Rotman

PROGRAMMING STRUCTURES

February 8, 2022 Prepared by Niti
TDMDAL & FinHUB

Rotman School of Management
UNIVERSITY OF TORONTO

Programming Structures

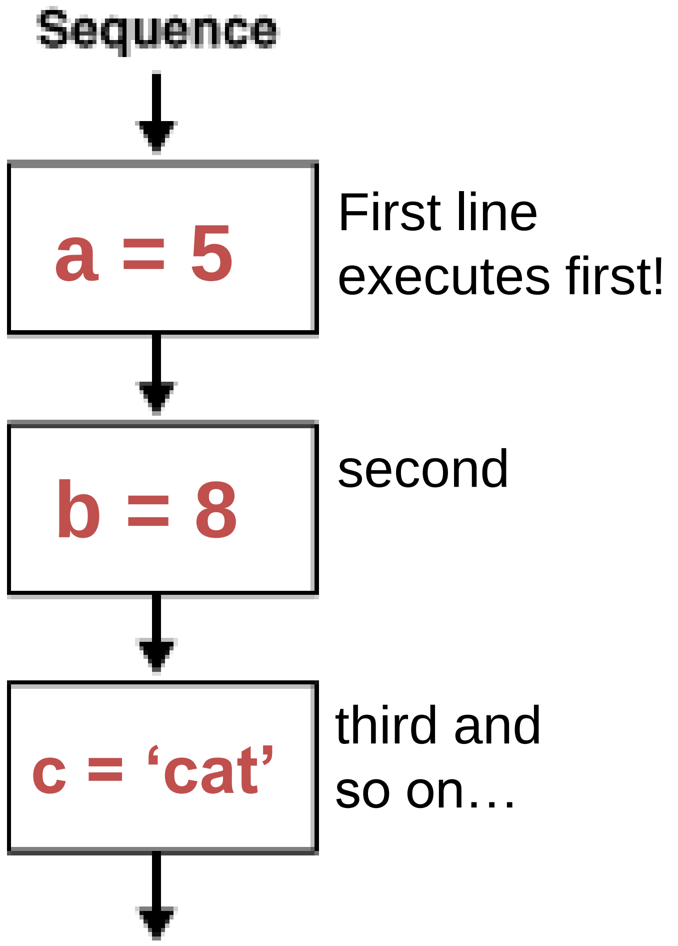
1. Sequential
2. Conditional
3. Iteration

Sequential

Programming Structures

SEQUENTIAL

Programs are mostly written sequentially, meaning the first line of program runs first followed by the program in the second line, then the third line and so on.



Rotman

Programming Structures: Sequential

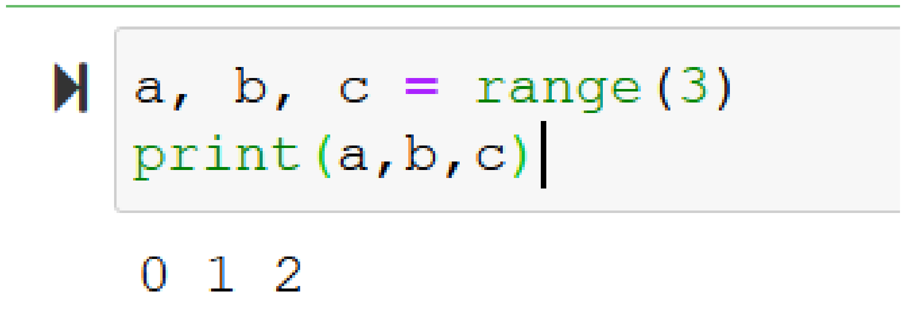
- **Statements**
  - Statements are basic units of instruction that Python interpreter parses and processes.
  - In general, Python executes statements sequentially.
  - It is possible to alter this sequential execution behavior by writing conditional or iteration statements.

Programming Structures: Sequential

- **Line Continuation**
  - Typically, one statements are written per line.
  - Long line of statements are generally considered poor practice. They should be split up across several lines.
  - The Style Guide for Python Code also known as PEP8 states that the maximum line length should be 79 characters in Python code.
  - Statements must be split such that it make syntactic sense. Otherwise, since the interpreter assumes a newline character terminates a statement, an error will be raised.

