions based on those variables?