Python Coding Schools

3rd lesson

Seed Academy

Agenda

- wk1. Installing Python, HelloWorld
- wk2. Arithmetic Operators
- wk3. Data Types : Integer, Floating point, Boolean, String
- wk4. Data Structures: List
- wk5. Data Structures: Set, Tuples
- wk6. Data Structures: Dictionary

Agenda

- wk7. Control flows
- wk8. Conditional
- wk9. Loops
- wk10. Function
- wk11. Class
- wk12. Data Visualization

Class materials

https://github.com/TaeheeJeong/seedacademy

https://github.com/TaeheeJeong/SummerCoding2023

Today’s topic: Data type

- Integer
- Floating point
- Strings
- Booleans

What Does “Type” Mean?

- In Python, variables and constants have a type
- Python knows the difference between an integer number and a string
- For example “+” means:
  - addition if something is a number
  - concatenate if something is a string

```
>>> ddd = 1 + 4
>>> print(ddd)
5
>>> eee = 'hello ' + 'there'
>>> print(eee)
hello there
```

Source: Python for Everybody, Charles Severance, University of Michigan School of Information

Type Matters

- Python knows what “type” everything is
- Some operations are prohibited
- You cannot “add 1” to a string
- We can ask Python what type something is by using the type() function

```
>>> eee = 'hello ' + 'there'
>>> eee = eee + 1
Traceback (most recent call last):  File
"<stdin>", line 1, in <module>TypeError:
Can't convert 'int' object to str
implicitly
>>> type(eee)
<class'str'>
>>> type('hello')
<class'str'>
>>> type(1)
<class'int’>
```

Source: Python for Everybody, Charles Severance, University of Michigan School of Information

Several Types of Numbers

- Numbers have two main types

  - Integers are whole numbers:
    -14, -2, 0, 1, 100, 401233

  - Floating Point Numbers have decimal parts:
  -2.5 , 0.0, 98.6, 14.0

- There are other number types:

  - they are variations on float and integer

```
>>> xx = 1
>>> type (xx)
<class 'int'>
>>> temp = 98.6
>>> type(temp)
<class'float'>
>>> type(1)
<class 'int'>
>>> type(1.0)
<class'float'>
```

Source: Python for Everybody, Charles Severance, University of Michigan School of Information

Type Conversions

- When you put an integer and floating point in an expression, the integer is implicitly converted to a float
- You can control this with the built-in functions int() and float()

```
>>> print(float(99) + 100)
199.0
>>> i = 42
>>> type(i)
<class'int'>
>>> f = float(i)
>>> print(f)
42.0
>>> type(f)
<class'float'>
```

Source: Python for Everybody, Charles Severance, University of Michigan School of Information

Integer Division

- Integer division produces a floating point result

```
>>> print(10 / 2)
5.0
>>> print(9 / 2)
4.5
>>> print(99 / 100)
0.99
>>> print(10.0 / 2.0)
5.0
>>> print(99.0 / 100.0)
0.99
```

Source: Python for Everybody, Charles Severance, University of Michigan School of Information

Booleans

- Logical value that indicates True or False
- Important for control flow & logic

Logical Operators (and, or, not)

```
In [15]: 1 True and True
Out[15]: True

In [16]: 1 True and False
Out[16]: False

In [18]: 1 True or False
Out[18]: True

In [19]: 1 False or False
Out[19]: False

In [20]: 1 not True
Out[20]: False

In [21]: 1 not False
Out[21]: True
```

String Conversions

- You can also use int() and float() to convert between strings and integer/float

```
>>> int('3')
>>> 3
>>> float('3')
>>> 3.
>>> str(3)
>>> '3'
```

Source: Python for Everybody, Charles Severance, University of Michigan School of Information

String Conversions

- You can also use int() and float() to convert between strings and integer/float
- You will get an error if the string does not contain numeric characters

```
>>> sval = '123'
>>> type(sval)
<class 'str'>
>>> print(sval + 1)
Traceback (most recent call last):  File
"<stdin>", line 1, in <module>
TypeError: Can't convert 'int' object to str
implicitly
>>> ival = int(sval)
>>> type(ival)
<class 'int'>
>>> print(ival + 1)
124
>>> nsv = 'hello bob'
>>> niv = int(nsv)
Traceback (most recent call last):  File
"<stdin>", line 1, in <module>
ValueError: invalid literal for int() with
base 10: 'x'
```

