Introduction to Programming Using Python

Fadoua Ghourabi
Ochanomizu University
ghourabi.fadoua@ocha.cc.ac.jp

April 16, 2019

Basic of programming

- A program is a sequence of instructions/commands/statements to compute a mathematical formula or to perform an action
- A program is written in a programming language understood by the computer
- A compiler checks that a program has no errors:
  - the instructions of a program are valid sentences (syntax)
    "3.2 -5/6" is a valid sentence, but "3.2 - print("Hello python")" is not
  - the instructions of the program have meanings (semantic)
    "a - 5/6" has no meaning if "a" is not defined
- When error, the compiler displays error messages, often helpful for us to correct the program
- An interpreter transforms a program into a script, called machine code, that can be executed by the computer

Basic of programming

Elements of any programming language

1. Objects, types and basic operations
2. Variables and assignment
3. Conditionals
4. Loops
5. Functions
6. Data structures

1) Objects, types and basic operations

Objects

- Scalars and non-scalar

  "3.2" is a scalar object but "Hello python" is not
- An object must have a type
  - to define the operations that can be performed on the object

    operation "3 - 4" is possible on objects of type integer
  - to tell the interpreter how to encode the object

1.1) Basic types for scalar objects

- type `int` represents integers, e.g 3, -5
- type `float` represents real numbers, e.g. 3.0, -1.4235, 1.6E3
- type `bool` represents the Boolean values `True` and `False`

1) Objects, types and basic operations

Operations on types `int` and `float`

- the results of `i + j`, `i - j` and `i * j` are of types
  - `int`, if i and j are of type `int`
  - `float`, if either of i and j is of type `float`
- `i // j` is integer division, the result is of type `int`
- `i / j` performs float division, the result is of type `float`
- `i % j` computes the remainder, the result is
  - `int`, if i and j are of type `int` (i modulo j)
  - `float`, if either of i and j is of type `float`
- `i ** j` computes the power, the result is
  - `int`, if i and j are of type `int`
  - `float`, if either of i and j is of type `float`
- Comparison is given by $==$ (equal), $!=$ (not equal), $>$, $<$, $>=$ and $<=$, the result is of type `bool`

1) Objects, types and basic operations

Operations on types `bool` are: `and`, `or`, `not`

		`and`	`or`
True	True	True	True
True	False	False	True
False	False	False	False

	not
True	False
False	True

1) Objects, types and basic operations

1.2) Type `str` for string object (non-scalar objects)

- Objects of type `str` are inclosed in ”.” or '.'
- Arithmetic operators can be applied to objects of type `str`. Which of the following operations are accepted?

```
3 * 'apple '
3 + 'apples '
'3 ' + 'apples '
```

- An object of type `str` is a sequence of indexed characters
  - `'I love apples.'[0]` displays the first character 'I'
  - `'I love apples.'[13]` displays the last character '.'
  - `'I love apples.'[20]` is an error because 20 exceeds the length
  - Negative numbers are used to index from the end:
    `'I love apples.'[-1]` displays '.'
- Slicing means extracting a substring from a string

  `string[start:end]`

  `'I love apples.'[8:12]` displays the substring 'apples'

  `'I love apples.'[:5]` starts from 0 and displays the substring 'I love'

  `'I love apples.'[8:]` displays the substring 'apples'

1) Objects, types and basic operations

- Character encoding is the internal representation of the characters
- For instance, the Unicode standard supports 120,000 characters, including characters from the Japanese language
- Most of the webpages use the Unicode standard
- Unicode standard is the default in python, otherwise include the following comment at the begining of your program:

```
# -*- coding:  utf-8 -*-
```

2) Variables and assignment

- A variable is a label of an object,
  recall variable x in mathematics
- An assignment gives a value to a variable

```
h = 4
w = 5.4
rectangle_perimeter = (h + w) * 2
rectangle_area = h * w
h2 = 6
rectangle_area = h2 * w
```

- Multiple assignment is allowed in python

```
h, w = 3, 5/6
```

- In python, the type of a variable is deduced from the assignment. Thus, we do not need to declare variables and their types (but mandatory in C or Java). Python is a dynamically typed programming language. Is it a good or a bad thing?

