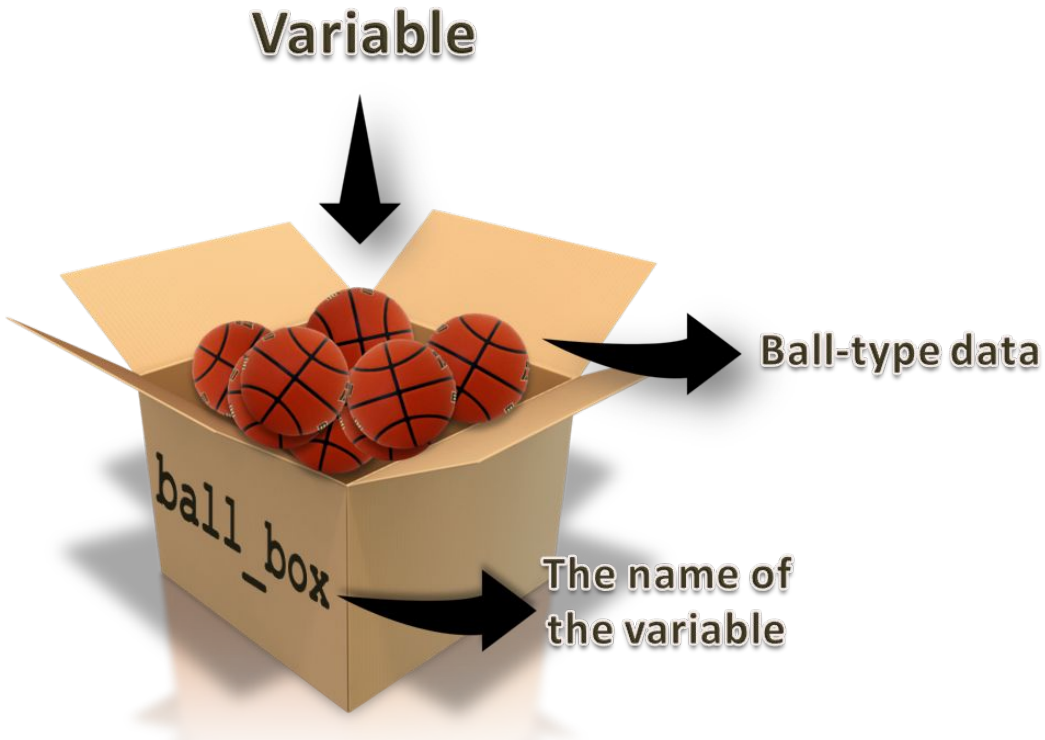
Table of Contents

Variables

Variables : refreshing



Naming Variables

Table of Contents

General Description

General Description

I didn't understand anything.

Con_PEP_N_Rl

Con_PEP_N_Rl

bad example of naming a variable

Conventional (PEP 8) Naming Rules

Conventional (PEP 8) Naming Rules

- **Some basic naming conventions**

  - Choose **lowercase** words and use underscore to split the words

```
variable = 3.14
var_one = 'something'
```

Conventional (PEP 8) Naming Rules

▶ Some basic naming conventions

- Do not use `'l'` **(lowercase letter "L")** as single character variable.

```
l = 3.14  # This is lowercase letter el
I = 3.14  # This is uppercase letter eye
1 = 'something wrong'  # This is number one.
```

Conventional (PEP 8) Naming Rules

- **Some basic naming conventions**

  - Do not use `'O'` **(uppercase letter “O”)** as single character variable.

```
time_O = '3.14'  # This is uppercase letter “O”
time_0 = '3.14'  # This is number zero
```

Some Important PEP 8 Rules

- **Some basic naming conventions**
  - The name of the variable must be legible, meaningful and relevant to the type of value

Good

```
students = ...
```

```
# Big Data
big_data = ...
```

```
# Average income of February
avg_income_feb = ...
```

Bad

```
s = ...  or  st = ...
```

```
# Big Data
b_dt = ...
```

```
# Average income of February
average_income_february = ..
```

Some Important PEP 8 Rules

- **Some basic naming conventions (reserved words)**

- Do not use specific Python keywords such as :

```
False   class     finally   is        return
None    continue  for       lambda    try
True    def       from      nonlocal  while
and     del       global    not       with
as      elif      if        or        yield
assert  else      import    pass      break
except  in        raise
```

Pythonic Rules

Examine these samples carefully

`2me`	👉 😔	`data4me`	👉 😀
`Big boss`	👉 😔	`first!`	👉 😔
`big-boss`	👉 😔	`xy#@!v`	👉 😔
`last_name`	👉 😀	`$price`	👉 😔

Naming Variable

```
amount of rotten fruits
```

```
the list of prime numbers
```

```
the list of mathematics exam scores
```

What can be the **Name** of these sentences in terms of variables.