String Data Type

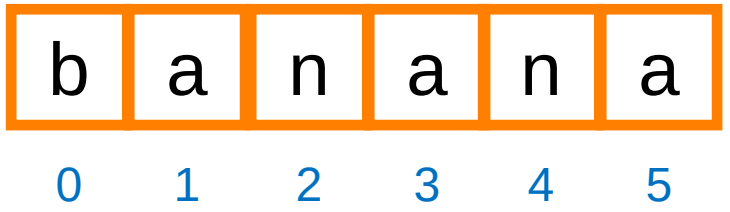
- A string is a sequence of characters
- A string literal uses quotes
  'Hello' or "Hello"
- For strings, + means “concatenate”
- When a string contains numbers, it is still a string
- We can convert numbers into a string using str()

```
>>> str1 = "Hello"
>>> str2 = 'there'
>>> bob = str1 + str2
>>> print(bob)
Hellothere
>>> str3 = '123'
>>> str3 = str3 + 1
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str'
and 'int' objects
>>> x = int(str3) + 1
>>> print(x)
124
>>> x = ‘123’ + str(1)
>>> print(x)
```

Source: Python for Everybody, Charles Severance, University of Michigan School of Information

Looking Inside Strings

- We can get at any single character in a string using an index specified in square brackets
- The index value must be an integer and starts at zero
- The index value can be an expression that is computed




```
>>> fruit = 'banana'
>>> letter = fruit[1]
>>> print(letter)
a
>>> x = 3
>>> w = fruit[x - 1]
>>> print(w)
n
```

Source: Python for Everybody, Charles Severance, University of Michigan School of Information

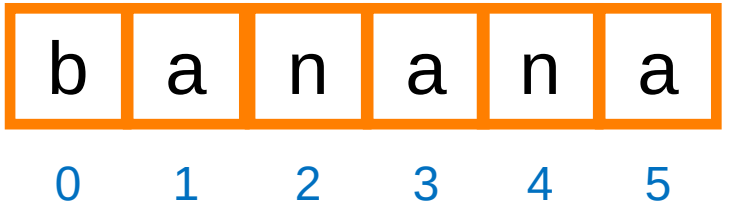
A Character Too Far

- You will get a python error if you attempt to index beyond the end of a string
- So be careful when constructing index values and slices

```
>>> zot = 'abc'
>>> print(zot[5])
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
IndexError: string index out of range
```

Strings Have Length

- The built-in function len gives us the length of a string

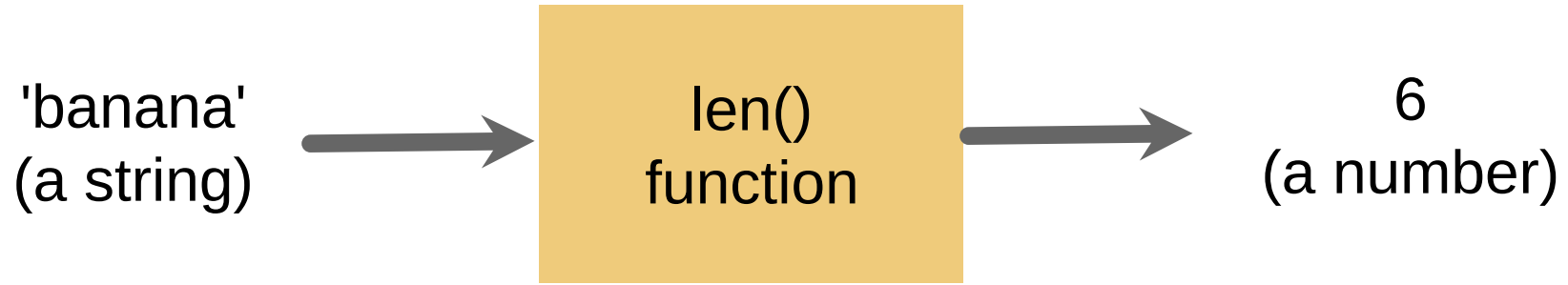



```
>>> fruit = 'banana'
>>> print(len(fruit))
6
```