3) Conditionals, 4) Loops

An example to start:

```
a,b=525,50
if(a<b):
    a,b,gcd = b,a,a
else:
    gcd = b

while(b!=0):
    gcd = b
    a,b = b, a%b
print(gcd)
```

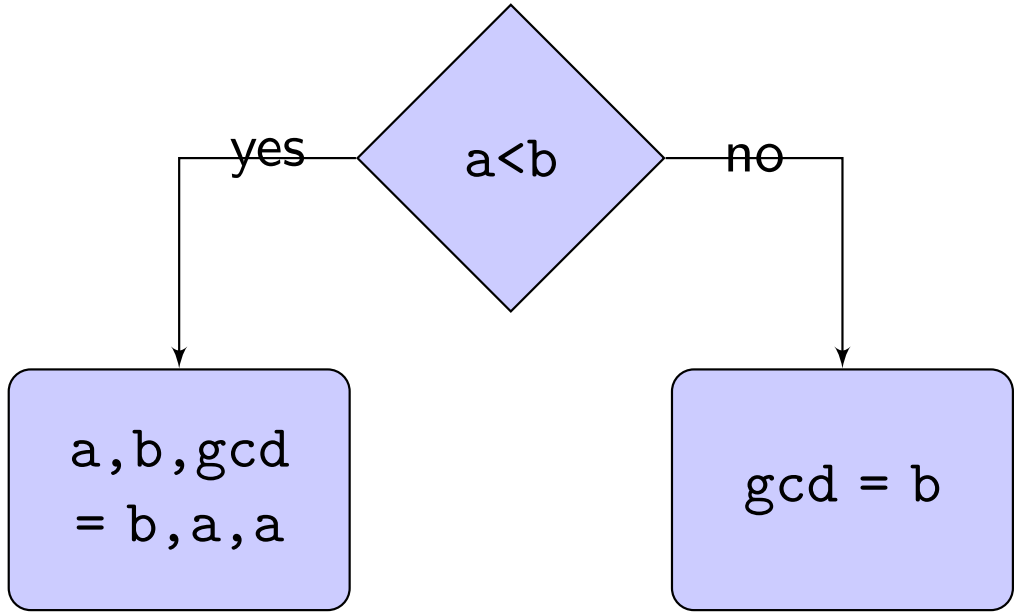
Euclid GCD algorithm:

$$a = q_0 b + r_0$$
$$b = q_1 r_0 + r_1$$
$$r_0 = q_2 r_1 + r_2$$
$$\dots$$
$$r_n = q_{n+2} r_{n+1} + 0$$

where $a, b \in \mathbb{Z}$

3) Conditionals

```
a,b=525,50
if(a<b):
    a,b,gcd = b,a,a
else:
    gcd = b
```