Naming variable

```
amount of rotten fruits
```

- **Good samples** :
  - `rotten_fruits = 33  # kg.`
  - `amount_rotten_fruits = 33  # kg.`

Naming variable

```
the list of prime numbers
```

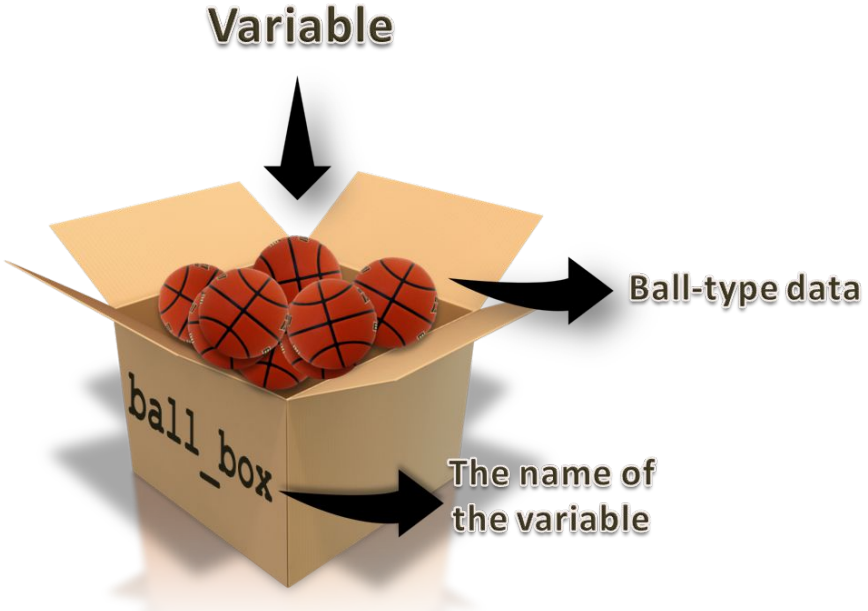
- **Good samples** :
  - prime_list
  - prime_no
  - list_prime
  - num_prime

Naming variable

```
the list of mathematics exam scores
```

- **Good samples :**
  - `math_scores`
  - `score_maths`

Variables



Variables

```
variable_name = value
```

```
planet = 'Saturn'
price = 140
pi_number = 3.14
```

The declaration happens automatically when you assign a value to a variable.

Variables

```
my_age = 33
your_age = 30
my_age = your_age
print(my_age)
```

Variables

```
my_age = 33
your_age = 30
my_age = your_age
print(my_age)
```

```
30
```

Variables

Assigning a value to a variable.

```
x = y = z = "same"

print(x)
print(y)
print(z)
```

Variables

The output

```
same
same
same
```

Write a Python code:

Hint : Use `int` *value of 30, 31, 28 only once.*

Which months have **31** days and which have **30** or **28**? Let's assign the number of days (30 or 31 or 28) to the months (the variables will be the name of the months) in totally ***three*** code ***lines*** then print their number of days in order of the months as follows.

```
print(january, february, march, april, may, june, july, august, september, october,
november, december)
```

```
31 28 31 30 31 30 31 31 30 31 30 31
```

Variables

⚠️ A probable answer :

```
january = march = may = july = august = october = december = 31
# multi assignments in a single line

april = june = september = november = 30

february = 28

print(january, february, march, april, may, june, july, august, september, october,
november, december)
```

If we don’t know the value of a variable, what can we assign to it? For example :

The ages of instructors :

```
thomas = 33
marry = 28
walter = ?
isabella = 46
```

Variables

```
thomas = 33
marry = 28
walter = None
isabella = 46
```

Variables

 Assigning a value to a variable.

```
website = "apple.com"
print(website)

# assigning a new variable to website
website = "clarusway.com"

print(website)
```

Variables

The output

```
apple.com
clarusway.com
```

Variables

 Assigning a value to a variable.

```
first_number = 100
second_number = first_number
print(second_number)
```

Variables

The output

```
100
```

Variables

Assigning a value to a variable.

```
x = 15
y = 33
z = x
x = y

print(x)
print(y)
print(z)
```

Variables

💡 The output

```
x = 33
y = 33
z = 15
```

Variables

Assigning a value to a variable.

```
a, b, c = 5, 3.2, "Hello"

print(a)
print(b)
print(c)
```

Variables

The output

```
5
3.2
Hello
```

Variables

- Pay attention to the value of variables and how they change.

```
man = "andrew"
color = "green"
age = 32
pi = 3.14
color = "yellow"
age = 31
man = "Aslan"

print(man, age, color)
```

Output

```
Aslan 31 yellow
```

Basic Data Types

Introduction to Data Types

Introduction to Data Types

- Each data has a type.

