Declaring Variables

Declaring Variables

How to declare, name, and update variables in Python

Declaring Variables

Declaring Variables

Actual footage of a real-life variable being declared (colorized):

Declaring Variables

Actual footage of a real-life variable being declared (colorized):

```
>>> x = 1
```

Declaring Variables

Actual footage of a real-life variable being declared (colorized):

```
>>> x = 1
>>>
```

Declaring Variables

Actual footage of a real-life variable being declared (colorized):

```
>>> x = 1
>>>
```

Memory:

Address	Contents
00000000	01101011
00000001	11001100
00000010	10001000
00000011	10101110
⋮	⋮
11111111	00100000

Declaring Variables

Actual footage of a real-life variable being declared (colorized):

```
>>> x = 1
>>>
```

Memory:

$\rightarrow$

Address	Contents
00000000	01101011
00000001	11001100
00000010	10001000
00000011	10101110
⋮	⋮
11111111	00100000

Declaring Variables

Actual footage of a real-life variable being declared (colorized):

```
>>> x = 1
>>>
```

$\rightarrow$

Memory:

Address	Contents
00000000	01101011
00000001	11001100
00000010	10001000
00000011	10101110
⋮	⋮
11111111	00100000

Declaring Variables

Actual footage of a real-life variable being declared (colorized):

```
>>> x = 1
>>>
```

$\rightarrow$

Memory:

Address	Contents
00000000	01101011
00000001	11001100
x	**00000001**
00000011	10101110
⋮	⋮
11111111	00100000

Updating Variables

Actual footage of a real-life variable being updated (colorized):

```
>>> x = 1
>>>
```

Memory:

Address	Contents
00000000	01101011
00000001	11001100
x	**00000001**
00000011	10101110
⋮	⋮
11111111	00100000

Updating Variables

Actual footage of a real-life variable being updated (colorized):

```
>>> x = 1
>>> x = 2
```

Memory:

Address	Contents
00000000	01101011
00000001	11001100
x	**00000001**
00000011	10101110
⋮	⋮
11111111	00100000

Updating Variables

Actual footage of a real-life variable being updated (colorized):

```
>>> x = 1
>>> x = 2
```

`Memory:`

Address	Contents
00000000	01101011
00000001	11001100
x	**00000001**
00000011	10101110
⋮	⋮
11111111	00100000

Updating Variables

Actual footage of a real-life variable being updated (colorized):

```
>>> x = 1
>>> x = 2
```

Memory:

Address	Contents
00000000	01101011
00000001	11001100
x	**00000010**
00000011	10101110
⋮	⋮
11111111	00100000

Updating Variables (continued)

Actual footage of a real-life variable being updated (again):

```
>>> x = 1
>>> x = 2
>>>
```

Memory:

Address	Contents
00000000	01101011
00000001	11001100
x	**00000010**
00000011	10101110
⋮	⋮
11111111	00100000

Updating Variables (continued)

Actual footage of a real-life variable being updated (again):

```
>>> x = 1
>>> x = 2
>>> x = x + 1
```

Memory:

Address	Contents
00000000	01101011
00000001	11001100
x	**00000010**
00000011	10101110
⋮	⋮
11111111	00100000

Updating Variables (continued)

Actual footage of a real-life variable being updated (again):

```
>>> x = 1
>>> x = 2
>>> x = x + 1
```

`Memory:`

Address	Contents
00000000	01101011
00000001	11001100
x	**00000010**
00000011	10101110
⋮	⋮
11111111	00100000

Updating Variables (continued)

Actual footage of a real-life variable being updated (again):

```
>>> x = 1
>>> x = 2
>>> x = x + 1
```

Memory:

Address	Contents
00000000	01101011
00000001	11001100
x	**00000011**
00000011	10101110
⋮	⋮
11111111	00100000

Garbage Collection

Actual footage of a real-life variable being released:

```
>>> x = 1
>>> x = 2
>>> x = x + 1
>>>
```

Memory:

Address	Contents
00000000	01101011
00000001	11001100
x	**00000011**
00000011	10101110
⋮	⋮
11111111	00100000

Garbage Collection

Actual footage of a real-life variable being released:

```
>>> x = 1
>>> x = 2
>>> x = x + 1
>>> quit()
```

Memory:

Address	Contents
00000000	01101011
00000001	11001100
x	**00000011**
00000011	10101110
⋮	⋮
11111111	00100000

