Python Revisited

Cholwich Nattee and Nirattaya Khamsemanan May 20, 2023

1 Colab

Colab, short for Google Colaboratory, is an online platform provided by Google that allows users to run and collaborate on Python code using Jupyter notebooks. It offers a cloud-based development environment where you can write, execute, and share Python code in a browser. One of the main advantages of Colab is that it provides free access to computing resources, including CPU, GPU, and even TPU (Tensor Processing Units), which are useful for accelerating machine learning tasks. Colab notebooks also come pre-installed with popular libraries and frameworks commonly used in data science and machine learning, such as TensorFlow and PyTorch.
Colab is available at `https://colab.research.google.com`.

2 Python

Python is a widely used high-level programming language known for its simplicity and readability. It has gained significant popularity in the field of artificial intelligence (AI) due to its extensive libraries and frameworks specifically designed for AI development. Python provides a versatile platform for building and training AI models, making it a preferred choice for researchers and practitioners in the AI community.

Listing 1: Calculate sum of squares using Python

```python
numbers = [2, 3, 5, 7]
sum_squares = 0
for num in numbers:
  sum_squares += num**2
print("Sum of squares =", sum_squares)
```

Listing 2: Calculate sum of squares using C

```c
#include<stdio.h>

int main() {
  int numbers[] = {2, 3, 5, 7};
  int n = 4;
  int sum_squares = 0;
  for(int i=0; i<n; i++) {
    sum_squares += numbers[i]*numbers[i];
  }
  printf("Sum of squares = %d\n", sum_squares);
}
```

3 Variable

Variables are defined with the assignment operator `(=)`.

- variable names must **not** be reserved words.
- variable names must start with a letter (`A-Za-z`) or the underscore (`_`) character.
- variable names cannot start with a number.
- variable names can contain only alpha-numeric characters and underscores: `A-Z`, `a-z`, `0-9`, and `_`.
- variable names cannot contain space or special character (except underscores).
- variable names are case sensitive.

```python
pi = 3.14159
print(pi)
```

Table 1: List of Python Reserved Words

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

4 Data Types

Here are basic data types:

Type	**Example**	**Description**
`bool`	`True`	either `True=1` or `False=0`
`int`	`28`	an integer
`float`	`28.0`	a floating-point number
`str`	`"28"`	a sequence of characters

5 Operators and Expressions

Operators	**Description**	**Associativity**
`( )`	parentheses	
`**`	exponentiation	right-to-left
`+x, -x`	positive, negative	right-to-left
`*, /, //, %`	multiplication, division, floor division, modulus	
`+, -`	addition, subtraction	

Example 1. Write an expression for a formula $\frac{a}{bc}$

Example 2. Evaluate the following expresssions

1. `-2**4`

2. `15-5*2`

6 Formatting Outputs

Since Python 3.6, f-strings (formatted string literals) can be used to format output.

```
name = "Cholwich"
age = 20
print(f"Hello, {name}. You are {age} years old.")
```

```
pi = 3.14159
print(f"pi = {pi:.4f}")
```

For more details, see `https://peps.python.org/pep-0498/`

7 Libraries

The `import` statements are for importing packages.

Listing 3: Importing `math` 1

```
import math
print(math.cos(math.pi))
```

Listing 4: Importing `math` 2

```
import math as m
print(m.cos(m.pi))
```

Listing 5: Importing `math` 3

```
from math import *
print(cos(pi))
```

Example 3. Write a program to solve a quadratic equation $2x^2 + 7x + 5 = 0$ using the quadratic formula

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Note that you can use `math.sqrt` function to calculate the square root.
Then, revise your program to solve an equation $2x^2 + 4x + 5 = 0$.

8 Defining Functions

The keyword `def` introduces a function definition. Here are simple rules to define a function in Python.

- A function block begins with the keyword `def` followed by the function name, parentheses and a colon (:).
- Parameters can be placed within these parentheses.
- The function body is an indented code block.
- The statement `return` exits a function and optionally passes back a value to the caller. A return statement with no parameters is the same as return `None`.

Listing 6: Defining a quadratic function

```python
def f(x):
    return 2*x**2 + 5*x + 7

print(f"f(5) = {f(5):.2f}")
```

Note that all variables created inside a function are local variables. They can only be used inside the function.

9 Control Structures

9.1 `if` Statements

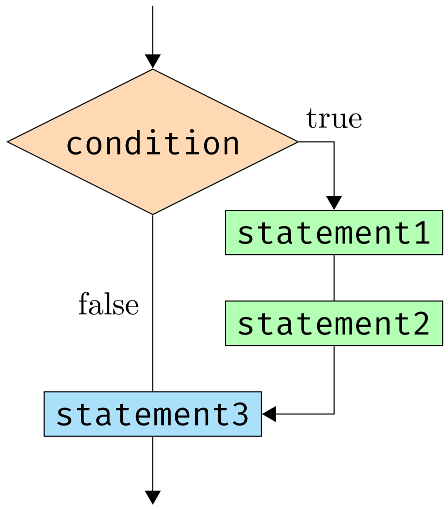
Here are a list of comparison and logical operators allowed in Python.