- This type of data defines how you store it in memory and it also describes which process can be applied to it.

Introduction to Data Types

- **Some simple data types commonly used in Python:**
  - String,
  - Integer,
  - Float,
  - Boolean.

Strings

"2022"

str

"string is the most used type"

"i have 3 lb. of apple"

Strings

- If you want to work with any **textual** characters in your code, you have to work with strings.

```
my_text = 'being a good person'
print(my_text)
```

String type is called `str`.

```
type(variable)
```

Strings

- If you want to work with any **textual** characters in your code, you have to work with strings.

```
my_text = 'being a good person'
print(my_text)
```

```
being a good person
```

String type is called `str`.

- Strings are identified as a set of characters represented in the single or double quotes.

```
type(variable)
```

Write down the followings on your Playground as `str` type and then print them...

- your mail address
- 632
- It's okay!

Strings

- **Let's** do some practices which cover string type.

```
1  str_number = '1923'
2  str_sign = '%(#&*?-'
3
4
5  print(str_number)
6  print(str_sign)
7  print(type(str_number), type(str_sign))
8
9
```

Strings

- **Let's** do some practices which cover string type.

```
1  str_number = '1923'
2  str_sign = '%(#&*?-'
3
4
5  print(str_number)
6  print(str_sign)
7  print(type(str_number), type(str_sign))
8
9
```

Output

```
1923
%(#&*?-
<class 'str'> <class 'str'>
```

Numeric Types `int` `float`

Numeric Types

- Three basic numeric types in Python :

  - Integers
  - Floats
  - Complexes

Numeric Types

- **Integer** types are whole numbers which don't contain decimal point.

```
my_integer = 40
negative_num = -18

print(my_integer)
print(negative_num)
```

Signed integer type is called `int`.

Numeric Types

- **Integer** types are whole numbers which don't contain decimal point.

```
my_integer = 40
negative_num = -18

print(my_integer)
print(negative_num)
```

```
40
-18
```

Signed integer type is called `int`.

Numeric Types

- **Float** types stand for real numbers with a decimal point.

```
my_float = 40.0
negative_float = -18.66

print(my_float)
print(negative_float)
```

Floating point type is called `float`.

Numeric Types

- **Float** types stand for real numbers with a decimal point.

```
my_float = 40.0
negative_float = -18.66

print(my_float)
print(negative_float)
```

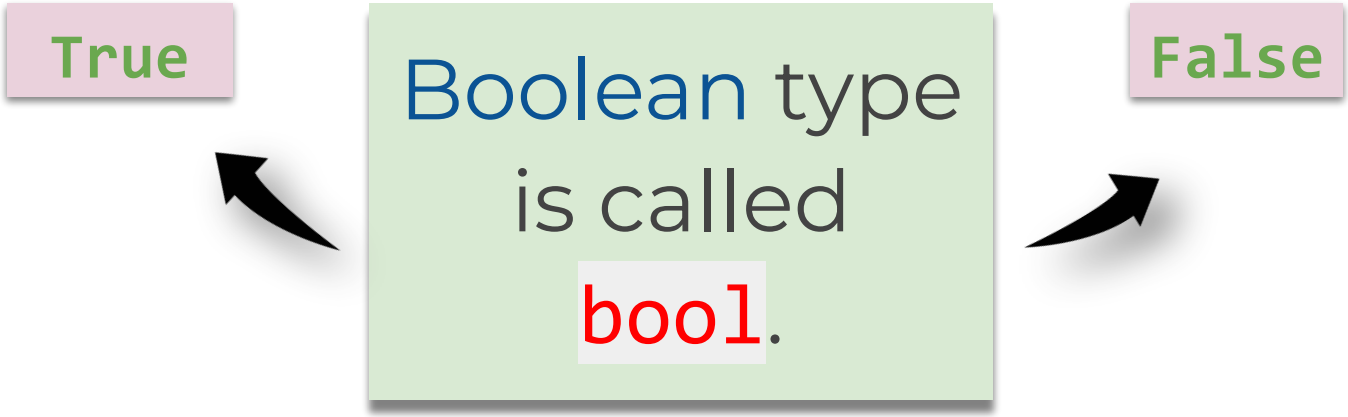
```
40.0
-18.66
```

Floating point type is called `float`.

