1. Variables and Assignments
   - assignment operators
   - memory management and references

2. Dynamic Typing
   - types and arithmetic
   - prompting for user input
Variables, Assignments, Dynamic Typing

1. Variables and Assignments
   - assignment operators
   - memory management and references

2. Dynamic Typing
   - types and arithmetic
   - prompting for user input
Assigning to Variables

how does an assignment work?

Stages in the execution of \( x = 3 \):

1. evaluate the right hand side of \( = \)
2. store the result of the evaluation in a register
3. compute the address of \( x \)
   and store it into the address register
4. copy the 3 from a register to the data register
5. put the value in data register at the memory location
   defined by the content of the address register
Memory Locations

A variable has a value and an address.

```python
>>> x = 3
>>> id(x)
33568624
```

The machine view:

- Address of `x`: 33568624
- Value of `x`: 3

In Python, the name `x` refers to the object 3.
Variables, Assignments, Dynamic Typing

1. Variables and Assignments
   - assignment operators
   - memory management and references

2. Dynamic Typing
   - types and arithmetic
   - prompting for user input
Memory Management, implicit and explicit

- Automatic allocation of memory for each object.
- Garbage collection frees space for unused variables.
- With `del` we may explicitly free space:

```python
>>> x = 3
>>> y = x
>>> id(x)
33568624
>>> id(y)
33568624

We see that both `x` and `y` refer to the same object.

```python
>>> del(x)
```

```python
>>> y
3
```

After deleting `x`, the object is still accessible via `y`. Only after `del(y)` will memory space be released.
Deleting References

```python
>>> x = 3; y = x
Y
refers to
name

x
refers to
object

>>> del(x)
X
refers to
object

Y
refers to
name
```
Variables, Assignments, Dynamic Typing

1. Variables and Assignments
   - assignment operators
   - memory management and references

2. Dynamic Typing
   - types and arithmetic
   - prompting for user input
Types of Variables – every variable has a type

- Python uses dynamic typing: the type of a variable is determined automatically at runtime.
  
  \texttt{type(name)} returns type of the variable \texttt{name}

- The type of a variable determines
  
  1. what kind of operations are available
  2. the outcome of those operations, e.g.: $1 + 2 \neq '1' + '2'$

- We have elementary and composite types:
  
  a character is an elementary type, while a string is a sequence of characters: a composite type.

- Elementary types: int, float, char, and boolean.
  
  A character is a string of length one. Python has no separate type for a character. The \texttt{+} operator applied to string concatenates.
Machine Integers

We distinguish between machine integers and long integers.

- The computer uses 32 or 64 bits to represent integers. First bit is sign bit: $0 = +$, $1 = -$.
- The largest machine int is $+2^{31} - 1$ or $+2^{63} - 1$:
  
  ```
  >>> 2**30 + (2**30 - 1)
  2147483647
  >>> 2**62 + (2**62 - 1)
  9223372036854775807
  ```
- The smallest machine int is $-2^{31}$ or $-2^{63}$:
  
  ```
  >>> -2**31
  -2147483648
  >>> -2**63
  -9223372036854775808
  ```

In Python, the use of long integers avoids overflow and underflow.
Boolean

We can store the outcome of logical expressions:

```python
>>> x = 3
>>> x == 3
True
>>> x == 4
False
```

For complicated logical expressions, we may want to save its outcome (Python session continued):

```python
>>> b = _
>>> type(b)
<type 'bool'>
```
Variables, Assignments, Dynamic Typing

1. Variables and Assignments
   - assignment operators
   - memory management and references

2. Dynamic Typing
   - types and arithmetic
   - prompting for user input
Getting User Input
unformatted or of specific type

Two ways to prompt the user for input:

1. \( s = \text{input}('\text{Give a number : }') \)
2. \( n = \text{int} (\text{input}('\text{Give a number : }')) \)

In both cases, the user sees Give a number :

The difference is in the type of the object returned

1. \( \text{input}() \) returns always a string
2. \( \text{int}(\text{input}()) \) converts the input string to an integer.

Why two ways?

1. the string returned by \( \text{input}() \) can be cast (converted) into an integer or a float, depending on its purpose;
2. a direct conversion is good if we assume correct input.
evaluation of Python literal structures

```python
>>> from ast import literal_eval
>>> sn = input('give a number : ')
give a number : 3
>>> n1 = literal_eval(sn)
>>> n1
3
>>> type(n1)
<class 'int'>
>>> sf = input('give a number : ')
give a number : 3.4
>>> n2 = literal_eval(sf)
>>> n2
3.4
>>> type(n2)
<class 'float'>
```

With `literal_eval` Python evaluates a string into the proper type.
Exercises

1. Can you explain why \( 1.1 + 0.1 \) shows \( 1.2000000000000002 \). Why is the outcome of \( 1.1 + 0.1 \) not exact?

2. Explain why the string \( \texttt{\%\.40e\% .5} \) is exactly the same as \( 0.5 \) while \( \texttt{\%\.40e\% .1} \) differs from the exact \( 0.1 \).

3. Write a Python program to convert a duration expressed in seconds in an hour, minutes, and seconds format. For example: 35781 seconds equals 9 hours, 56 minutes, and 21 seconds.

4. The formula \( f = \frac{9}{5}c + 32 \) converts \( c \) degrees of Celsius into \( f \) degrees of Fahrenheit. Write a Python program that prompts the user for \( c \) and prints \( f \).

5. Modify the yield and balance program for loans, i.e.: what is the balance of a loan when making a number of payments when the interest rate is fixed?