

Variables and Strings

Printing

What You Will Learn

- Variables
- Strings
- String formatting
- Built-in functions
- Introduction to OOP
- Methods

Variables

- Variables are:
 - storage locations that have a name
 - name-value pairs

```
fruit = 'apple'
```

```
fruit = 'orange'
```

Variables

- Case sensitive. (Case matters!)
 - Fruit and fruit are different variables.
- Must start with a letter.
 - Can *contain* numbers.
- Underscores allowed in variable names
- Not allowed:
 - +
 - -

Valid Variable Names

```
first3letters = 'ABC'
```

```
first_three_letters = 'ABC'
```

```
firstThreeLetters = 'ABC'
```

Strings

- Represent text
- Surrounded by quotes

```
fruit = 'apple'
```

```
fruit = "apple"
```

<https://t.me/learningnets>

Using Quotes within Strings

```
sentence = 'She said, "That is a great tasting apple!'"
```

```
sentence = "That's a great tasting apple!"
```

Using Quotes within Strings

```
double = "She said, \"That's a great tasting apple!\""
```

```
single = 'She said, "That\'s a great tasting apple!"'
```

Indexing

```
String:   a p p l e  
Index:   0 1 2 3 4
```

```
a = 'apple'[0]
```

```
e = 'apple'[4]
```

```
fruit = 'apple'
```

```
first_character = fruit[0]
```

Functions

- A function is a section of reusable code that performs an action.
- A function has a name and is called, or executed, by that name.
- Optionally, functions can accept arguments and return data.

The print() Function

```
fruit = 'apple'  
print(fruit)  
print('orange')
```

```
apple  
orange
```

The len() Function

```
fruit = 'apple'  
fruit_len = len(fruit)  
print(fruit_len)
```

5

Nesting Functions

```
fruit = 'apple'  
print(len(fruit))
```

5

Nesting Functions

```
print(len('apple'))
```

5

String Methods

Basic OOP

- Everything in Python is an object.
- Every object has a type.
- 'apple' is an object of type "str".
- 'apple' is a string object.
- fruit = 'apple'.
 - fruit is a string object.
- Methods are functions run against an object.
 - `object.method()`

The lower() String Method

```
fruit = 'Apple'  
print(fruit.lower())
```

```
apple
```

The upper() String Method

```
fruit = 'Apple'  
print(fruit.upper())
```

APPLE

String Concatenation

String Concatenation

```
print('I ' + 'love ' + 'Python.')
```

```
print('I' + ' love' + ' Python.')
```

I love Python.

I love Python.

String Concatenation

```
print('I' + 'love' + 'Python.')
```

```
IlovePython.
```

String Concatenation

```
first = 'I'  
second = 'love'  
third = 'Python'  
sentence = first + ' ' + second + ' ' +  
third + '.'  
print(sentence)
```

I love Python.

Repeating Strings

```
print('-' * 10)
```

Repeating Strings

```
happiness = 'happy' * 3  
print(happiness)
```

```
happy happy happy
```

The str() Function

```
version = 3  
print('I love Python ' + str(version) + '.')
```

```
I love Python 3.
```

The str() Function

```
version = 3
print('I love Python ' + version + '.')
```

```
File "string_example.py", line 2, in <module>
    print('I love Python ' + version)
TypeError: Can't convert 'int' object to str implicitly
```

Formatting Strings

Formatting Strings

```
print('I {} Python.'.format('love'))  
print('{} {} {}'.format('I', 'love', 'Python.'))
```

I love Python.

I love Python.

Formatting Strings

```
print('I {0} {1}. {1} {0}s me.'.format('love', 'Python'))
```

I love Python. Python loves me.

Formatting Strings

```
first = 'I'  
second = 'love'  
third = 'Python'  
print('{} {} {}'.format(first, second, third))
```

I love Python.

Formatting Strings

```
version = 3  
print('I love Python {}'.format(version))
```

```
I love Python 3.
```

Formatting Strings

```
print('{0:8} | {1:8}'.format('Fruit', 'Quantity'))  
print('{0:8} | {1:8}'.format('Apple', 3))  
print('{0:8} | {1:8}'.f
```

```
Fruit      | Quantity  
Apple      |          3  
Oranges    |          10
```

Formatting Strings

```
print('{0:8} | {1:<8}'.format('Fruit', 'Quantity'))  
print('{0:8} | {1:<8}'.format('Apple', 3))  
print('{0:8} | {1:<8}'.format('Oranges', 10))
```

```
Fruit      | Quantity  
Apple      | 3  
Oranges    | 10
```

Formatting Strings

```
print('{0:8} | {1:<8}'.format('Fruit', 'Quantity'))  
print('{0:8} | {1:<8.2f}'.format('Apple', 2.33333))  
print('{0:8} | {1:<8.2f}'.format('Oranges', 10))
```

```
Fruit      | Quantity  
Apple     | 2.33  
Oranges   | 10.00
```

Formatting Strings Alignment

< Left

^ Center

> Right

Formatting Strings - Data Types

`f` Float

`.Nf` N = The number of decimal places

Example:

```
{ : .2f }
```

Getting User Input

`input()` Accepts Standard Input

`input('Prompt to display')`

Getting User Input

```
fruit = input('Enter a name of a fruit: ')  
print('{} is a lovely fruit.'.format(fruit))
```

```
Name a fruit: apple  
apple is a lovely fruit.
```

Summary

- Variables are names that store values.
- Variables must start with a letter, but may contain numbers and underscores.
- Assign values to variables using the `variable_name = value` syntax.

Summary

- Strings are surrounded by quotation marks.
- Each character in a string is assigned an index.
- A function is reusable code that performs an action.

Summary

- Built-in functions:
 - `print()`: Displays values.
 - `len()`: Returns the length of an item.
 - `str()`: Returns a string object.
 - `input()`: Reads a string.

Summary

- Everything in Python is an object.
- Objects can have methods.
- Methods are functions that operate on an object.

Summary

- String methods:
 - `upper()`: Returns a copy of the string in uppercase.
 - `lower()`: Returns a copy of the string in lowercase.
 - `format()`: Returns a formatted version of the string.

Practice Exercise Solution