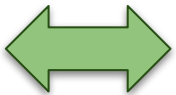
Boolean

Boolean

- Boolean types' values are the two constant objects **False** and **True**.



Type Conversion

`int`  `str`

Type Conversion

We can print the types of data using `type()` function

```
my_data = 'I am string'
print(type(my_data))
```

Type Conversion

We can print the types of data using `type()` function

```
my_data = 'I am string'
print(type(my_data))
```

```
<class 'str'>
```

Type Conversion

Type conversion functions.

`str()` converts to string type

`int()` converts to signed integer type

`float()` converts to floating point type

`bool()` converts to boolean value type

The value of any type in Python can be converted to a `str`.

True

False

Type Conversion

Converting `float` to `str`

```
pi = 3.14

converted_pi = str(pi)
print(converted_pi)
print(type(converted_pi))
```

Type Conversion

Converting `float` to `str`

```
pi = 3.14

converted_pi = str(pi)
print(converted_pi)
print(type(converted_pi))
```

```
3.14
<class 'str'>
```

Type Conversion

Converting `float` to `int`

```
pi = 3.14

converted_pi = int(pi)
print(converted_pi)
print(type(converted_pi))
```

Type Conversion

Converting `float` to `int`

```
pi = 3.14

converted_pi = int(pi)
print(converted_pi)
print(type(converted_pi))
```

```
3
<class 'int'>
```

Type Conversion

Converting `int` to `float`

```
no = 3

converted_no = float(no)
print(converted_no)
print(type(converted_no))
```

Type Conversion

Converting `int` to `float`

```
no = 3

converted_no = float(no)
print(converted_no)
print(type(converted_no))
```

```
3.0
<class 'float'>
```

Type Conversion

input :

```
1  x = 39
2  v = "11"
3  y = "2.5"
4  z = "I am at_"
5
6  print(x-int(v))
7  print(x-float(y))
8  print(z+str(x))
9
```

Type Conversion

input :

```
1  x = 39
2  v = "11"
3  y = "2.5"
4  z = "I am at_"
5
6  print(x-int(v))
7  print(x-float(y))
8  print(z+str(x))
9
```

x-int("11") = 39-11 = **28**

output :

```
1  28
2  36.5
3  I am at_39
4
5
```

Type Conversion

input :

```
1  x = 39
2  v = "11"
3  y = "2.5"
4  z = "I am at_"
5
6  print(x-int(v))
7  print(x-float(y))
8  print(z+str(x))
9
```

x-int("11") = 39-11 = **28**

x-float("2.5") = 39-2.5 = **36.5**

output :

```
1  28
2  36.5
3  I am at_39
4
5
```

Type Conversion

input :

```
1  x = 39
2  v = "11"
3  y = "2.5"
4  z = "I am at_"
5
6  print(x-int(v))
7  print(x-float(y))
8  print(z+str(x))
9
```

x-int("11") = 39-11 = **28**

x-float("2.5") = 39-2.5 = **36.5**

z+str(39) = "I am at_" + "39" = **I am at_39**

output :

```
1  28
2  36.5
3  I am at_39
4
5
```

Type Conversion, Quiz

Without using any Interpreter/IDLE, just try to guess the output.

```
number_int = 123
number_flt = 1.23

number_new = number_int + number_flt

print("datatype of number_int:", type(number_int))
print("datatype of number_flt:", type(number_flt))

print("Value of number_new:", number_new)
print("datatype of number_new:", type(number_new))
```

Type Conversion, Quiz

Without using any Interpreter/IDLE, just try to guess the output.

```
number_int = 123
number_str = "456"

print("Data type of number_int:", type(number_int))
print("Data type of number_str:", type(number_str))

print(number_int + number_str)
```

Type Conversion, Quiz

💡 Without using any Interpreter/IDLE, just try to guess the output.

```
number_int = 123
number_str = "456"

print("Data type of number_int:", type(number_int))
print("Data type of number_str before Type Casting:", type(number_str))

number_str = int(number_str)
print("Data type of number_str after Type Casting:", type(number_str))

number_sum = number_int + number_str

print("Sum of number_int and number_str:", number_sum)
print("Data type of the sum:", type(number_sum))
```