Programming Structures: Sequential

- **Multiple Statements per Line**
  - Multiple statements per line are allowed if they are separated by semicolon.
  - However, PEP8 generally discourages it.
  - Instead use a pythonic way to combine multiple statements in one line.



```
a, b, c = range(3)
print(a,b,c)
```

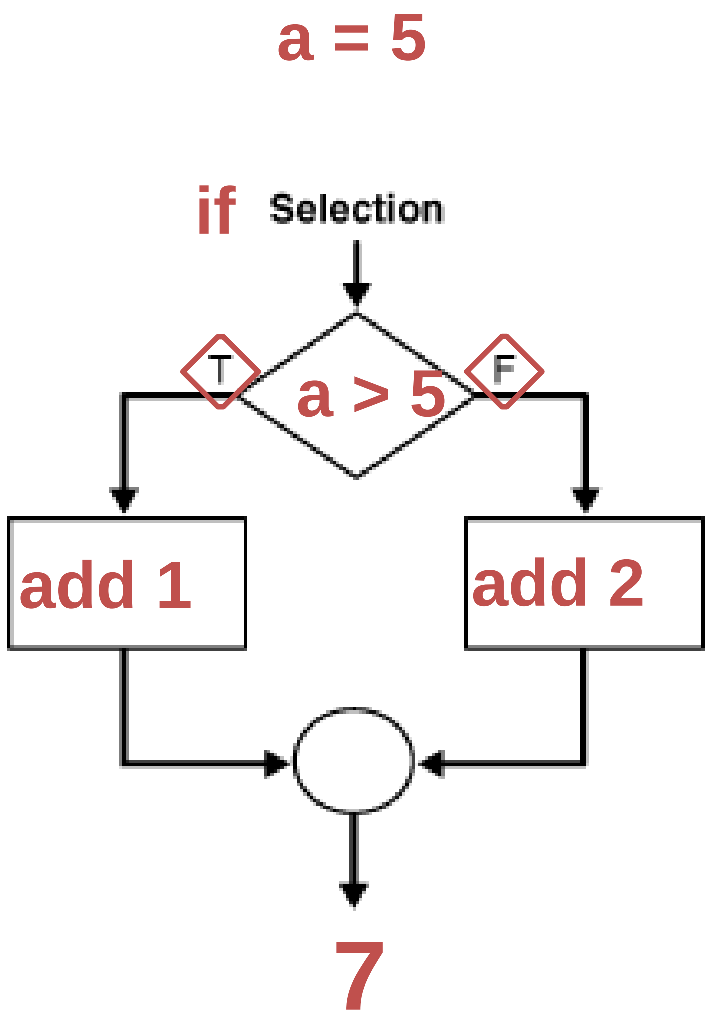
```
0 1 2
```

Conditional

Programming Structures

CONDITIONAL

Programs become more useful when we can change its behavior given a condition is satisfied.

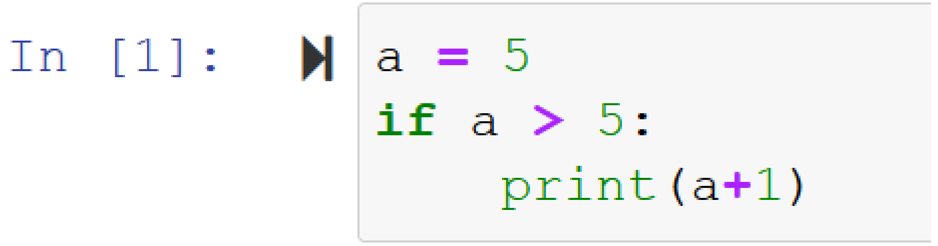


Programming Structures: Conditional

- **Logical Expressions**
  - Also known as Boolean logic, logical expression are expressions which return True or False upon evaluation.
  - Any combination of operators can be used to create logical expression as long as they are syntactically valid.
  - More than one logical expression can be constructed to determine whether:
    - all criteria have met, or
    - at least one of the criteria has met.
  - Logical operators are used to combine multiple conditions.

Programming Structures: Conditional

- **if statement**
  - constructed using the ***if*** keyword followed by logical expression(s)
  - executes a block of code only if certain conditions are met
  - once the condition is met, the rest of the code will not be executed by the interpreter

