06.2_Conditionals_Strings

February 22, 2023

1 Introduction to Python for Open Source Geocomputation

- Instructor: Dr. Wei Kang
- Class Location and Time: ENV 336, Mon & Wed 12:30 pm - 1:50 pm

Content:

- Conditional Execution
- Strings

2 Conditionals with `if` statements

- Give us the ability to check conditions and change the behavior of the program accordingly.
- Check `True` or `False`
- Intense use of logical operations or comparison operations
- One of the five component of a program: input, output, conditions, repetition, math

2.0.1 (1) If statement on its own:

[1]:
```
x = 0
```

[2]:
```
x
```

[2]: 0

[3]:
```
x == 0
```

[3]:
```
True
```

[4]:
```
if x > 0:
    print('x is positive')
```

[5]:
```
x = -1
if x > 0:
    print('x is positive')
```

[6]:
```
x = 1
if x > 0:
    print('x is positive')
```

```
x is positive
```

2.0.2 Syntax of a simple `if` statement: check one condition

```
if x > 0:
    print('x is positive')
```

- `if`: keyword for the Conditional Execution
- `x > 0`: the condition to check (logical/comparison operations)
  - if `True` (boolean value), the block following that condition `print('x is positive')` is executed
  - if `False` (boolean value), the block following that condition `print('x is positive')` is not executed
- colon `:`
- A new line
- indentation: 4 spaces

[7]:
```
if True:
    print("True")
```

```
True
```

[8]:
```
if False:
    print("True")
print("True")
```

```
True
```

[9]:
```
if False:
print("True")
```

```
  File "/var/folders/6m/8n2ktxl566j8yp0n_qx7x5bw0000gt/T/ipykernel_42182/
↪1194173230.py", line 2
    print("True")
    ^
```

```
IndentationError: expected an indented block
```

[10]:
```
a = 1
b = 3

if a > b:
    print('a is bigger than b')
```

[11]:
```
a = 8
b = 3
c = 10

if a > b and c < b:
    print('a is bigger than b and c is smaller than b')
```

[12]:
```
a = 8
b = 3
c = 1

if a > b and c < b:
    print('a is bigger than b and c is smaller than b')
```

```
a is bigger than b and c is smaller than b
```

2.0.3 (2) If-else statement:

[13]:
```
x = 1
if x > 0:
    print('x is positive')
else:
    print('x is zero or negative')
```

```
x is positive
```

[14]:
```
x = 0
if x > 0:
    print('x is positive')
else:
    print('x is zero or negative')
```

```
x is zero or negative
```

[15]:
```
18 % 17
```

[15]: 1

[20]:
```
x = 1547
if x % 17 == 0:
    print('Your number is a multiple of 17.')
else:
    print('Your number is not a multiple of 17.')
```

```
Your number is a multiple of 17.
```

[21]:
```
x = int(input('Insert your number: '))
if x % 17 == 0:
    print('Your number is a multiple of 17.')
else:
    print('Your number is not a multiple of 17.')
```

```
Insert your number: 2
Your number is not a multiple of 17.
```

[22]:
```
# x = input('Insert your number: ')
x = 17.02
if x % 17 == 0:
    print('Your number is a multiple of 17.')
else:
    print('Your number is not a multiple of 17.')
```

```
Your number is not a multiple of 17.
```

2.0.4 Syntax of a `if-else` statement: check one condition and two potential executions

```
if x > 0:
    print('x is positive')
else:
    print('x is not positive')
```

- `if` and `else`: keywords for the Conditional Execution
- `x > 0`: the first condition
  - if `True`, the block following that condition is executed and the else statement is ignored.
  - if `False`, the block following the else statement is executed.

