墨学教育
—MELBSTUDY—

FIT9136 Week 1

Basic Elements

授课老师：Joe

FIT9136平时班 – Week 1

Atomic – one single data value

e.g.	Integer:	3
	float:	3.0
	Boolean	True/False

Collective – one or more data values

e.g.	string:	‘abcd’
	tuple:	(‘a’,3,5,2.0, True)
	list:	[‘a’,3,5,2.0, True]
	dictionary:	{‘name’ : ‘John’, ‘age’: 25}
	set:	{1,2.0, ‘John’} $\leftarrow$ No duplicates

Tuples and strings are immutable!

FIT9136平时班 – Week 1

Numbers supports arithmetic operations such as, +, -, *, /

Boolean are results of comparison e.g.

- Item in list
- a > b

String

- Common functions:
    - String.upper()
    - String.lower()
    - String.count()
    - String.isalpha()
    - String.isnumeric()
    - String.isupper()
    - String.islower()
    - String.split()
    - String.strip()

FIT9136平时班 – Week 1

- For collective data types, the first index is 0 not 1

- Slicing

    [start_index, end_index, step_size] ← Does not include the end index

For string comparison, python compares value base on lexicographical order

FIT9136平时班 – Week 1

Order of operators

Arithmatic (+,-,*,/, etc) →

Relational (>,<, ==, etc) →

Logical (and, or)

FIT9136平时班 – Week 1

Input

Input('prompt message')

- All input are string, if you want to do arithmetic operations, need to change type

Int(Input('prompt message'))

Output

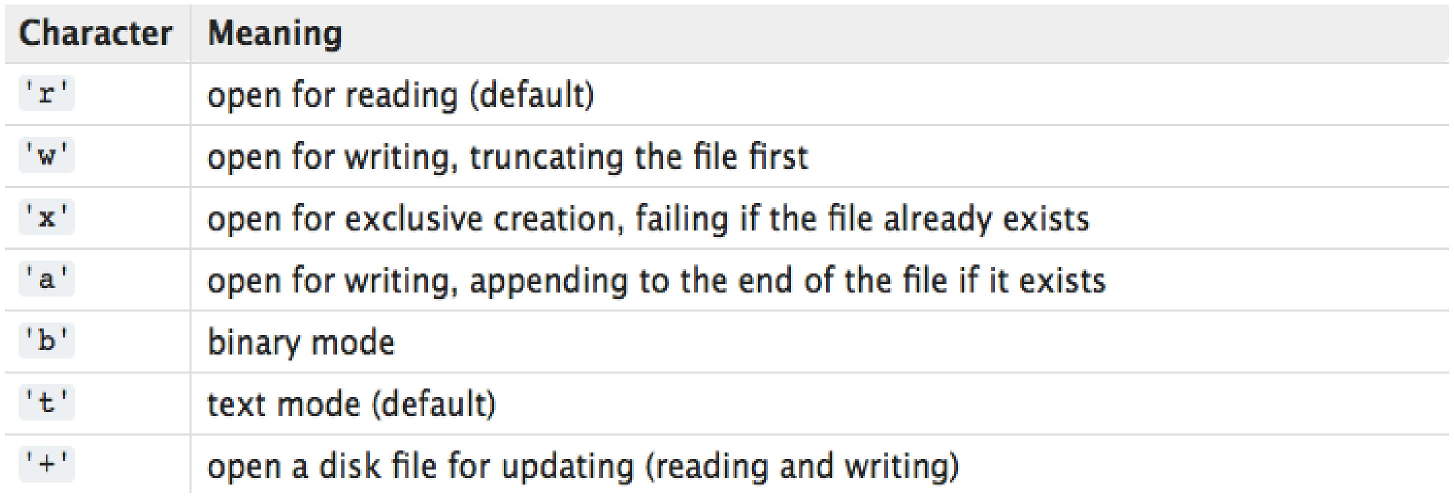
Print()

FIT9136平时班 – Week 1

File input

Open(file, mode)