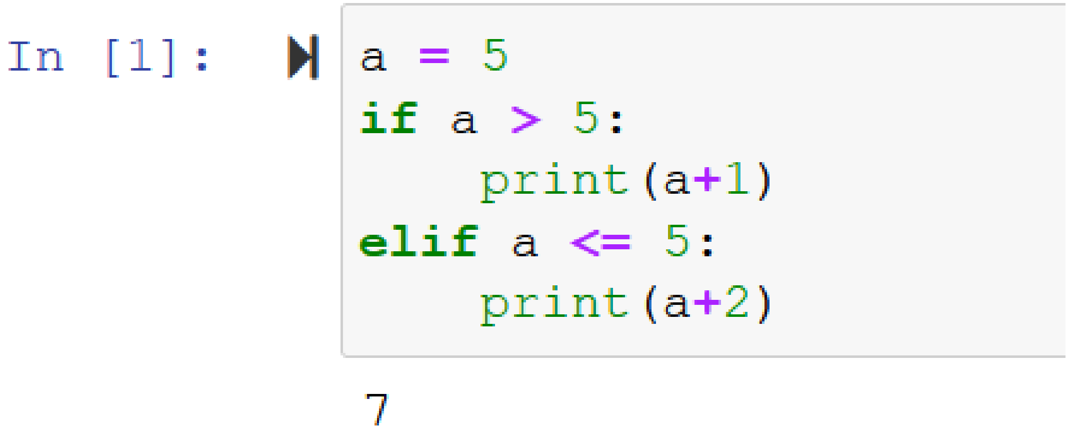


```
In [1]:     a = 5
            if a > 5:
                print(a+1)
```

Rotman

Programming Structures: Conditional

- **elif statement**
  - ***elif*** can be used to define second condition onwards
  - Any number of *elif* statement is allowed as long as there is at least one *if* statement

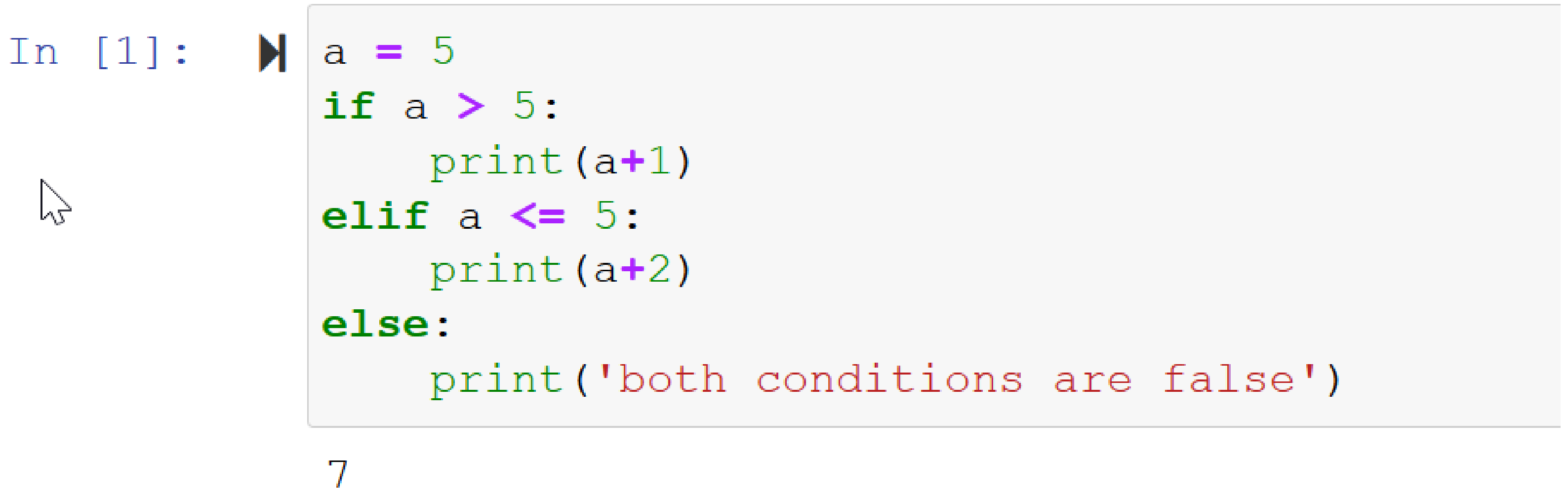


```
In [1]:  a = 5
         if a > 5:
             print(a+1)
         elif a <= 5:
             print(a+2)

         7
```

Programming Structures: Conditional

- **else statement**
  - ***else*** defines the last condition, which is not required but can be added
  - only one *else* statement allowed



