# Variables

## Contents

A variable is an object that contains an assigned value:

```
a = 10          # a is a variable containing a number
b = "hello"     # b is a variable containing a string (text)
```

Certain words have a special meaning in Python and cannot be used as variable names. These are: `and, as, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, with, while,` and `yield`.

Good variable names are usually descriptive, can be one letter symbols, abbreviation of words, containing lower/upper case, underscores, digits. They cannot start with digits. Variables are case sensitive so variable `a` is not the same as `A`.

## Types of variables

Common types of Python variables are:

- *str* - string, e.g. `"Hello world!"`
- *int* - integer numbers, e.g. `5`
- *float* - floating point numbers, e.g. `5.0`, `5.2`
- *bool* - boolean value, i.e. `True` or `False`

Integer and floating point numbers are different types of variables and a computer stores them differently. Computers use a discrete representation of numbers, which means that there is a gap between two adjacent numbers. For integers this gap is 1 and for floats it is much much smaller. For example, in integer arithmetic 0 and 0.999999 are the same numbers. We will therefore almost always use floating point numbers.

```
a = int(0.99999999)
b = int(1.00000001)
print(a, b)
```

```
0 1
```

To check what type a variable is, we can use the in-built `type()` function that takes the variable as an argument:

```
a = "some text"     # string
b = 5               # integer
c = 5.              # float
d = True            # boolean

print(type(a))
print(type(b))
print(type(c))
print(type(d))
```

```
<class 'str'>
<class 'int'>
<class 'float'>
<class 'bool'>
```

## Operations on variables

### Numbers

An integer-type variable will automatically change to float if it is required for mathematical accuracy:

```
b = 10      # An integer
b = 1 + b   # b should be still an int
print(b, type(b))

b = 1.1 + b # b should change into float
print(b, type(b))

b = 10      # An integer
b = b/4     # A float
print(b, type(b))
```

```
11 <class 'int'>
12.1 <class 'float'>
2.5 <class 'float'>
```

We can make calculations on numbers stored in variables and program mathematical formulae. For example, a position for a ball in vertical motion is given by: $y(t) = v_0 t - \frac{1}{2} g t^2$ where the ball starts at $y = 0$ at time $t = 0$ and:

- $y$ is the height (position) as a function of time $t$
- $v_0$ is the initial velocity (at $t = 0$)
- $g$ is the acceleration due to gravity

Given $v_0$, $g$ and $t$, we can compute the value $y$:

```
v0 = 12.5   # m/s
g = 9.81    # m/s2
t = 0.3     # s
y = v0*t - 0.5*g*t**2
print("The position of the ball at time t = %.2f s is y = %.2f m." % (t, y))
```

```
The position of the ball at time t = 0.30 s is y = 3.31 m.
```

## Strings

We can add strings together:

```python
a = "some text"           # A string
print(a, type(a))
a = a + " and more text"   # Add a string at the end of 'a'
print(a, type(a))
```

```
some text <class 'str'>
some text and more text <class 'str'>
```

You can slice strings to get specific parts of them by using the following syntax:

```python
# Print characters 0, 1 excluding 2
# (Python counts from zero!!!)
print(a[0:2])

# Print all characters but last two
print(a[0:-2])

# Print every second character
# from first 10 characters
print(a[0:10:2])
```

```
so
some text and more te
sm et
```

Searching strings:

```python
s = "To what use serves learning, if understanding be away."

print(s)
print(s.find("use"))  # returns the index of the first character if found
print("if" in s)      # returns True if string 'if' is contained in 's', False othe
print("is" not in s)  # returns True if string 'is' is not contained in 's', False

s2 = s.replace("learning", "studying")  # replaces learning with studying in 's'
print(s2)

s3 = s.replace("learnning", "studying")  # does nothing if we make a typo
print(s3)
```

```
To what use serves learning, if understanding be away.
8
True
True
To what use serves studying, if understanding be away.
To what use serves learning, if understanding be away.
```

