Exercise 6: Values, variables, and expressions

Values

We have now seen examples of two types of values: integers and strings. As demonstrated using the `print` function, these two types of values are interpreted differently. Strings, which are just sequences of characters, such as *Hello, world!*, have the designation **`str`**. Integers, which are of course whole numbers, are one type of numerical value of type **`int`**. Floating-point numbers (numbers containing a decimal point) belong to a second type called **`float`**. Numbers, both **`int`** and **`float`** type, can be treated as strings but strings cannot be treated as numbers, as we demonstrated using the **`print`** function.

Variables

Variables can be assigned numbers (either **`int`** or **`float`**) or strings (**`str`**). For example, we can create a variable **`dna`**, and assign a sequence of As, Cs, Ts, and Gs to it as follows (recall that variables are assigned with the syntax **`variable_name = value`**):

In [3]:
```
dna = 'ATGATGCAT'
print(dna)
```

```
ATGATGCAT
```

Because we are assigning a string to the variable **`dna`**, as with the **`print`** function, the value has to be in quotes. What if we were assigning a number to the variable **`dna`**?

In [5]:
```
dna = '5'
dna + 5
```

```
---------------------------------------------------
TypeError                 Traceback (most recent call last)
<ipython-input-5-abcefc319b77> in <module>()
      1 dna = '5'
----> 2 dna + 5

TypeError: must be str, not int
```

Things start to get a little bit tricky. If we include quotes around a number, than it becomes a string and a string no longer has a numerical value. So even though the variable **`dna`** may appear to be a number, it will depend on if it was assigned with or without quotes.

Assign a number to a variable without quotes:

In [7]:
```
dna = 5
```

Use the variable in a mathematical operation:

In [8]:
```
dna + 5
```

Out[8]:
```
10
```

Now try assigning a number to a variable with quotes:

In [33]:
```
num = '5'
```

Try using the variable in a mathematical operation:

In [34]:
```
num + 5
```

```
---------------------------------------
TypeErrorTraceback (most recent call last)
<ipython-input-34-c338d09802bb> in <module>()
----> 1 num + 5

TypeError: must be str, not int
```

To figure out what type of value a variable is, use **`type(variable_name)`** :

In [10]:
```
dna = '5'
type(dna)
```

Out[10]:
```
str
```

Variable names can be just about anything in Python but they cannot start with a number or have spaces or special characters (underscores are ok) . It is best to give variables descriptive names and use lowercase letters, in particular the first letter should be lower case. And although it is permissible, it is sometimes confusing to give a variable the same name as a function (such as **`print`**) and Python has ~30 special kewords that are off limits.

Statements

In Python, a statement is any code that can be executed. **`1+1`** is a statement. **`dna = "ATGCC"`** is also a statement. Statements can be thought of as any action or command.

Expressions

An expression is something that represents something. An expression can be a number or a string. `1+1` is an expression. Basically, anything that has a value is an expression. It can be a single value, a combination of values, variable, and operators, and calls to functions as long as it boils down to a single value. Expressions are things that need to be evaluated. As we saw earlier, in interactive mode, the value of expressions are returned upon hitting enter but in a script, they are not. Any section of code that evaluates to a value within a statment is an expression.

```
In [36]: 1
```

```
Out[36]: 1
```

```
In [35]: 1+1
```

```
Out[35]: 2
```

Operators

Earlier we introduced several mathematical operators: `+`, `-`, `/`, `*`. The operator `%` is called the modular operator but it returns the remainder of a division.

What is the remainder of `11/3`?

```
In [13]: 11%3
```

```
Out[13]: 2
```

It may not seem particularly useful now, but when working with large datasets, it can come in handy.

We saw the plus operator, `+`, in a mathematical context, but it can also be used to concatenate strings:

```
In [14]: 'color' + 'ado'
```

```
Out[14]: 'colorado'
```

The `input` function

Python's input function - **`input()`** - allows you to collect input from a user and then perform actions on that input:

In [15]:
```
input()
```

```
ACTGACTAC
```

Out[15]:
```
'ACTGACTAC'
```

Not particularly useful on its own, but the input can also be stored as a variable or directly incorporated into a function.

Try storing input as a variable:

In [18]:
```
dna = input("please enter a dna sequence: ")
```

```
please enter a dna sequence: ATGACTGCATCA
```

Now print the user input using the **`print`** function.

In [19]:
```
print(dna)
```

```
ATGACTGCATCA
```

What if we wanted to print a literal string and the value of a variable? Python actually makes it somewhat complicated because literal strings require quotes and variables are not interpolated if containted in quotes.

Print the following statement: **Your sequence is `variable`.**, where ***`variable`*** is the value assigned to the variable by the user. There are several ways to do this, but for now we will focus on using `+` signs to delimit strings from variables. The syntax is as follows: **`print("string" + variable + "string")`** . A drawback to this approach is that it does not work with **`int`** and **`float`** type values. But **`int`** and **`float`** values can be converted to string type using **`str(value)`** within most statements.

In [32]:
```
dna = 'ACTGACT'
print('Your sequence is: ' + str(dna))
```

```
Your sequence is: ACTGACT
```

Independent Exercise 6

Write a script that prompt the user for two separate RNA sequences and then concatenate the sequences and print the result.

!!! Submit your script on Canvas as Exercise 6 !!!

Questions

Q. What are the three different types of values we discussed?

A. integers (int), floating-point (float), and strings (str)

Q. Does Python care about white space and empty lines? If so, when?

A. Spaces are typicall okay, but not in variable names. Tabs are required for delimiting blocks of code in certain contexts, such as conditional statements. But most of the time Python is pretty forgiving about spaces.

Q. When should you put quotes around the value you are assigning to a variable? Does it matter if you use single or double quotes?

A. Quotes are required around strings. Single and double quotes are interchangable.

Q. We have now seen two functions in Python: `print()` and `input()`. What similarities do you notice between these functions? These are two of several built in funtions in Python.

A. Syntax is similar in that you call the function and provide arguments within the parentheses. The function than does some action and that action is specific to the function. Functions are a major part of python programming.