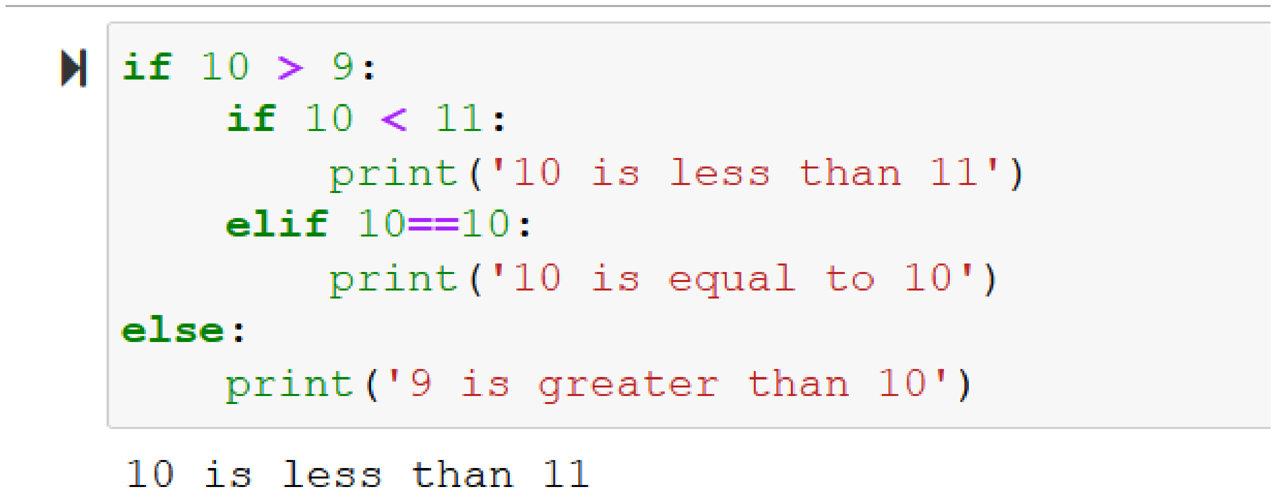
```
In [1]:   a = 5
          if a > 5:
              print(a+1)
          elif a <= 5:
              print(a+2)
          else:
              print('both conditions are false')

          7
```

Rotman

Programming Structures: Conditional

- **Nested if statement**
  - A statement is nested if a statement contains another statement of the same kind.
  - Nested if-statements contains another if-statements inside of it.
  - Indentations must be followed inside the nested if-statements as well.



```python
if 10 > 9:
    if 10 < 11:
        print('10 is less than 11')
    elif 10==10:
        print('10 is equal to 10')
else:
    print('9 is greater than 10')
```

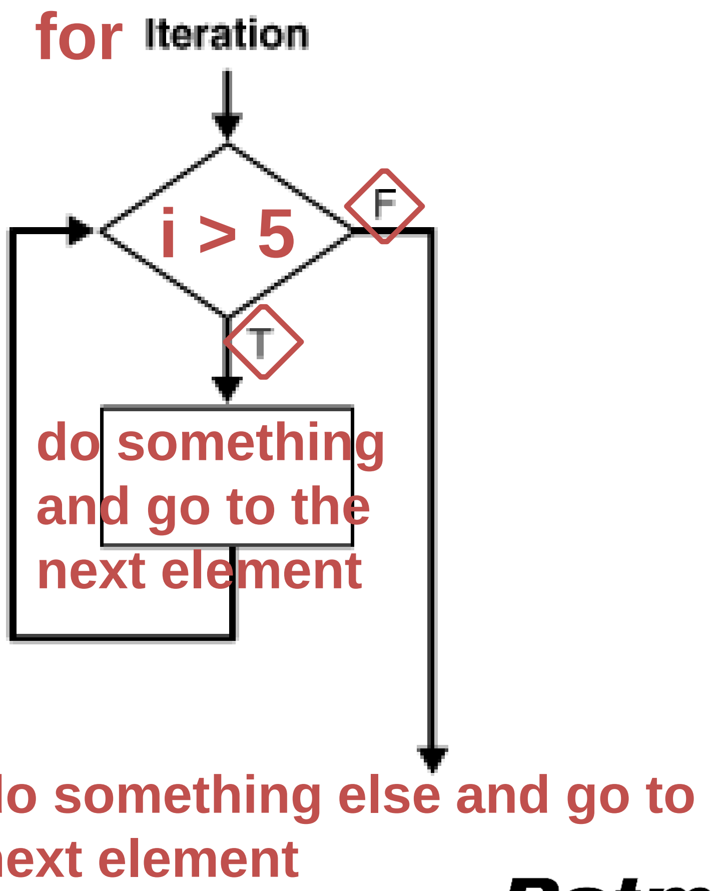
```
10 is less than 11
```

Iteration

ITERATION

Programs become powerful when the same block of code can be repeatedly executed on either identical tasks or similar tasks.

numlist = [4,8,10,15]

