SCHOOL OF ENGINEERING, KMITL

//WK03

# Advance Computer Programming

For RAI

Topic of
intermediate
Python Programming

Dr. Sarucha Yanyong,
Department of Robotics
and AI Engineering

# Conditions

if

Syntax of an **if** statement

```
if some_condition:
    # execute some code
```

```
if 3>2:
    print('ITS TRUE!')
```

# Conditions

if..else

Syntax of an **if/else** statement

```
if some_condition:
    # execute some code
else:
    # do something else
```

```
hungry = True

if hungry==True:
    print('FEED ME!')
else:
    print("Im not hungry")
```

```
FEED ME!
```

Advance
Computer
Programming
for RAI

# Conditions

if...elif...else

```
loc = 'Bank'

if loc == 'Auto Shop':
    print("Cars are cool!")
elif loc == 'Bank':
    print("Money is cool!")
elif loc == 'Store':
    print("Welcome to the store!")
else:
    print("I do not know much.")
```

```
Money is cool!
```

Advance
Computer
Programming
for RAI

# Generators

using range() with list

# list

using range()

## Purpose

The range() function in Python generates a sequence of numbers, which is often used for iterating over with loops, particularly in for loops, or for creating number sequences in list comprehensions.

## Syntax

```
range(stop)
range(start, stop)
range(start, stop, step)
```

# list

using range()

**Example:** Basic usage

```
for i in range(5):
 print(i)
# Output: 0, 1, 2, 3, 4
```

**Example:** Specifying start and stop

```
for i in range(2, 7):
    print(i)
# Output: 2, 3, 4, 5, 6
```

**Example:** Specifying start, stop and step

```
for i in range(1, 10, 2):
    print(i)
# Output: 1, 3, 5, 7, 9
```

Advance
Computer
Programming
for RAI

# Loops

for

Syntax of a for loop

```
my_iterable = [1,2,3]
for item_name in my_iterable:
    print(item_name)
```

```
for i in range(0, 5):
    print(i)
```

Advance
Computer
Programming
for RAI

# Loops

while

## Syntax of a while loop

```
while some_boolean_condition:
    #do something
```

```
i = 0
while i < 5:
    print(f"No. is {i}")
    i = i + 1
```

```
No. is 0
No. is 1
No. is 2
No. is 3
No. is 4
```

Advance
Computer
Programming
for RAI

# Loops

while

## Syntax of a while loop

```
while some_boolean_condition:
    #do something
```

```
i = 0
while i < 5:
    print(f"No. is {i}")
    i = i + 1
```

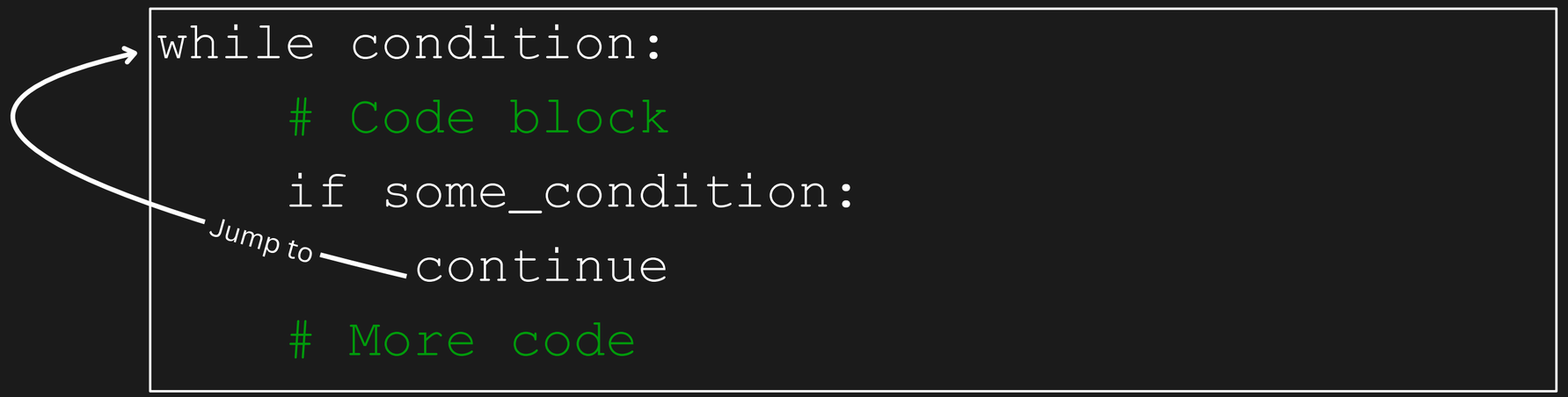
```
No. is 0
No. is 1
No. is 2
No. is 3
No. is 4
```