You will use ‘r’ and ‘w’ most of the time



Character	Meaning
`'r'`	open for reading (default)
`'w'`	open for writing, truncating the file first
`'x'`	open for exclusive creation, failing if the file already exists
`'a'`	open for writing, appending to the end of the file if it exists
`'b'`	binary mode
`'t'`	text mode (default)
`'+'`	open a disk file for updating (reading and writing)

FIT9136平时班 – Week 1

File input

Readline() – read one line at a time (until \n is reached)
Readlines() – read all line and store each line in a list
Read() – read everything and return a single string

File.close() – close the file after using it, good practice but not really necessary

FIT9136平时班 – Week 1

File output

Open(file, ‘w’) – overwrite existing content
Open(file, ‘a’) – append at the end

File_handle.write(content)
Or
Print(content, file = file_handle)

Control Structure

FIT9136平时班 – Week 1

```
flag = bool(input("I love programming. True/False?"))

if flag == True:
    print("YES")
    print("It is true!")

else:
    print("NO")
    print("It is false!")
```

FIT9136平时班 – Week 1

```
message = "Welcome to FIT9136"
letter = 'o'
count = message.count(letter)
if count < 1:
    print(letter + " doesn't exist in " + message)
else:
    print(letter + " exists in " + message)
    print(letter + " occurs " + str(count) + " times")
```

FIT9136平时班 – Week 1

```
message = "Welcome to FIT9133"
letter = 'o'
count = message.count(letter)
if count < 1:
    print(letter + " doesn't exist in " + message)
else:
    print(letter + " exists in " + message)
    if count >= 5:
        print(letter + " occurs 5 times or more")
    else:
        print(letter + " occurs less than 5 times")
```

```
message = "Welcome to FIT9133"
letter = 'o'
count = message.count(letter)
if count < 1:
    print(letter + " doesn't exist in " + message)
elif count >= 5:
    print(letter + " exists in " + message)
    print(letter + " occurs 5 times or more")
else:
    print(letter + " exists in " + message)
    print(letter + " occurs less than 5 times")
```

FIT9136平时班 – Week 1

While loop
- Continue to execute as long as the condition is True

For loop
- No condition needs to be defined
- For i in range(len(list)):
- For i in list:

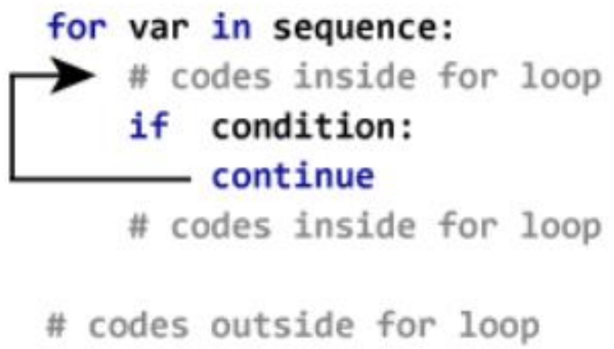
A while loop can always substitute a for loop but a for loop cannot substitute a while loop

FIT9136平时班 – Week 1

Use a for loop
- When u know how many items are in the collection
- When u want to traverse the collection in a regular manner.

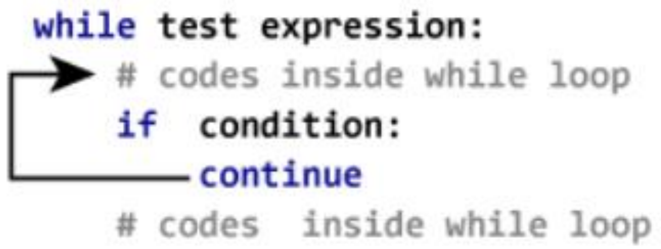
Use a while loop
- When u don’t know how many items are in the collection
- When you want to traverse the collection in an irregular manner.