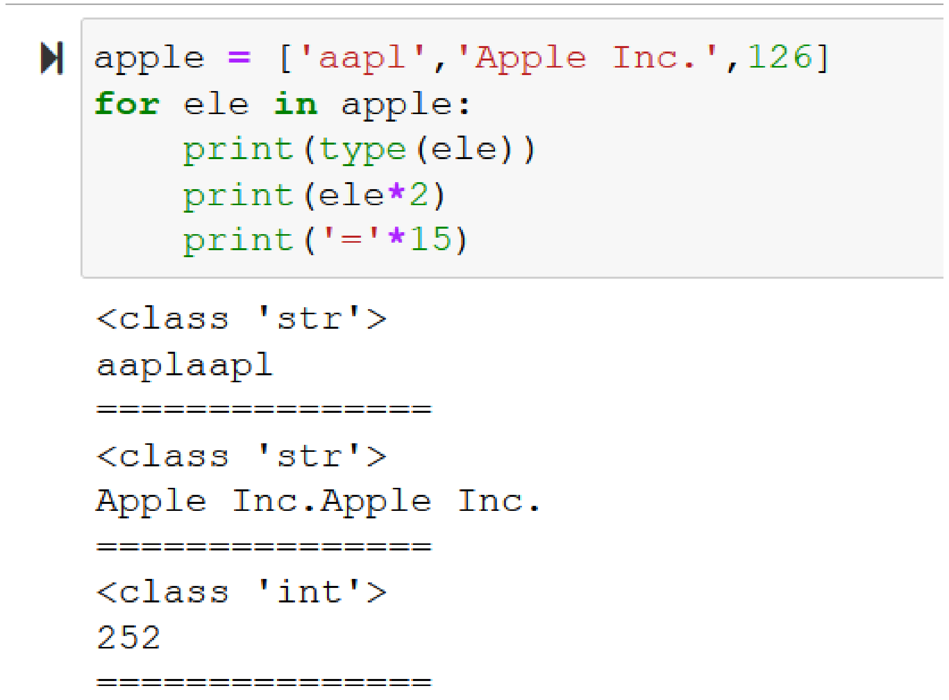

for
Iteration
i > 5
F
T
do something and go to the next element
do something else and go to next element

[8,10,15]

Rotman

Programming Structures: Iteration

- **for loop**
  - Used to perform a given operation on every element of a sequence.
  - Also referred to as definite iteration because the number of repetition is defined in advance.
  - The program terminates when there are no more elements in the sequence to operate on.
  - Nested for-loops are also possible.



```python
apple = ['aapl','Apple Inc.',126]
for ele in apple:
    print(type(ele))
    print(ele*2)
    print('='*15)
```

```
<class 'str'>
aaplaapl
===============
<class 'str'>
Apple Inc.Apple Inc.
===============
<class 'int'>
252
===============
```

Rotman

Programming Structures : Iteration

```
numlist = [4,8,10,15]
```

1. What if we want to add 1 to each item of this list if greater than 5?

→ Use conditional and iteration together

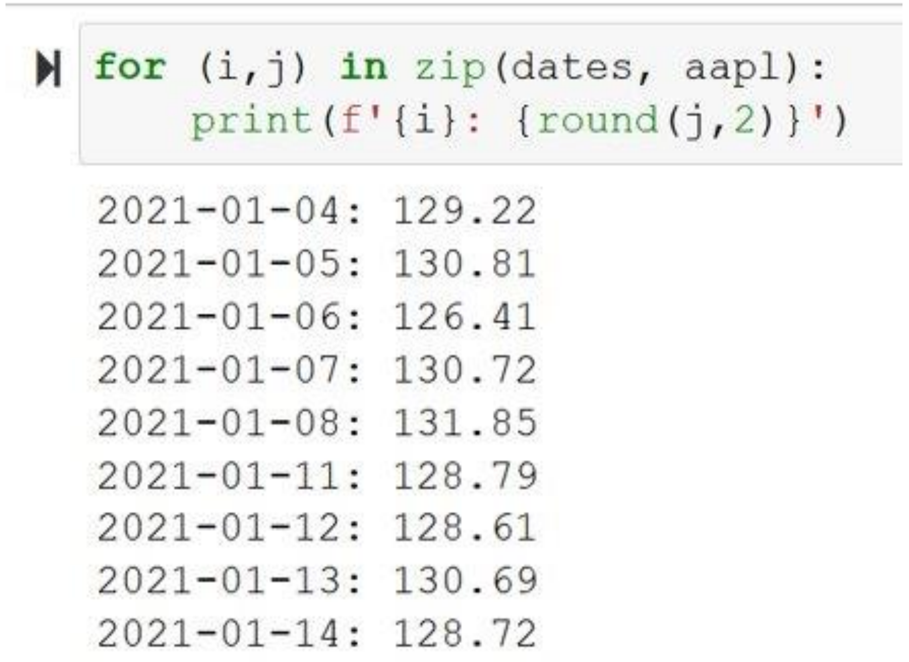
```
for i in numlist:
    if i > 5:
        print(i+1)
    else:
        print(i)
```

```
→ 4
  9
  11
  16
```