2.0.5 (3) If-elif-else statement:

[23]:
```
a = 3
b = 5

if a > b:
    print('a is bigger than b')
elif a < b:
    print('a is smaller than b')
else:
```

```
    print('a is equal to b')
```

```
a is smaller than b
```

[24]:
```
a = 3
b = 3

if a > b:
    print('a is bigger than b')
elif a < b:
    print('a is smaller than b')
else:
    print('a is equal to b')
```

```
a is equal to b
```

[25]:
```
a == b
```

[25]:
```
True
```

[26]:
```
a = 3
b = 5

if a > b:
    print('a is bigger than b')
elif a > b:
    print('a is bigger than b')
else:
    print('a is not bigger than b')
```

```
a is not bigger than b
```

[27]:
```
a = 3
b = 5

if a > b:
    print('a is bigger than b')
elif a > b:
    print('a is bigger than b')
```

[28]:
```
if a > b:
    print('a is bigger than b')
elif a > b:
    print('a is bigger than b')
```

[29]:
```
a = 100
b = 3
```

2.0.6 Syntax of a `if-elif-else` statement: check more than one conditions

```
if a > b:
    print('a is bigger than b')
elif a < b:
    print('a is smaller than b')
elif a < b:
    print('a is smaller than b')
elif a < b:
    print('a is smaller than b')
else:
    print('a is equal to b')
```

- `if`, `elif`, and `else`: keywords for the Conditional Execution
- `a > b`: the first condition
  - if `True`, the block following that condition is executed and rest is ignored.
  - if `False`, check the second condition after `elif`
- `a < b`: the second condition
  - if `True`, the block following that condition is executed and rest is ignored.
  - if `False`, the else statement is executed.

Conditions do not have to be mutually exclusive, but it makes better sense if they do!

2.0.7 Group Exercise (work in a two (or three-people group) randomly assigned by the instructor)

Using `if`, `elif` and `else` statements write a code where you check whether number a is larger than b and whether c is larger than d, and all the other potential relationships. For instance,

- if number a is larger than b and c is larger than d, the program print "a is larger than b and c is larger than d".
- if number a is smaller than b and c is smaller than d, the program print "a is smaller than b and c is smaller than d".
- if number a is larger than b and c is smaller than d, the program print "a is larger than b and c is smaller than d".
- if number a is smaller than b and c is larger than d, the program print "a is smaller than b and c is larger than d".
- if none of the above is true, the program print "a is equal to b or c is equal to d".

When you are done, raise your hand!

[30]:
```
a = 4
b = 5
c = 6
d = 7

if a > b and c>d:
    print("a is larger than b and c is larger than d")
elif a <b and c<d:
    print("a is smaller than b and c is smaller than d")
```

```
elif a > b and c<d:
    print("a is larger than b and c is smaller than d")
elif a < b and c>d:
    print("a is smaller than b and c is larger than d")
else:
    print("a is equal to b or c is equal to d")
```

```
a is smaller than b and c is smaller than d
```

2.0.8 *Translate that!*

What does a `if-elif-else` statement do?

3 Standard Data Types in Python - strings

Category of Data type	Data type	Example
Numeric, scalar	Integer	1
	Floats	1.2
	Complex	1.5+0.5j
	Booleans	True
Container	strings	"Hello World"
	List	[1, "Hello World"]
	Tuple	(1, "Hello World")
	Set	{1, "Hello World"}
	Dictionary	{1: "Hello World", 2: 100}

3.1 What is a string in python?

- A sequence of characters
- Characters are ordered
- Immutable: characters can't be changed once created

3.2 Creating a String

- Assignment statement with `=`
- Function `str()`

[31]:
```
s = "A string of words"
s
```

[31]:
```
'A string of words'
```

[32]:
```
type(s)
```

[32]:
```
str
```

[33]:
```
x = 10**2
x
```

[33]: 100

[34]:
```
xs = str(x)
xs
```

[34]: '100'

[35]:
```
type(x)
```

[35]: int

[36]:
```
type(xs)
```

[36]: str

[37]:
```
int(xs)
```

[37]: 100

[38]:
```
"python"
```

[38]: 'python'

[39]:
```
'python'
```

[39]: 'python'

[40]:
```
"python'
```

```
  File "/var/folders/6m/8n2ktxl566j8yp0n_qx7x5bw0000gt/T/ipykernel_42182/
↪255721797.py", line 1
    "python'
           ^
SyntaxError: EOL while scanning string literal
```