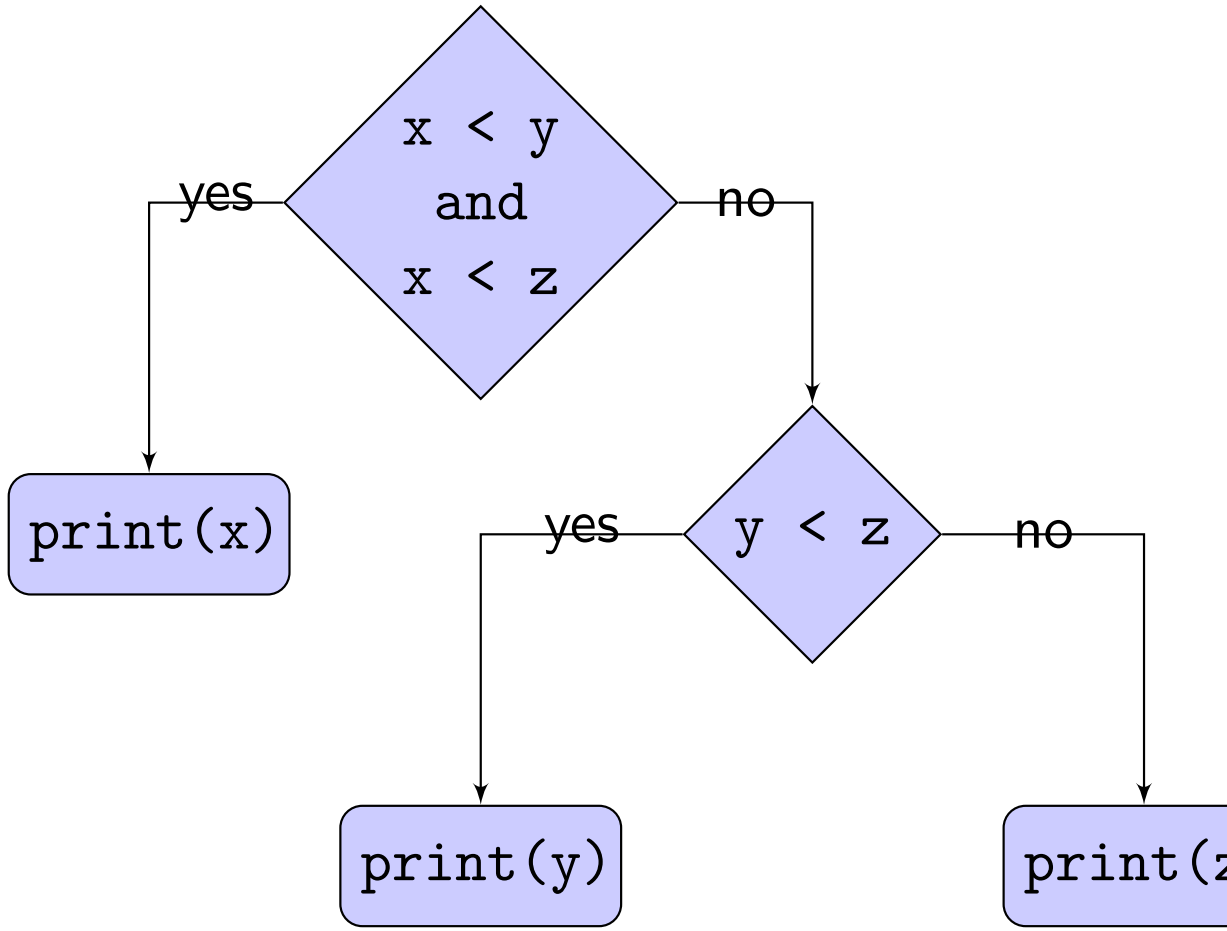


A conditional statement is of the following form

```
if <boolean expression>:
    <code>
else:
    <code>
```

3) Conditionals

```
x,y,z=2,-5,3
if (x < y) and (x < z):
    print(x)
elif (y < z):
    print(y)
else:
    print(z)
```



3) Conditionals

- Nested conditional statement: when the true block or the false block contains another conditional
- `elif` stands for `else if`

```
if x% 2 == 0:
    if x%3 == 0:
        print('Divisible by 2 and 3')
    else:
        print('Divisible by 2 and not 3')
elif x%3 == 0:
    print('Divisible by 3 and not 2')
```

Practice: Compare the pQE score of the GAP team with the average pQE scores of PBTS 1 and 2

3) Loops (Iteration)

- When we want the program to do the same thing many times, we use iteration
- Two kinds of iteration: while-loop and for-loop

```
while <boolean expression>:
    <code>
```