Operator	Description
`==`	True if the values of two operands are equal.
`!=`	True if values of two operands are not equal.
`>`	True if the left operand value is greater than the right operand value.
`<`	True if the left operand value is less than the right operand value.
`>=`	True if the left operand value is greater than or equal to the right operand value.
`<=`	True if the left operand value is less than or equal to the right operand value.
`and`	True if both the operands are true.
`or`	True if any of the two operands are true.
`not`	Reverse the logical state of its operand.

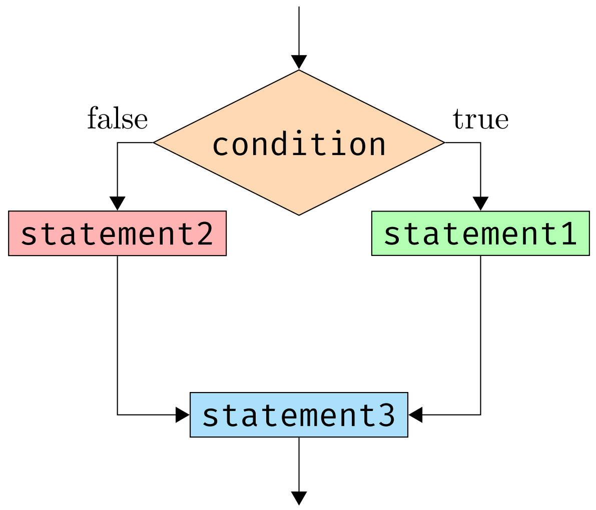
Listing 7: Comparison and Logical Operators

```
found = False
print(not found)
score = 50
print((score >= 0) and (score <= 100))
print(0 <= score <= 100)
print((score != 50) or (score != 40))
```

```python
if condition:
    statement1
    statement2
statement3
```




```python
if condition:
    statement1
else:
    statement2
statement3
```

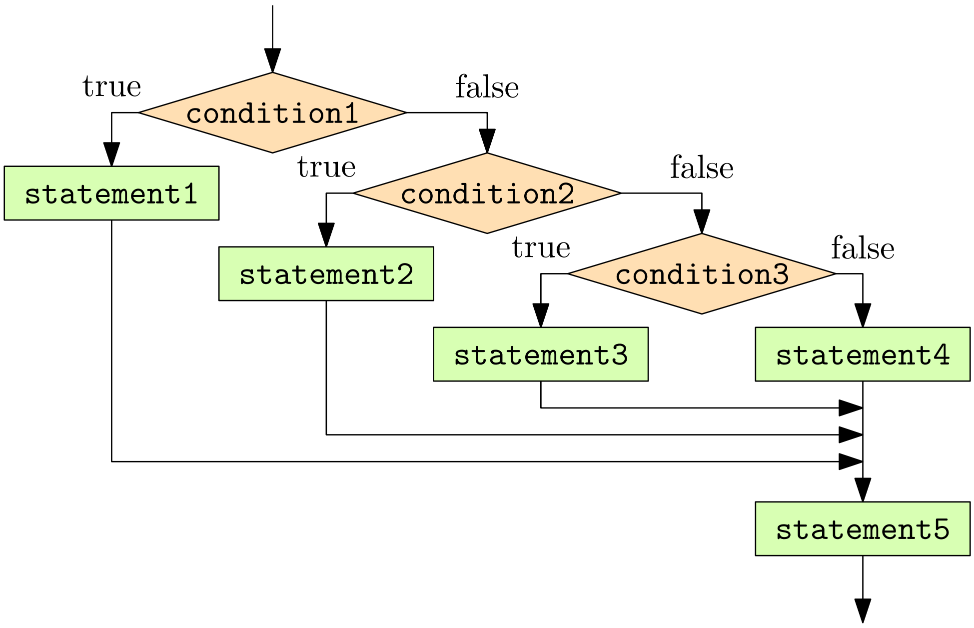


Example 4. Write a program to find real-number solutions of a quadratic equation $ax^2 + bx + c = 0$ using the quadratic formula

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

The program displays "No real-number solutions" when both solutions are complex numbers.

```
if condition1:
    statement1
elif condition2:
    statement2
elif condition3:
    statement3
else:
    statement4
statement5
```



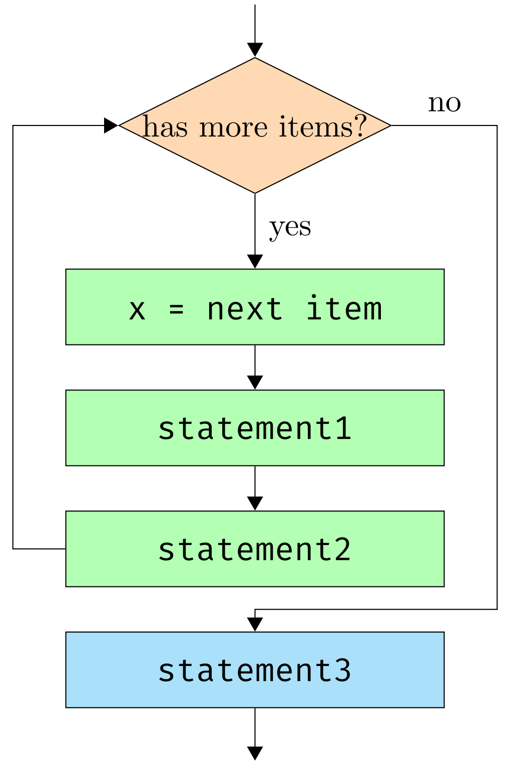
Listing 8: Solving a quadratic equation

```
import math as m

a, b, c = 2, 5, 1

if b**2-4*a*c < 0:
    print("There are no real-number solutions.")
elif a == 0:
    print("This is not a quadratic equation.")
else:
    s = m.sqrt(b**2 - 4*a*c)
    x1, x2 = (-b + s)/(2*a), (-b - s)/(2*a)
    print("Solution = %.2f %.2f" % (x1, x2))
```