[41]:
```
int(s)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/var/folders/6m/8n2ktxl566j8yp0n_qx7x5bw0000gt/T/ipykernel_42182/2179186071.py␣
↪in <module>
----> 1 int(s)
```

```
ValueError: invalid literal for int() with base 10: 'A string of words'
```

[42]:
```
a = "1000"
int(a)
```

[42]:
```
1000
```

[43]:
```
a = 1000
type(a)
```

[43]:
```
int
```

[44]:
```
a = 1000.0
type(a)
```

[44]:
```
float
```

[45]:
```
a = '1000'
type(a)
```

[45]:
```
str
```

[46]:
```
a = int("10100")
a
```

[46]:
```
10100
```

[47]:
```
a = int("010100")
a
```

[47]:
```
10100
```

[48]:
```
int("python")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/var/folders/6m/8n2ktxl566j8yp0n_qx7x5bw0000gt/T/ipykernel_42182/1217632113.py␣
 ↪in <module>
----> 1 int("python")

ValueError: invalid literal for int() with base 10: 'python'
```

3.3 String concatenation

"addition" of two strings (with +)

[49]:
```
1 +2
```

[49]: 3

[50]:
```
str_1 = 'hello'
str_2 = 'world'
```

[51]:
```
str_1 + str_2
```

[51]: 'helloworld'

[52]:
```
new_string = str_1 + str_2
new_string
```

[52]: 'helloworld'

Add a space (string ' ') in the middle of the two variables. A space is a character!

[53]:
```
a = str_1 + ' ' + str_2 + " "
a
```

[53]: 'hello world '

[54]:
```
2 * a
```

[54]: 'hello world hello world '

[55]:
```
str_1 * str_2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/var/folders/6m/8n2ktxl566j8yp0n_qx7x5bw0000gt/T/ipykernel_42182/1947353786.py␣
 ↪in <module>
----> 1 str_1 * str_2

TypeError: can't multiply sequence by non-int of type 'str'
```

[56]:
```
my_string = "hello world"
```

3.3.1 Group Exercise:

Create a new string variable `final_string` that adds three exclamation marks to the end of `my_string`.

```
my_string = "hello world"
```

When you are done, raise your hand!

[57]:
```
final_string = my_string + '!!!!'
final_string
```

[57]:
```
'hello world!!!!'
```

3.4 Indexing

To access each separate character in a string

Structure: `string[index]` * string variable name * square brackets * index: integer (starts from 0 in python)

[58]:
```
my_string = 'hello world'
```

[59]:
```
my_string[0]
```

[59]:
```
'h'
```

[60]:
```
my_string[2]
```

[60]:
```
'l'
```

3.4.1 strings are immutable

[61]:
```
s
```

[61]:
```
'A string of words'
```

[62]:
```
s[0]
```

[62]:
```
'A'
```

[63]:
```
s[0] = "B"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/var/folders/6m/8n2ktxl566j8yp0n_qx7x5bw0000gt/T/ipykernel_42182/1529794431.py␣
 ↪in <module>
----> 1 s[0] = "B"

TypeError: 'str' object does not support item assignment
```

Group Exploration Task: How do we get the last character in this string?

```
my_string = 'hello world'
```

When you are done, raise your hand

[64]:
```
len(my_string)
```

[64]:
```
11
```

[65]:
```
my_string[11]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
/var/folders/6m/8n2ktxl566j8yp0n_qx7x5bw0000gt/T/ipykernel_42182/4123106262.py␣
 ↪in <module>
----> 1 my_string[11]

IndexError: string index out of range
```

[66]:
```
my_string[11 -1 ]
```

[66]:
```
'd'
```

[67]:
```
my_string[len(my_string)-1]
```

[67]:
```
'd'
```

4 Assignment 2

- Where: You will use your UNT EUID and password to login the Jupyter Hub https://jupyterhub.cas.unt.edu/ to complete the assignment and submit it. You need to make sure to connect to UNT VPN beforehand.
- Programming exercises (5 points)
  - complete the solution in a function
  - the function is defined
  - Use the test case to evaluate whether your solution is correct
  - submit your completed work in the Jupyter Hub by 10 pm on 02/28/2023

5 Next Class

- String
- Iterations

Readings: Chapter 7

[ ]:
```

```