len Function

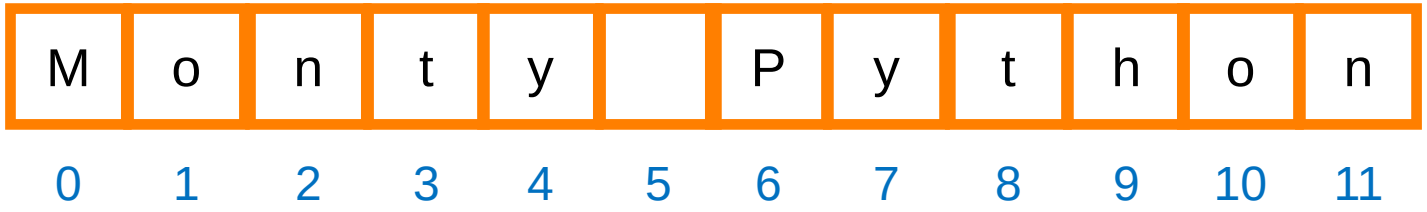
```
>>> fruit = 'banana'
>>> x = len(fruit)
>>> print(x)
6
```

A function is some stored code that we use. A function takes some input and produces an output.



Source: Python for Everybody, Charles Severance, University of Michigan School of Information

Slicing Strings

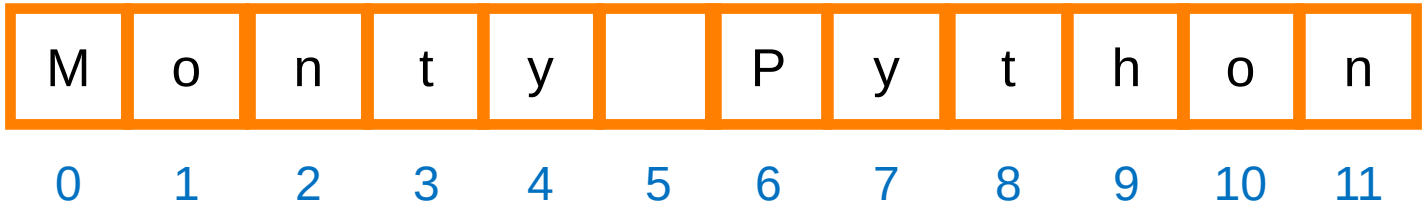


- We can also look at any continuous section of a string using a colon operator
- The second number is one beyond the end of the slice - “up to but not including”
- If the second number is beyond the end of the string, it stops at the end

```
>>> s = 'Monty Python'
>>> print(s[0:4])
Mont
>>> print(s[6:7])
P
>>> print(s[6:20])
Python
```

Source: Python for Everybody, Charles Severance, University of Michigan School of Information

Slicing Strings



If we leave off the first number or the last number of the slice, it is assumed to be the beginning or end of the string respectively

```
>>> s = 'Monty Python'
>>> print(s[:2])
Mo
>>> print(s[8:])
thon
>>> print(s[:])
Monty Python
```

Source: Python for Everybody, Charles Severance, University of Michigan School of Information

String Concatenation

When the + operator is applied to strings, it means “concatenation”

```
>>> a = 'Hello'
>>> b = a + 'There'
>>> print(b)
HelloThere
>>> c = a + ' ' + 'There'
>>> print(c)
Hello There
```

Source: Python for Everybody, Charles Severance, University of Michigan School of Information

Using **in** as a Logical Operator

The in keyword can also be used to check to see if one string is “in” another string

The in expression is a logical expression that returns True or False and can be used in an if statement

```
>>> fruit = 'banana'
>>> 'n' in fruit
True
>>> 'm' in fruit
False
>>> 'nan' in fruit
True
>>> if 'a' in fruit :
...     print('Found it!')
...
Found it!
```

Source: Python for Everybody, Charles Severance, University of Michigan School of Information

Acknowledgements / Contributions

Initial Development: Charles Severance, University of Michigan School of Information

Modification: Taehee Jeong, San Jose State University