9.2 `for` Statements

The `for` statement in Python repeats for each member in a provided list. For loops are generally used when a block of code is executed repeatedly a fixed number of times.

```
for x in list1:
    statement1
    statement2
statement3
```



The `range` function is a way to create a list of integers:

$$\texttt{range(start, stop, step)}$$

- The first parameter, `start`, is a starting number of the list.
- The second parameter, `stop`, an *open* end of the list. That means numbers in the list are always least than but never equal to the number `stop`.
- The third parameter, `step`, represents a size of increment step.
- All three parameters must be integers.

The `break` statement immediately stops the innermost loop.
The `continue` statement skips the rest of the current iteration and continue to the next round.

Example 5. Write a code that computes the value of $\sum_{n=1}^{N} n^i$

Example 6. Newton's method

Let $f(x)$ be a differentiable function. Given x_0, a solution of $f(x) = 0$ can be approximated by the tangent line at x_0:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Given f and df, write a code using a for loop to find a solution of $f(x) = 0$ using the Newton's method. The loop ends when

1. $|f(x_n)| < \varepsilon$ and x_n is a solution,

   (Note: use `abs` function to compute the absolute value)

2. $f'(x_n) = 0$ and there is no solution,

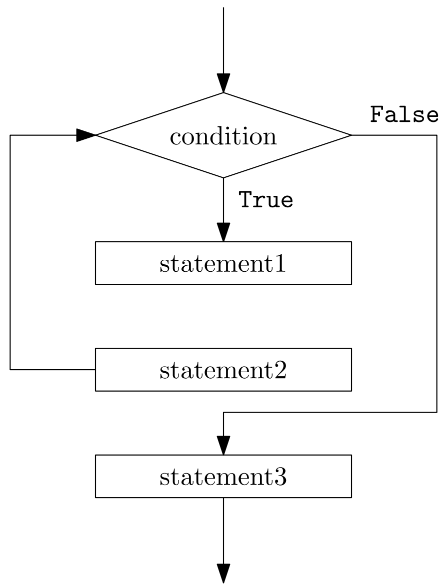
3. $n > N$ where N is the maximum number of iterations and there is no solution.

```
def f(x):
  return x**3 - x**2 - 1

def df(x):
  return 3*x**2 - 2*x

epsion = 1e-10          # 1e-10 == 1*(10**(-10))
N = 20
```

9.3 `while` Statements

The `while` statement repeats while a condition is evaluated `True`.

```
while condition:
    statement1
    statement2
statement3
```




```
sum = 0
i = 0
while i <= 100:
    sum += i
    i += 1
print(f"Sum = {sum}")
```

Example 7. Complete the following function using a while loop to check if `p` is a prime number.

```python
def isprime(p):
    ....
```

10 Data Structures

Here are basic data structures in Python:

Type	Example	Description
`list`	`[1, 1, 2, 3, 5, 8]`	a mutable list of values
`tuple`	`(1, 1, 2, 3, 5, 8)`	an immutable list of values
`set`	`{2, 3, 5, 7}`	a set of values
`dict`	`{"a": 2, "b": 3, "c": 5}`	a collection of key-value pairs

Each element in a data structure can be accessed using a subscript.

```python
l = [2, 3, 5, 7, 11, 13]

print(l[0], l[3], l[5])
print(l[-1], l[-2], l[-5])

d = {"a": 2, "b": 3, "c": 5}
print(d["a"], d["b"])
```

List Comprehension

```
l = [2, 3, 5, 7, 11, 13]
m = [x+3 for x in l]
```

```
m = []
for x in l:
    m.append(x+3)
```

```
m = [5, 6, 8, 10, 14, 16]
```

```
l = [2, 3, 5, 7, 11, 13]
m = [x+3 for x in l if x>3]
```

```
m = []
for x in l:
    if x>3:
        m.append(x+3)
```

```
m = [8, 10, 14, 16]
```

Example 8. The Sieve of Eratosthenes finds all the prime numbers that is less than or equal to a given integer n by

1. Create a list of n integers where all elements are set to `True`.
2. Start from $p = 2$.
3. Set element at $2p, 3p, 4p, \ldots, n$ to `False`.
4. Set p to the next prime number and repeat step 3.
5. Stop when $p \geq \sqrt{n}$