Garbage Collection

Actual footage of a real-life variable being released:

```
>>> x = 1
>>> x = 2
>>> x = x + 1
>>> quit()
```

`Memory:`

Address	Contents
00000000	01101011
00000001	11001100
x	**00000011**
00000011	10101110
⋮	⋮
11111111	00100000

Garbage Collection

Actual footage of a real-life variable being released:

```
>>> x = 1
>>> x = 2
>>> x = x + 1
>>> quit()
```

Memory:

Address	Contents
00000000	01101011
00000001	11001100
00000010	00000011
00000011	10101110
⋮	⋮
11111111	00100000

How to Name Your Variables

How to Name Your Variables

Things that are allowed:

How to Name Your Variables

Things that are allowed:

- Lower-case letters

How to Name Your Variables

Things that are allowed:

- Lower-case letters
- Upper-case letters

How to Name Your Variables

Things that are allowed:

- Lower-case letters
- Upper-case letters
- Numbers (but not the first character)

How to Name Your Variables

Things that are allowed:

- Lower-case letters
- Upper-case letters
- Numbers (but not the first character)
- Underscores

How to Name Your Variables

Things that are allowed:

- Lower-case letters
- Upper-case letters
- Numbers (but not the first character)
- Underscores

Warning! We are not the same!

`my_var` `MY_VAR`

How to Name Your Variables

Things that are allowed:

- Lower-case letters
- Upper-case letters
- Numbers (but not the first character)
- Underscores

Warning! We are not the same!

`my_var` `MY_VAR`

Python is case-sensitive

How to Name Your Variables

How to Name Your Variables

Things that are not allowed:

How to Name Your Variables

Things that are not allowed:

- Special characters. Ex: `#`, `$`, `%`, `*`, `-`, `=`, `+`, `?`, `.`

How to Name Your Variables

Things that are not allowed:

- Special characters. Ex: `#`, `$`, `%`, `*`, `-`, `=`, `+`, `?`, `.`
- Spaces (Python is whitespace-sensitive)

How to Name Your Variables

Things that are not allowed:

- Special characters. Ex: `#`, `$`, `%`, `*`, `-`, `=`, `+`, `?`, `.`
- Spaces (Python is whitespace-sensitive)
- Names that start with numbers. Ex: `x1` is allowed, but `1x` is not

Concept Check!

Which of the following are valid ways to declare a variable?

1. `>>> x = 3`
2. `>>> 3 = x`
3. `>>> tyler = 3.0`
4. `>>> __name__ = "tyler"`
5. `>>> my-name = "tyler"`
6. `>>> mySecondName = "chang"`
7. `>>> my2name = "chang"`
8. `>>> 2name = "chang"`
9. `>>> my_second_name = "chang"`
10. `>>> myname#2 = "chang"`

Solutions

Which of the following are valid ways to declare a variable?

Solutions

Which of the following are valid ways to declare a variable?

1. `>>> x = 3` ✓

Solutions

Which of the following are valid ways to declare a variable?

1. `>>> x = 3` ✓
2. `>>> 3 = x` ✗

Solutions

Which of the following are valid ways to declare a variable?

1. `>>> x = 3` ✓
2. `>>> 3 = x` ✗
3. `>>> num = 3.0` ✓

Solutions

Which of the following are valid ways to declare a variable?

1. `>>> x = 3` ✓
2. `>>> 3 = x` ✗
3. `>>> num = 3.0` ✓
4. `>>> __name__ = "tyler"` ✓

Solutions

Which of the following are valid ways to declare a variable?

1. `>>> x = 3` ✓
2. `>>> 3 = x` ✗
3. `>>> num = 3.0` ✓
4. `>>> __name__ = "tyler"` ✓
5. `>>> my-name = "tyler"` ✗

Solutions

Which of the following are valid ways to declare a variable?

1. `>>> x = 3` ✓
2. `>>> 3 = x` ✗
3. `>>> num = 3.0` ✓
4. `>>> __name__ = "tyler"` ✓
5. `>>> my-name = "tyler"` ✗
6. `>>> mySecondName = "chang"` ✓

Solutions

Which of the following are valid ways to declare a variable?

1. `>>> x = 3` ✓
2. `>>> 3 = x` ✗
3. `>>> num = 3.0` ✓
4. `>>> __name__ = "tyler"` ✓
5. `>>> my-name = "tyler"` ✗
6. `>>> mySecondName = "chang"` ✓
7. `>>> my2name = "chang"` ✓

Solutions

Which of the following are valid ways to declare a variable?

