02.2_Program-Variables-Operators

January 25, 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: * The Way of Program * Operators * Variables

1.1 The Way of Program

Program for problem solving: * formulate problems * think creatively about solutions * express a solution clearly and accurately

1.2 What is a program? - Computation

A sequence of instructions that specifies how to perform a **computation**.

Different types of computation:

Type	Example (1)	Examples (2)
Mathematical	Solving a system of equations	Finding the roots of a polynomial
Symbolic	Searching and replacing text in a document	processing an image or playing a video

1.3 What is a program? - Instructions

Instruction	Function
input	Get data from the keyboard, a file, the network, or some other device.
output	Display data on the screen, save it in a file, send it over the network, etc.
math/operation	Perform basic mathematical operations like addition and multiplication.
conditional execution	Check for certain conditions and run the appropriate code.
repetition	Perform some action repeatedly, usually with some variation.

Divde and conquer!!!

```
[1]: input()
```

```
2
```

```
[1]: '2'
```

1.4 Problem 1 : Print "Hello world!"

Think about the five basic instructions: input, output, math, conditional execution, repetition.

For this problem: which instructions will we need?

input, output

```
[2]: ?print
```

```
[3]: print("Hello world!")
```

```
Hello world!
```

1.4.1 Analyze the program `print("Hello world!")`

- A statement
- `print()`: function
  - indicated by `()`
  - displays the value of the input (required `argument`) on the screen
  - one of the many Python *built-in* functions
- `"Hello world!"`:
  - `string` data type
  - input
  - required argument: the string of characters it should print out for you

```
[4]: print(1)
```

```
1
```

1.5 Problem 2 : Print "Hello world!" three times

Think about the five basic instructions: input, output, math, conditional execution, repetition.

For this problem: which instructions will we need?

input, output, repetition

[5]:
```python
for i in range(3):
    print("Hello world!")
```

```
Hello world!
Hello world!
Hello world!
```

1.6 Python as a calculator (Arithmetic operators)

The symbols are what you would expect, except for the "raise-to-the-power-of" operator, which you obtain with two asterisks: `**`. Try all of these:

`+   -   *   /   **   %   //`

The `%` symbol is the *modulo* operator (divide and return the remainder), and the double-slash is *floor division*.

Operators: special symbols that represent computations like addition and multiplication

[6]:
```python
2 + 2
```

[6]:
```
4
```

[7]:
```python
1.25 + 3.65
```

[7]:
```
4.9
```

[8]:
```python
5 - 3
```

[8]:
```
2
```

[9]:
```python
2 * 4
```

[9]:
```
8
```

[10]:
```python
7 / 2
```

[10]:
```
3.5
```

[11]:
```python
2**3
```

[11]:
```
8
```

```
[12]: 2 * 2 *2
```

```
[12]: 8
```

```
[13]: 5%2
```

```
[13]: 1
```

```
[14]: 5//2
```

```
[14]: 2
```

```
[15]: 5/2
```

```
[15]: 2.5
```

Let's see an interesting case:

```
[16]: 9**1/2
```

```
[16]: 4.5
```

1.6.1 Discuss with your neighbor:

What happened? Isn't $9^{1/2} = 3$? (Raising to the power 1/2 is the same as taking the square root.) Did Python get this wrong?

Compare with this:

```
[17]: 9 ** 0.5
```

```
[17]: 3.0
```

```
[18]: 9**(1/2)
```

```
[18]: 3.0
```

Yes! The order of operations matters!

Order	Operation
1	Parentheses means brackets()
2	Exponents (and Roots) means power
3	Multiplication & Division
4	Addition & Subtraction

1.6.2 Another example of order of Arithmetic operations

```
[19]: 3 + 3 / 2
```

```
[19]: 4.5
```

```
[20]: (3 + 3) / 2
```

```
[20]: 3.0
```

```
[21]: 3 + (3 / 2)
```

```
[21]: 4.5
```

1.6.3 Group Exercise:

Discuss and work with your neighbor:

Suppose the cover price of a book is \$24.95, but bookstores get a 40% discount. Shipping costs \$3 for the first copy and 75 cents for each additional copy. What is the total wholesale cost for 60 copies?

When you are done with your calculation, raise your hand.

```
[22]: (24.95 * (1-0.4) * 60) + (3 *1 + (60-1) * 0.75)
```

```
[22]: 945.4499999999999
```

1.7 Value and types

A value is one of the basic things a program works with, like a letter or a number

Each value has a type:

- 60 is an integer, 24.95 is a float, "Hello World" is a string
- we can use built-in function `type()` to check the type of a value

```
[23]: type(60)
```

```
[23]: int
```

```
[24]: type(24.95)
```

```
[24]: float
```

```
[25]: type("Hello World")
```

```
[25]: str
```

1.8 Variables

Two parts in a variable: **Name** and **Value**.