Splitting and joining:

```
s = "Dare to be wise; begin!"
words = s.split()       # Splits the string based on spaces
print(words)

phrases = s.split(";") # Add a string by which splitting will happen
print(phrases)

print(":".join(phrases)) # Join phrases with a collon
```

```
['Dare', 'to', 'be', 'wise;', 'begin!']
['Dare to be wise', ' begin!']
Dare to be wise: begin!
```

Other useful functions:

```
s = "I like watermelons! I like trains! I like books!"

print(s.count("!"))      # Count how many times string '!' occurs in 's'
print(s[5].isnumeric())  # Check if the sixth character is a number
print(s.lower())         # Make all characters lower case
print(s.upper())         # Make all characters upper case
print(s.isalpha())       # Check if all characters are letters
print(s[2:5].isalpha())  # Check if the characters 3 to 5 are letters

s = "    " + s + "    "
print(s)
print(s.strip())         # Strips strings from tabs, spaces, new line characters fr
                         # beginning and end of the string

print(len(s))            # Print how many characters the string has
```

```
3
False
i like watermelons! i like trains! i like books!
I LIKE WATERMELONS! I LIKE TRAINS! I LIKE BOOKS!
False
True
    I like watermelons! I like trains! I like books!
I like watermelons! I like trains! I like books!
56
```

## Booleans

Booleans are objects in Python that are `True` or `False`. The simplest booleans are:

```
print(True)
print(False)
```

```
True
False
```

Objects can be compared to one another, where `print` statements will return `True` or `False`. Standard mathematical formulae in Boolean expressions are:

- $a = b$ is `a == b`
- $a \neq b$ is `a != b`
- $a > b$ is `a > b`
- $a \geq b$ is `a >= b`
- $a < b$ is `a < b`
- $a \leq b$ is `a <= b`

An example use below:

```
a = 5
print(a > 10)
print(a == 10)
print(a <= 5)
print(a != 5)
print(a != "cat")
```

```
False
False
True
False
True
```

Compound conditions with logical operators can also be used:

```python
a = 5
b = 20

print((a < 6) and (b > 18)) # For 'and' both expressions need to be True to return
print((a < 5) and (b > 18)) # Returns false because the first expression is False
print((a < 5) or (b > 18))  # For 'or' only one expression is needed to be True

# 'and' and 'or' can be combined, but brackets matter!!

print((a > 10 and b < 10) or b > 13)
print(a > 10 and (b < 10 or b > 13))
```

```
True
False
True
True
False
```

## Exercises

- **Print** the value of the expression $(a + b - c^4) \div d$ where $a = 1; b = 4; c = 3; d = 17$. Check the type of the expression.

Answer

```
a = 1
b = 4
c, d = 3, 17

result = (a + b - c**4)/d

print(result)
print(type(result))
```

- **Integer division**: Very popular operators in Computer Science include integer (or *floor*) division `//` and modulus operator `%`. Think what operation combined with type casting can result in integer division. Check the working of your operator using the equality $div(a, b) * b + a\%b = a$

Answer

```
print(50//6 * 6 + 50%6 == 50)
```

- **String slicing**: Let `a = "Bananas"` and `b = "Apples"`:
  - concatenate the strings
  - take even characters from the resulting string
  - take the last three characters

Print the intermediate results.

Answer

```
a = "Bananas"
b = "Apples"

c = a + b
print(c)
c = c[::2]
print(c)
c = c[-3:]
print(c)
```

---

- **Lorem Ipsum**: Count the number of 'l' characters in the following text (used as dummy text in web development), then count the number of words:

*Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam ultrices tempus faucibus. Mauris id porta turpis, a sodales leo. Nam fringilla dolor at tellus auctor, ut sollicitudin augue gravida. Proin vitae sem ligula. Suspendisse vel erat elit. Sed fringilla massa ut iaculis vestibulum. Fusce placerat est id felis semper facilisis.*

**HINT**: There is whitespace between any two words!

Answer

```
a = "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam ultrice

print(a.count('l'))
print(a.count(' ') + 1)  # number of words
```

- **Boolean short circuit**: Consider the following code:

```
def f(a):
    a += 1
    return True

a = 0
if(1 > 2 or f(a)):
    print(a)
```

f defines an increment function, what will the programme print?

**Answer**

0