FIT9136平时班 – Week 1

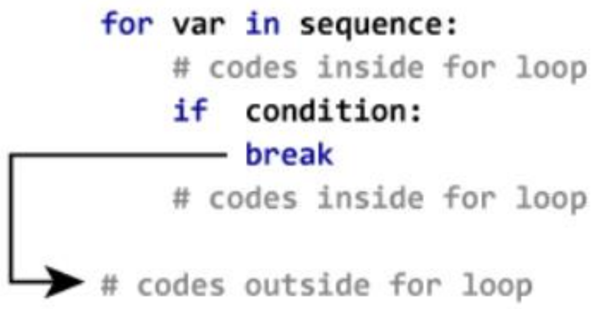


```
for var in sequence:
    # codes inside for loop
    if  condition:
        continue
    # codes inside for loop

# codes outside for loop
```

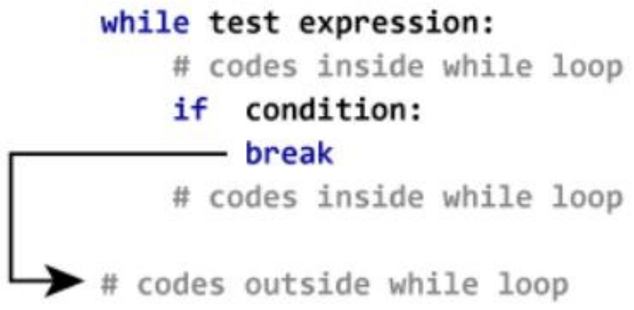


```
while test expression:
    # codes inside while loop
    if  condition:
        continue
    # codes  inside while loop
```



```
for var in sequence:
    # codes inside for loop
    if  condition:
        break
    # codes inside for loop

# codes outside for loop
```



```
while test expression:
    # codes inside while loop
    if  condition:
        break
    # codes inside while loop

# codes outside while loop
```

FIT9136平时班 – Week 1

```
number_list = [3, 11, 9, 7, 6, 5, 100, 20, 9, 6, 3, 1, 0]
target = 9
for number in number_list:
    if number == target:
        print("The target number is in the list")
        break
```

```
a_list = [1, 2, 3]
b_list = [2, 5, 6]
for itemA in a_list :
    for itemB in b_list:
        if itemA == itemB:
            break
        print(itemA, itemB)
```

Functions

FIT9136平时班 – Week 1

```
x = 5
y = 10

print(x,"+",y,"=",x+y)
print(x,"-",y,"=",x-y)
print(x,"/",y,"=",x/y)
print(x,"*",y,"=",x*y)
```