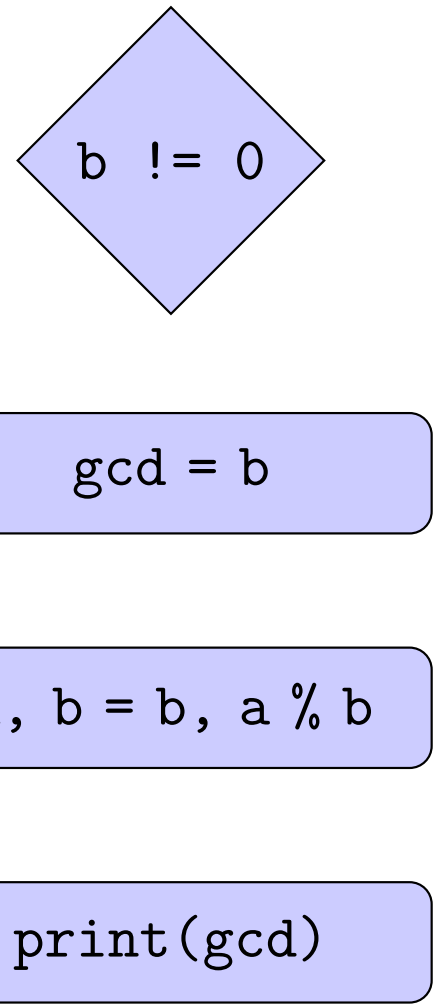
```
for <variable> in <sequence>:
    <code>
```

3) Loops (Iteration)

```
a,b=525,50
if(a<b):
    a,b,gcd = b,a,a
else:
    gcd = b

while(b!=0):
    gcd = b
    a,b = b, a%b
print(gcd)
```

Complete the flowchart of the while-loop:

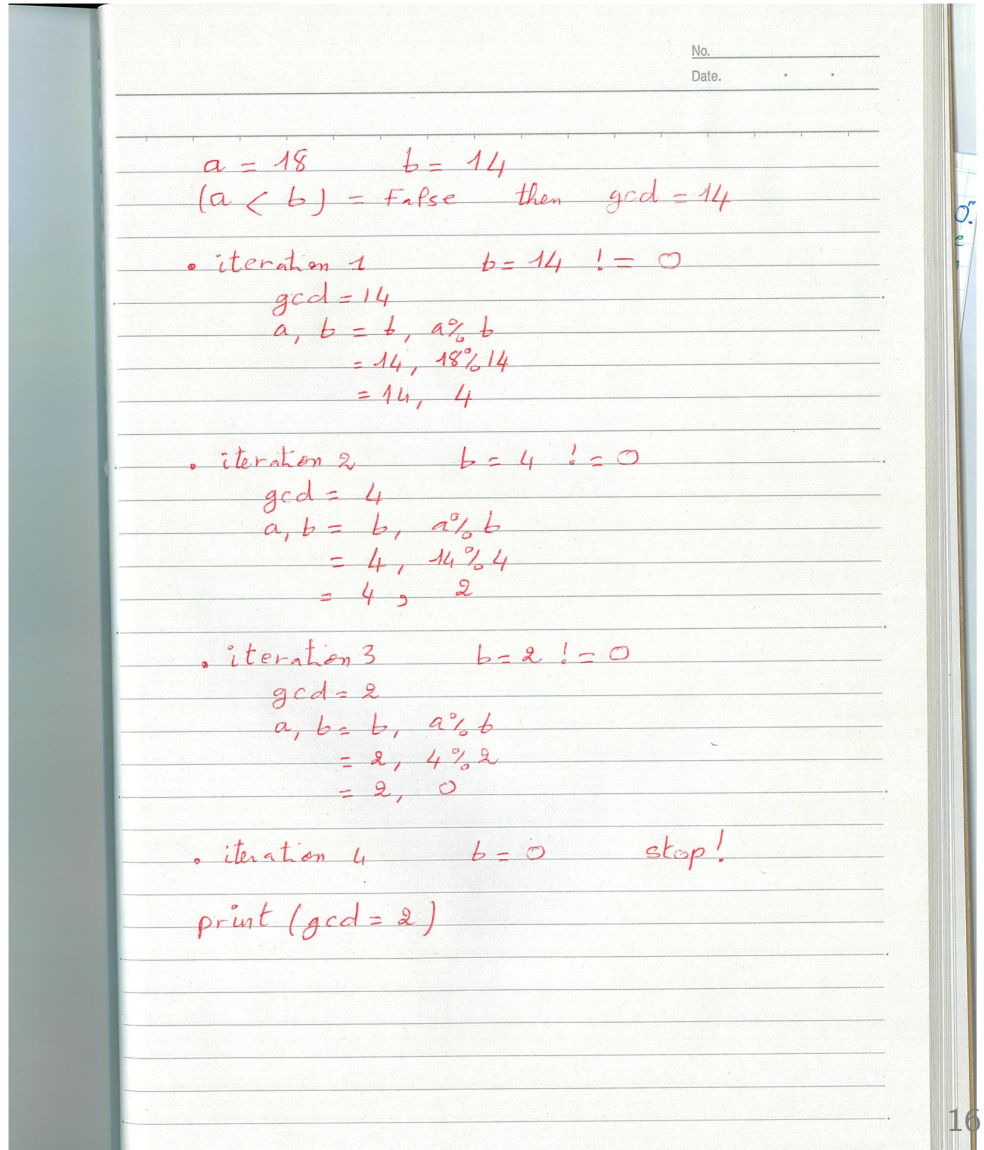


3) Loops (Iteration)

To understand how a loop behaves, we pretend to be a computer and perform a pen-and-paper simulation:

```
a,b=18,14
if(a<b):
    a,b,gcd = b,a,a
else:
    gcd = b

while(b!=0):
    gcd = b
    a,b = b, a%b
print(gcd)
```



3) Loops (Iteration)

Practice:

1. Write a code (using while-loop) to compute x^a, for any x and a of type `int`
2. What is infinite looping?

3) Loops (Iteration)

Try the following for-loops and deduce how `range` works:

```
for i in range(3):
    print(i)
```

```
for i in range(4, 12, 5):
    print(i)
```

```
for i in range(24, 3, -5):
    print(i)
```

3) Loops (Iteration)

We can use strings for a sequence. What does the following code do?

```
count = 0
for c in "I love apples":
    if (c == 'e'):
        count = count + 1
print(count)
```

5) Functions

- In mathematics, a functions takes a sequence of variables and computes a result,
  e.g. $f(x, y) = x^3 - 2xy^2 + y - 10$, where $x, y \in \mathbb{Z}$
- A mathematical function is an abstraction of computation.
  We can apply $f(x, y)$ to any values for x and y in $\mathbb{Z}$
- Similarly, a program function is an abstraction of computation that have the following properties:
  - Generality: we want to apply the gcd code to any values of a and b (not only 525 and 50)
  - Modularity: we want to call the interface of a function (not the code lines)

```
a,b=525,50
if(a<b):
    a,b,gcd = b,a,a
else:
    gcd = b

while(b!=0):
    gcd = b
    a,b = b, a%b
print(gcd)
```

5) Functions

```
def gcd(a,b):
    if(a<b):
        a,b,gcd = b,a,a
    else:
        gcd = b

    while(b!=0):
        gcd = b
        a,b = b, a%b
    return gcd
```

- `gcd(18,4)` is a function call of function `gcd`
- `a` and `b` are arguments/parameters of function `gcd`
- `gcd(18, 4)` returns 2

5) Functions

```
def gcd(a,b=4):
    if(a<b):
        a,b,gcd = b,a,a
    else:
        gcd = b

    while(b!=0):
        gcd = b
        a,b = b, a%b
    return gcd
```

- Parameter `b` has a default value, in this case 4
- `gcd(18)` does not specify the value of `b` meaning that the function computes with the default value of `b` which is 4
- `gcd(400, 16)` returns 4