Python Cheat Sheet - The Basics

Math Operations

<table>
<thead>
<tr>
<th>Operators</th>
<th>Operation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>**</td>
<td>Exponent</td>
<td>2 ** 3 = 8</td>
</tr>
<tr>
<td>%</td>
<td>Modulus/Remainder</td>
<td>22 % 8 = 6</td>
</tr>
<tr>
<td>//</td>
<td>Integer division</td>
<td>22 // 8 = 2</td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
<td>22 / 8 = 2.75</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
<td>3 * 3 = 9</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
<td>5 - 2 = 3</td>
</tr>
<tr>
<td>+</td>
<td>Addition</td>
<td>2 + 2 = 4</td>
</tr>
</tbody>
</table>

Note: from highest to lowest precedence.

Walrus Operator

```python
>>> print(my_var:="Hello World!")
# 'Hello world!'

>>> my_var="Yes"
>>> print(my_var)
# 'Yes'

>>> print(my_var:="Hello")
# 'Hello'
```

Data Types

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integers</td>
<td>-2, -1, 0, 1, 2, 3, 4, 5</td>
</tr>
<tr>
<td>Floating-point numbers</td>
<td>-1.25, -1.0, -0.5, 0.0, 0.5, 1.0, 1.25</td>
</tr>
<tr>
<td>Strings</td>
<td>'a', 'aa', 'aaa', 'Hello!', '11 cats'</td>
</tr>
</tbody>
</table>

Variables

- It can be only one word.
- It can use only letters, numbers, and the underscore (_) character.
- It can’t begin with a number.
- Variable names starting with an underscore (_) are considered as “unuseful”.

Comments

- Inline comment:
  # This is a comment
- Multiline comment:
  # This is a
  # multiline comment
- Code with a comment
  a = 1  # initialization
- Function docstring:
  
  ```python
def foo():
    """
    This is a function docstring
    You can also use:
    """" Function Docstring """
    """
  
  The `input()` Function
  This function takes the input from the user and converts it into a string:
  
  ```python
  >>> print('What is your name?')  # ask for their name
  >>> my_name = input()
  >>> print('Hi, {}'.format(my_name))
  # What is your name?
  # Martha
  # Hi, Martha
  
  `input()` can also set a default message without using `print()`:
  
  ```python
  >>> my_name = input('What is your name? ')  # default message
  >>> print('Hi, {}'.format(my_name))
  # What is your name? Martha
  # Hi, Martha
  ```
  
  It is also possible to use formatted strings to avoid using `.format`:
  
  ```python
  >>> my_name = input('What is your name? ')  # default message
  >>> print(f'Hi, {my_name}')
  # What is your name? Martha
  # Hi, Martha
  ```

Python Built-In Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>abs()</code></td>
<td>Return the absolute value of a number.</td>
</tr>
<tr>
<td><code>aiter()</code></td>
<td>Return an asynchronous iterator for an asynchronous iterable.</td>
</tr>
<tr>
<td><code>all()</code></td>
<td>Return True if all elements of the iterable are true.</td>
</tr>
<tr>
<td><code>any()</code></td>
<td>Return True if any element of the iterable is true.</td>
</tr>
<tr>
<td><code>ascii()</code></td>
<td>Return a string with a printable representation of an object.</td>
</tr>
</tbody>
</table>
**bin()**  
Convert an integer number to a binary string.