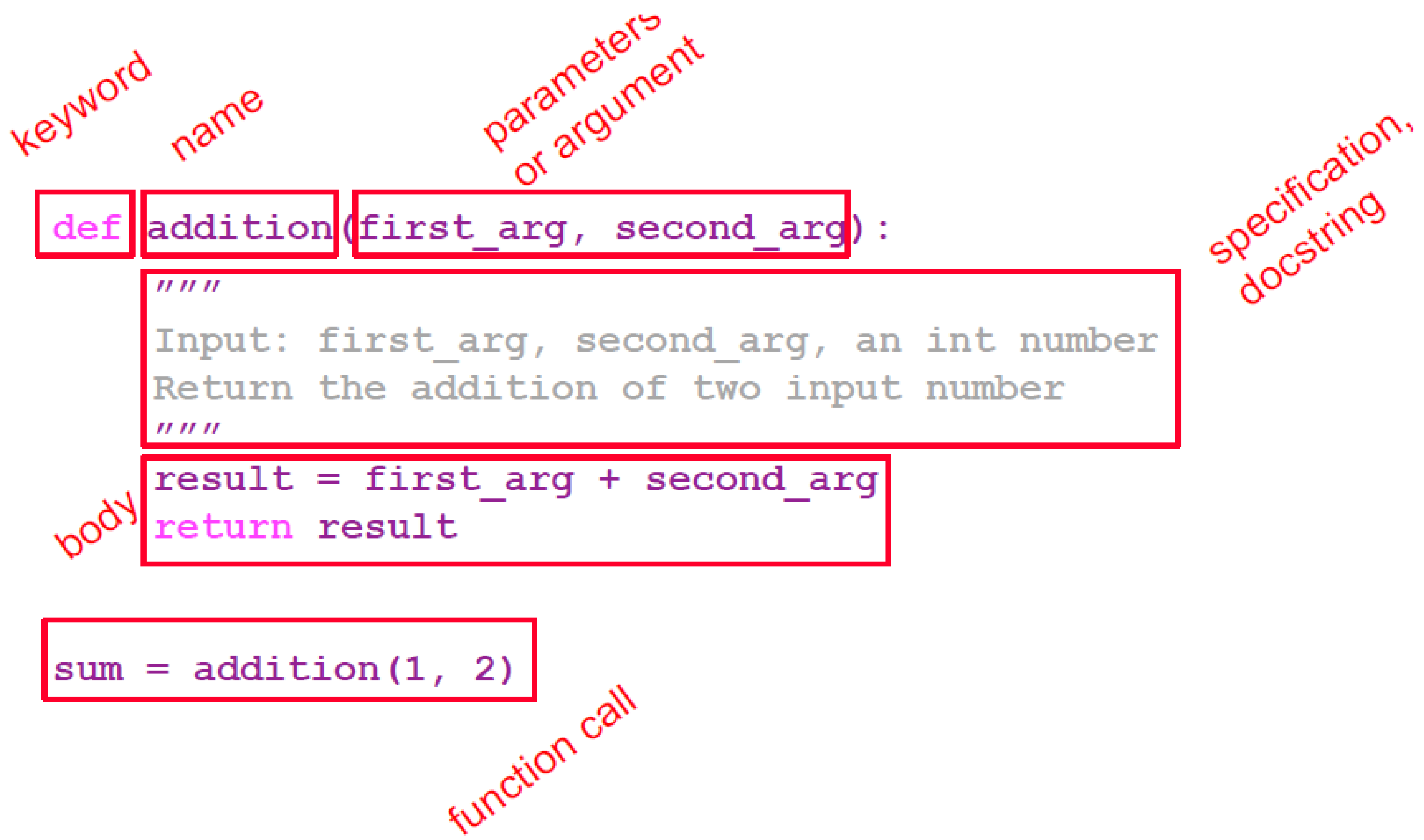
```
def mathinfo():
    x = 5
    y = 10

    print(x,"+",y,"=",x+y)
    print(x,"-",y,"=",x-y)
    print(x,"/",y,"=",x/y)
    print(x,"*",y,"=",x*y)
```

```
def mathinfo(x, y):

    print(x,"+",y,"=",x+y)
    print(x,"-",y,"=",x-y)
    print(x,"/",y,"=",x/y)
    print(x,"*",y,"=",x*y)
```

FIT9136平时班 – Week 1



```
def addition(first_arg, second_arg):
    """
    Input: first_arg, second_arg, an int number
    Return the addition of two input number
    """
    result = first_arg + second_arg
    return result

sum = addition(1, 2)
```