1. `>>> x = 3` ✓
2. `>>> 3 = x` ✗
3. `>>> num = 3.0` ✓
4. `>>> __name__ = "tyler"` ✓
5. `>>> my-name = "tyler"` ✗
6. `>>> mySecondName = "chang"` ✓
7. `>>> my2name = "chang"` ✓
8. `>>> 2name = "chang"` ✗

Solutions

Which of the following are valid ways to declare a variable?

1. `>>> x = 3` ✓
2. `>>> 3 = x` ✗
3. `>>> num = 3.0` ✓
4. `>>> __name__ = "tyler"` ✓
5. `>>> my-name = "tyler"` ✗
6. `>>> mySecondName = "chang"` ✓
7. `>>> my2name = "chang"` ✓
8. `>>> 2name = "chang"` ✗
9. `>>> my_second_name = "chang"` ✓

Solutions

Which of the following are valid ways to declare a variable?

1. `>>> x = 3` ✓
2. `>>> 3 = x` ✗
3. `>>> num = 3.0` ✓
4. `>>> __name__ = "tyler"` ✓
5. `>>> my-name = "tyler"` ✗
6. `>>> mySecondName = "chang"` ✓
7. `>>> my2name = "chang"` ✓
8. `>>> 2name = "chang"` ✗
9. `>>> my_second_name = "chang"` ✓
10. `>>> myname#2 = "chang"` ✗

Picking a Good Name

Picking a Good Name

Some general rules for naming your variables:

Picking a Good Name

Some general rules for naming your variables:

- Name should be descriptive of what the variable does

Picking a Good Name

Some general rules for naming your variables:

- Name should be descriptive of what the variable does
- Not too long

Picking a Good Name

Some general rules for naming your variables:

- Name should be descriptive of what the variable does
- Not too long
- Don't re-use variable names in a confusing way

Picking a Good Name

Some general rules for naming your variables:

- Name should be descriptive of what the variable does
- Not too long
- Don't re-use variable names in a confusing way

Suggestions for handling multiple word names:

Picking a Good Name

Some general rules for naming your variables:

- Name should be descriptive of what the variable does
- Not too long
- Don't re-use variable names in a confusing way

Suggestions for handling multiple word names:

- `writeInCamelCase`

Picking a Good Name

Some general rules for naming your variables:

- Name should be descriptive of what the variable does
- Not too long
- Don't re-use variable names in a confusing way

Suggestions for handling multiple word names:

- `writeInCamelCase`
- `use_underscores_to_separate_words`

Concept Check!

How would you name the following variables?

- A variable that you will use to count the number of times something happens
- A variable that you will use to store the average temperature in January
- A variable that you will use to store the user's last name
- A variable that you will use to store the user's personal ID

Named constants

Challenge: you want to give a name to some immediate value that you will use a lot in your code, such as Pi.

Named constants

Challenge: you want to give a name to some immediate value that you will use a lot in your code, such as Pi.

What do you do?

Named constants

Challenge: you want to give a name to some immediate value that you will use a lot in your code, such as Pi.

What do you do?

Declare a named constant!

```
>>> PI = 3.14159265358979
```

Now you can use it in your code, without having to constantly type-out the long number.

Named constants

A name constant is a variable that contains some pre-determined value, which you don't intend to change in the future.

Named constants

A name constant is a variable that contains some pre-determined value, which you don't intend to change in the future.

Python doesn't explicitly support named constants, but that doesn't stop you from creating a variable and never changing its value!

Named constants

A name constant is a variable that contains some pre-determined value, which you don't intend to change in the future.

Python doesn't explicitly support named constants, but that doesn't stop you from creating a variable and never changing its value!

When using named constants, follow these rules so other programmers know:

Named constants

A name constant is a variable that contains some pre-determined value, which you don't intend to change in the future.

Python doesn't explicitly support named constants, but that doesn't stop you from creating a variable and never changing its value!

When using named constants, follow these rules so other programmers know:

- define the named constant at the top of your script, before getting any input from the user

Named constants

A name constant is a variable that contains some pre-determined value, which you don't intend to change in the future.

Python doesn't explicitly support named constants, but that doesn't stop you from creating a variable and never changing its value!

When using named constants, follow these rules so other programmers know:

- define the named constant at the top of your script, before getting any input from the user
- `GIVE_YOUR_CONSTANT_A_NAME_IN_ALL_CAPS`

Concept Check!

Change the `temperatures.py` script from Lecture 5 to use good variable names and named constants