**bool()**  
Return a Boolean value.

**breakpoint()**  
Drops you into the debugger at the call site.

**bytearray()**  
Return a new array of bytes.

**bytes()**  
Return a new “bytes” object.

**callable()**  
Return True if the object argument is callable, False if not.

**chr()**  
Return the string representing a character.

**classmethod()**  
Transform a method into a class method.

**compile()**  
Compile the source into a code or AST object.

**complex()**  
Return a complex number with the value real + imag*1j.

**delattr()**  
Deletes the named attribute, provided the object allows it.

**dict()**  
Create a new dictionary.

**dir()**  
Return the list of names in the current local scope.

**divmod()**  
Return a pair of numbers consisting of their quotient and remainder.

**enumerate()**  
Return an enumerate object.

**eval()**  
Evaluates and executes an expression.

**exec()**  
This function supports dynamic execution of Python code.

**filter()**  
Construct an iterator from an iterable and returns true.

**float()**  
Return a floating point number from a number or string.

**format()**  
Convert a value to a “formatted” representation.

**frozenset()**  
Return a new frozenset object.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getattr()</code></td>
<td>Return the value of the named attribute of object.</td>
</tr>
<tr>
<td><code>globals()</code></td>
<td>Return the dictionary implementing the current module namespace.</td>
</tr>
<tr>
<td><code>hasattr()</code></td>
<td>True if the string is the name of one of the object’s attributes.</td>
</tr>
<tr>
<td><code>hash()</code></td>
<td>Return the hash value of the object.</td>
</tr>
<tr>
<td><code>help()</code></td>
<td>Invoke the built-in help system.</td>
</tr>
<tr>
<td><code>hex()</code></td>
<td>Convert an integer number to a lowercase hexadecimal string.</td>
</tr>
<tr>
<td><code>id()</code></td>
<td>Return the “identity” of an object.</td>
</tr>
<tr>
<td><code>input()</code></td>
<td>This function takes an input and converts it into a string.</td>
</tr>
<tr>
<td><code>int()</code></td>
<td>Return an integer object constructed from a number or string.</td>
</tr>
<tr>
<td><code>isinstance()</code></td>
<td>Return True if the object argument is an instance of an object.</td>
</tr>
<tr>
<td><code>issubclass()</code></td>
<td>Return True if class is a subclass of classinfo.</td>
</tr>
<tr>
<td><code>iter()</code></td>
<td>Return an iterator object.</td>
</tr>
<tr>
<td><code>len()</code></td>
<td>Return the length (the number of items) of an object.</td>
</tr>
<tr>
<td><code>list()</code></td>
<td>Rather than being a function, list is a mutable sequence type.</td>
</tr>
<tr>
<td><code>locals()</code></td>
<td>Update and return a dictionary with the current local symbol table.</td>
</tr>
<tr>
<td><code>map()</code></td>
<td>Return an iterator that applies function to every item of iterable.</td>
</tr>
<tr>
<td><code>max()</code></td>
<td>Return the largest item in an iterable.</td>
</tr>
<tr>
<td><code>min()</code></td>
<td>Return the smallest item in an iterable.</td>
</tr>
<tr>
<td><code>next()</code></td>
<td>Retrieve the next item from the iterator.</td>
</tr>
<tr>
<td><code>object()</code></td>
<td>Return a new featureless object.</td>
</tr>
<tr>
<td><code>oct()</code></td>
<td>Convert an integer number to an octal string.</td>
</tr>
</tbody>
</table>
open()  Open file and return a corresponding file object.
ord()   Return an integer representing the Unicode code point of a character.
pow()   Return base to the power exp.
print() Print objects to the text stream file.
property() Return a property attribute.
repr()  Return a string containing a printable representation of an object.
reversed() Return a reverse iterator.
round() Return number rounded to ndigits precision after the decimal point.
set()   Return a new set object.
setattr() This is the counterpart of getattr().
slice() Return a sliced object representing a set of indices.
sorted() Return a new sorted list from the items in iterable.
staticmethod() Transform a method into a static method.
str()   Return a str version of object.
sum()   Sums start and the items of an iterable.
super() Return a proxy object that delegates method calls to a parent or sibling.
tuple() Rather than being a function, is actually an immutable sequence type.
type()  Return the type of an object.
vars()  Return the dict attribute for any other object with a dict attribute.
zip()   Iterate over several iterables in parallel.
import() This function is invoked by the import statement.
More resources: https://www.pythoncheatsheet.org/
Adapted from: https://www.pythoncheatsheet.org/