Rotman

Programming Structures: Iteration

- **for loop on multiple sequences**
  - It is possible to loop over each item of multiple lists simultaneously given they all are of the same length.
  - To do so, the lists must be zipped together using the “zip” function.

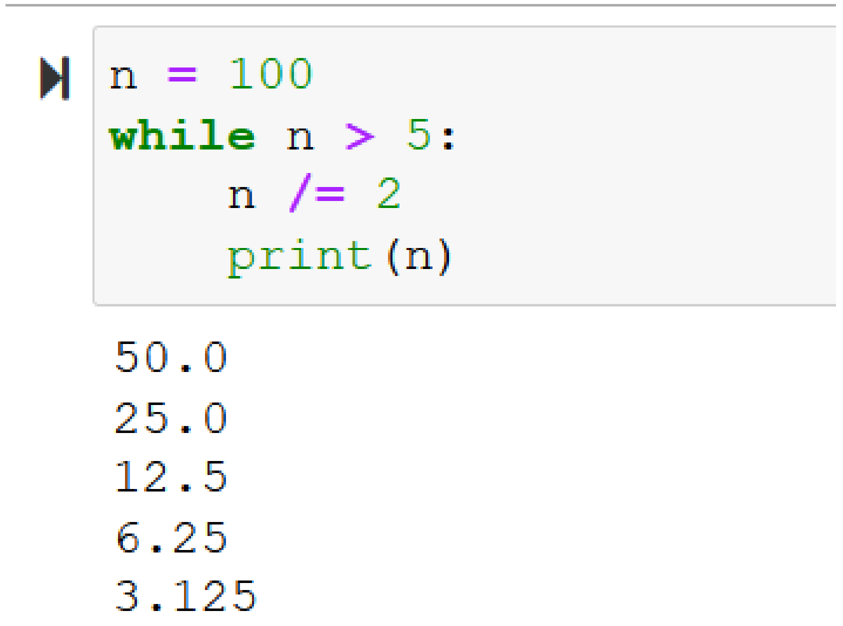


```python
for (i,j) in zip(dates, aapl):
    print(f'{i}: {round(j,2)}')
```

```
2021-01-04: 129.22
2021-01-05: 130.81
2021-01-06: 126.41
2021-01-07: 130.72
2021-01-08: 131.85
2021-01-11: 128.79
2021-01-12: 128.61
2021-01-13: 130.69
2021-01-14: 128.72
```

Programming Structures: Iteration

- **while loop**
  - Used to repeat a given operation until some condition is met.
  - Also referred to as definite iteration because the number of repetition is defined in advance.
  - The program terminates when there are no more elements in the sequence to operate on.



```
n = 100
while n > 5:
    n /= 2
    print(n)
```

```
50.0
25.0
12.5
6.25
3.125
```