- Name: state
- value: "Texas", 48, "48"

1.8.1 Creating a variable: Assignment Statement

- Use the equal sign `name = value`
- The name of the variable goes on the left and the value on the right.
- Think of it as an arrow pointing from `name` to `value`.

We do not need to declare the type of a newly defined variable (makes the program more succinct than C/C++), python will infer the type based on the value.

[26]:
```
state = "Texas"
print(state)
```

```
Texas
```

[27]:
```
type(state)
```

[27]:
```
str
```

1.8.2 Updating a variable: Assignment Statement

- Use the equal sign `name = value`

[28]:
```
state = 48
print(state)
```

```
48
```

[29]:
```
type(state)
```

[29]:
```
int
```

1.8.3 Rules of Variable Names

- Leading character:
  - Must be a letter or the underscore character
  - Cannot be a number

[30]:
```
_state = 48
```

[31]:
```
$state = 48
```

```
  File "/var/folders/6m/8n2ktxl566j8yp0n_qx7x5bw0000gt/T/ipykernel_3771/
↪3927349684.py", line 1
    $state = 48
    ^
SyntaxError: invalid syntax
```

[32]:
```
4state = 48
```

```
  File "/var/folders/6m/8n2ktxl566j8yp0n_qx7x5bw0000gt/T/ipykernel_3771/
↪1752543828.py", line 1
    4state = 48
     ^
SyntaxError: invalid syntax
```

- can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)

[33]:
```
state_id = 48
```

[34]:
```
state&id = 48
```

```
  File "/var/folders/6m/8n2ktxl566j8yp0n_qx7x5bw0000gt/T/ipykernel_3771/
↪3220935798.py", line 1
    state&id = 48
    ^
SyntaxError: cannot assign to operator
```

- Case-sensitive

[35]:
```
state = 48
```

[36]:
```
STATE = 40
```

[37]:
```
state
```

[37]:
```
48
```

1.8.4 "Good" Variable Names

1. Be clear and concise.
2. Be written in English.
3. Not conflict with any Python keywords, such as `for`, `True`, `False`, `and`, `if`, or `else`. These are reserved for speical operations in Python and cannot be used as variable names.

[38]:
```
HSHDHAHFASFHAHF = 1
```

[39]:
```
for = 9
```

```
  File "/var/folders/6m/8n2ktxl566j8yp0n_qx7x5bw0000gt/T/ipykernel_3771/
↪338946984.py", line 1
    for = 9
        ^
SyntaxError: invalid syntax
```

[40]:
```
help("keywords")
```

```
Here is a list of the Python keywords.  Enter any keyword to get more help.

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

1.8.5 Good Variable Naming Format: pothole_case_naming

- lowercase words separated by underscores _.
- our suggested format as the underscores make it easy to read the variable, and don't add too much to the length of the variable name.
- As an example, consider the variable `temp_celsius`.

[41]:
```
fire_station_id = "101533"
```

[42]:
```
x = 1
```

[43]:
```
xx = 3
```

[44]:
```
xxxx
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/var/folders/6m/8n2ktxl566j8yp0n_qx7x5bw0000gt/T/ipykernel_3771/3574181483.py i
 ↪<module>
----> 1 xxxx
```

```
NameError: name 'xxxx' is not defined
```

[ ]:
```
university_name = "UNT"
```

1.8.6 Automation 1: arithmetic operations on variables

Example: Group Exercise Suppose the cover price of a book is \$24.95, but bookstores get a 40% discount. Shipping costs \$3 for the first copy and 75 cents for each additional copy. What is the total wholesale cost for 60 copies?

[45]:
```
24.95 * 0.6
```

[45]:
```
14.969999999999999
```

[46]:
```
price = 24.95
number = 0.6
price * number
```

[46]:
```
14.969999999999999
```

[47]:
```
book_price = price * number
print(book_price)
```

```
14.969999999999999
```

[48]:
```
price = 20
number = 1000

book_price = price * number
print(book_price)
```

```
20000
```

1.8.7 Automation 2: functions

[49]:
```
def calc_book_price(price, number):
    book_price = price * number
    return book_price
```

[50]:
```
calc_book_price(24.95, 0.6)
```

[50]:
```
14.969999999999999
```

[51]:
```
calc_book_price(20, 1000)
```

[51]:
```
20000
```

1.8.8 Group Exercise:

Discuss and work with your neighbor:

Suppose the cover price of a book is \$24.95, but bookstores get a 40% discount. Shipping costs \$3 for the first copy and 75 cents for each additional copy. What is the total wholesale cost for 60 copies?

> how to automate our solution in a function?

```
def calc_total_cost(copies):
    return
```

[52]:
```
def calc_total_cost(copies):
    return
```

2 Next Class

- Topic: Functions
- Readings: Chapters 3