FIT9136平时班 – Week 1

return vs. print

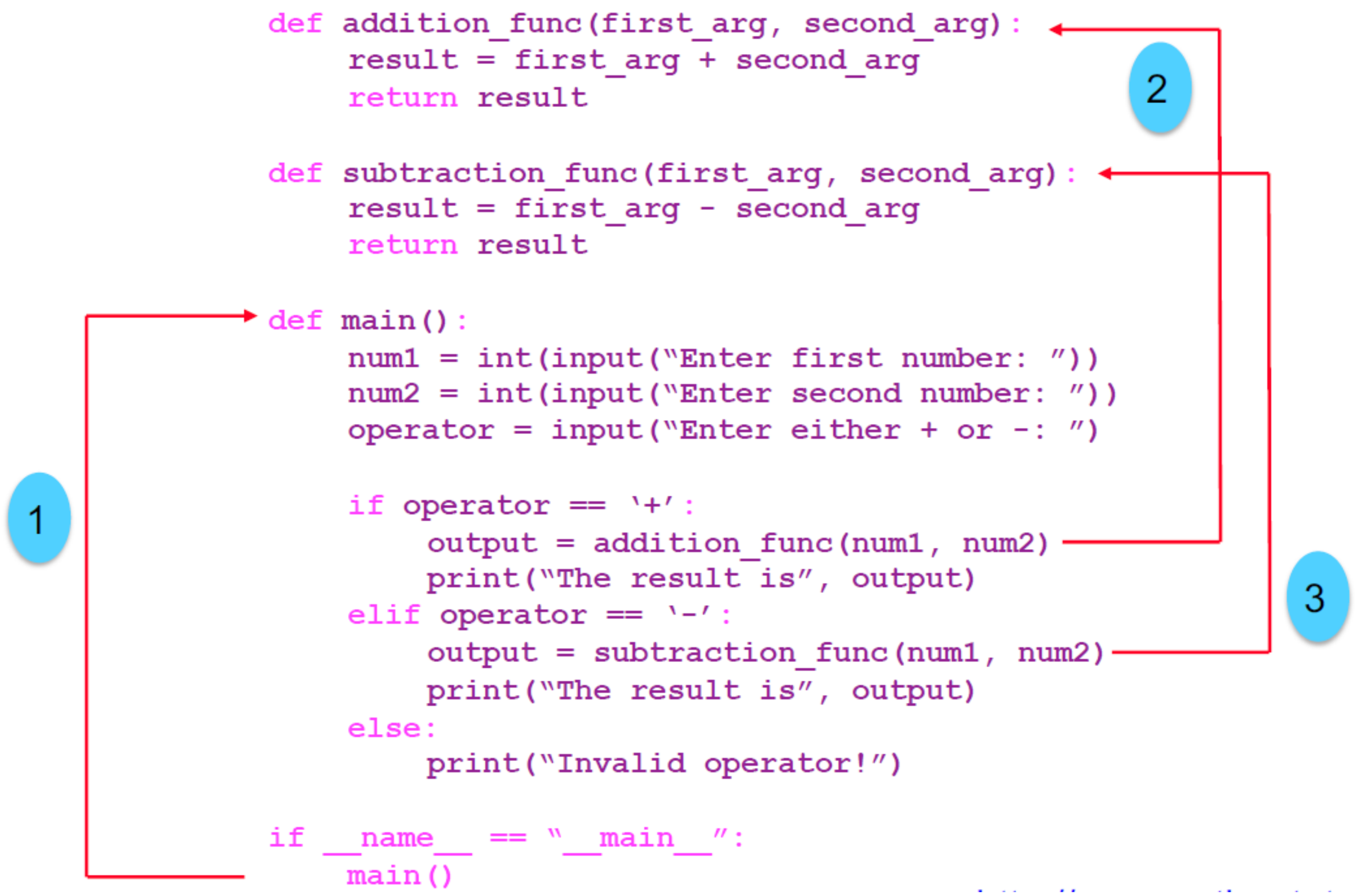
return

- Return only has meaning inside a function
- Only one return executed inside a function
- Code inside function but after return statement not executed
- Has a value associated with it, given to function caller

print

- print can be used outside functions
- Can execute many print statement inside a function
- Code inside function can be executed after a print statement
- Has a value associated with it, outputted to the console

FIT9136平时班 – Week 1



```
def addition_func(first_arg, second_arg):
    result = first_arg + second_arg
    return result

def subtraction_func(first_arg, second_arg):
    result = first_arg - second_arg
    return result

def main():
    num1 = int(input("Enter first number: "))
    num2 = int(input("Enter second number: "))
    operator = input("Enter either + or -: ")

    if operator == '+':
        output = addition_func(num1, num2)
        print("The result is", output)
    elif operator == '-':
        output = subtraction_func(num1, num2)
        print("The result is", output)
    else:
        print("Invalid operator!")

if __name__ == "__main__":
    main()
```