Rotman

List Comprehension

LIST COMPREHENSION

- Creates sequences from other sequences using a very compact syntax

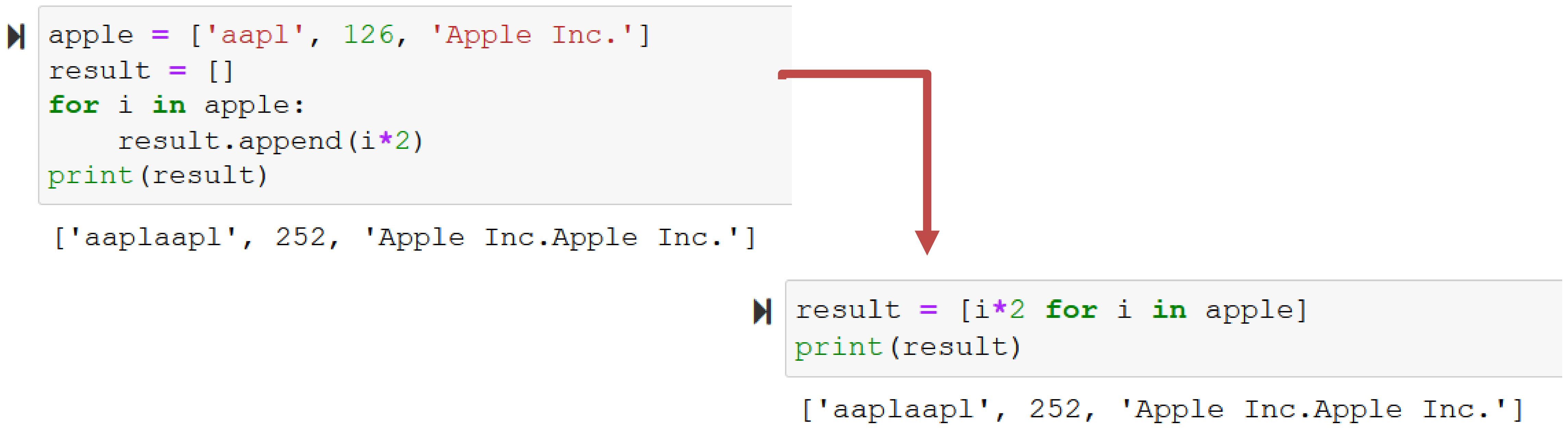
operation	iteration	sequence
[print(i)	for i in	somelist]

```
[ print(i)   for i in   somelist ]
```

Rotman

Programming Structures: List Comprehension

- It does not improve performance but it reduces the lines of code



```
apple = ['aapl', 126, 'Apple Inc.']
result = []
for i in apple:
    result.append(i*2)
print(result)
```

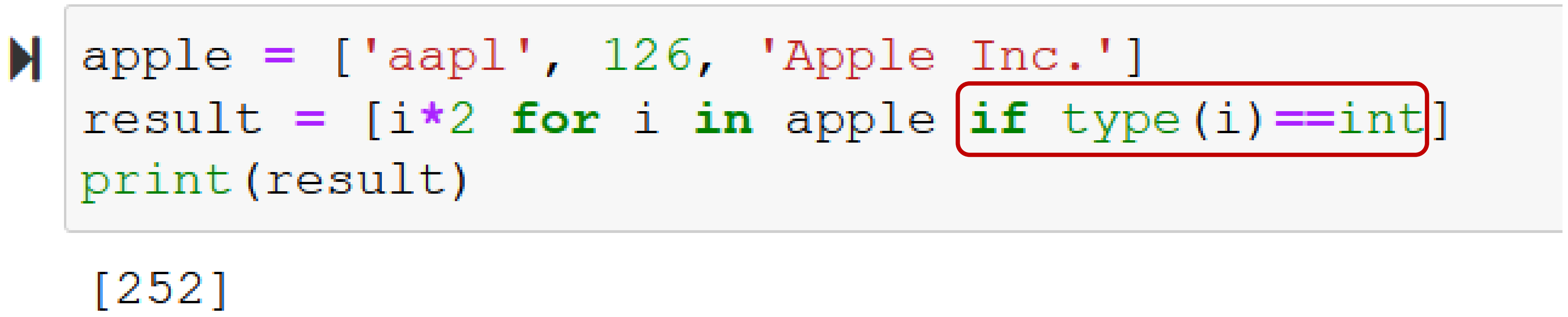
```
['aaplaapl', 252, 'Apple Inc.Apple Inc.']
```

```
result = [i*2 for i in apple]
print(result)
```

```
['aaplaapl', 252, 'Apple Inc.Apple Inc.']
```

Programming Structures: List Comprehension

1. What if we want to we want to conditionally operate on elements?



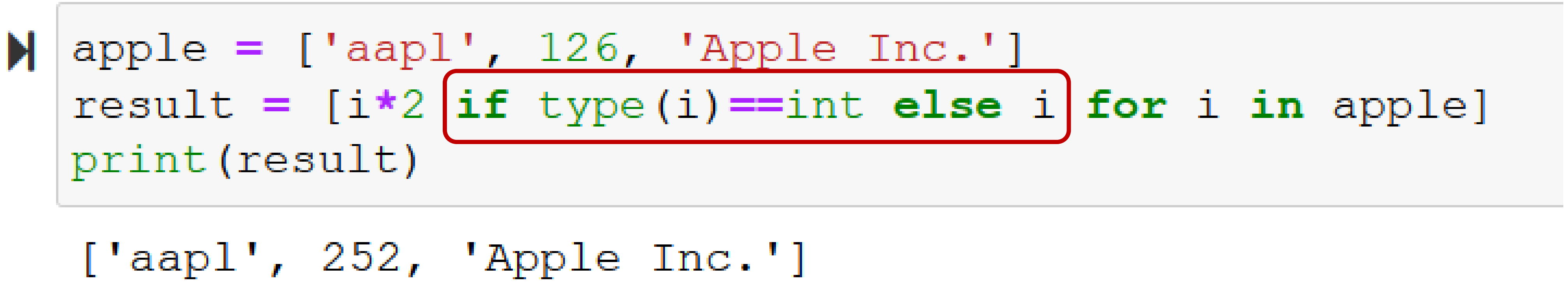
```
apple = ['aapl', 126, 'Apple Inc.']
result = [i*2 for i in apple if type(i)==int]
print(result)
```

```
[252]
```

Rotman

Programming Structures: List Comprehension

2. What if we want to we add more conditions?

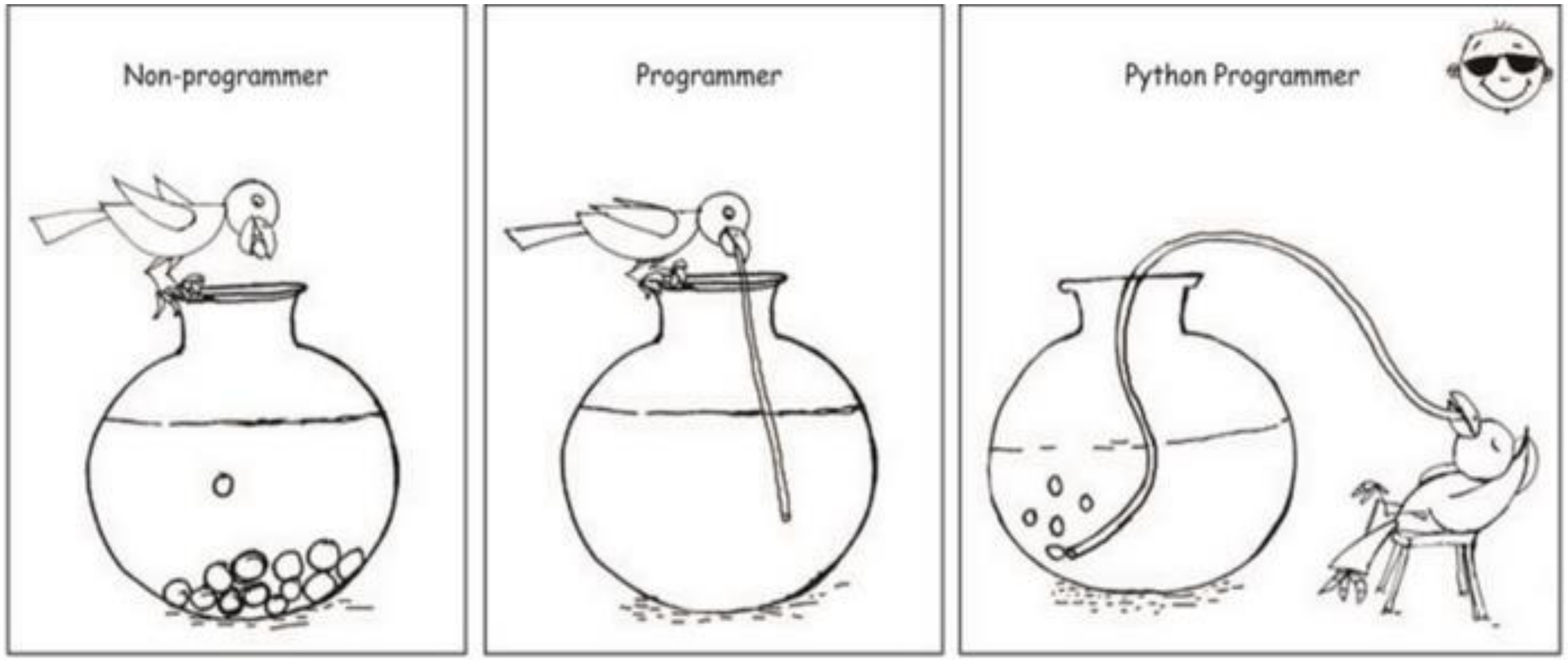


```
apple = ['aapl', 126, 'Apple Inc.']
result = [i*2 if type(i)==int else i for i in apple]
print(result)
```

```
['aapl', 252, 'Apple Inc.']
```

Rotman

Questions?



Who wants to become a Python Programmer?

Thank you