# Loops continue

The while loop allows repeated execution of a **block of code as long** as a specified condition is true. The **continue** statement is used within loops to **skip** the current iteration and continue with the next iteration of the loop.

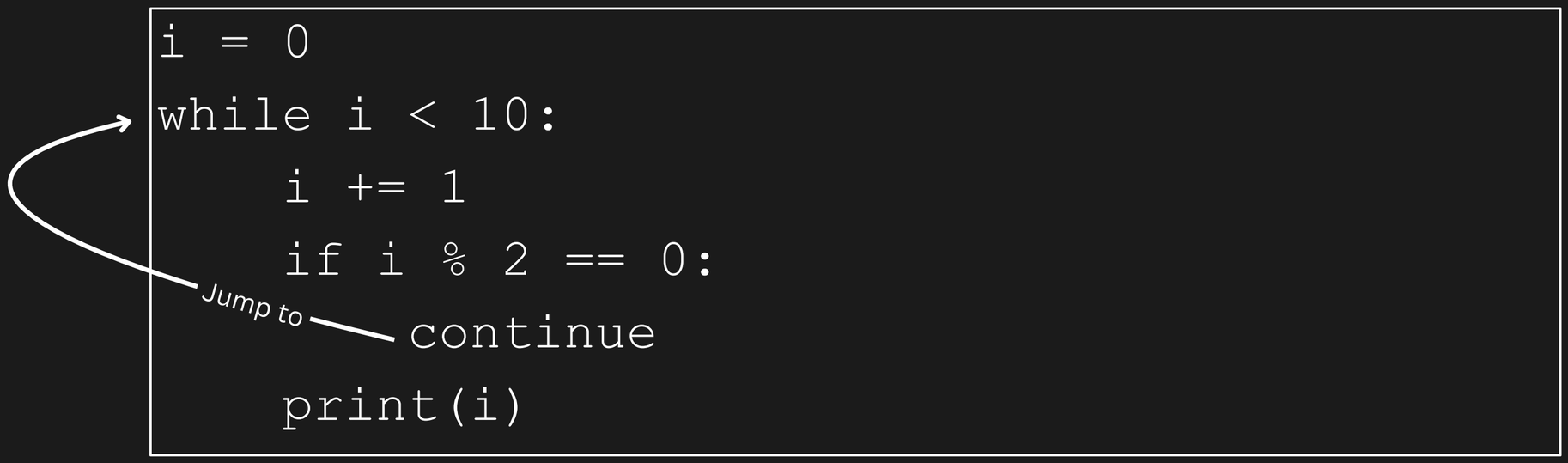
# Loops continue

syntax



# Loops continue

example




```
i = 0
while i < 10:
    i += 1
    if i % 2 == 0:
        continue
    print(i)
```

In this example, the while loop runs until i is less than 10. The continue statement skips the even numbers, so only odd numbers are printed.

# Loops break

The while loop allows repeated execution of a block of code as long as a specified condition is true. The **break** statement is used to exit the loop immediately, regardless of the original loop condition.

# Loops break

syntax

```
while condition:
    # Code block
    if some_condition:
        break
    # More code
# Outside loop
```

# Loops break

example

```
i = 0
while i < 10:
    print(i)
    if i == 5:
        print("Breaking out of the loop")
        break
    i += 1
print("Outside the loop")
```


In this example, the while loop runs until i is less than 10. The break statement exits the loop when i equals 5, and control jumps to the code after the loop, indicated by the print statement "Outside the loop".

Advance
Computer
Programming
for RAI

# list comprehensions

List comprehensions in Python provide a concise way to create lists. They consist of brackets containing an expression followed by a for clause, and can include additional for or if clauses for filtering.

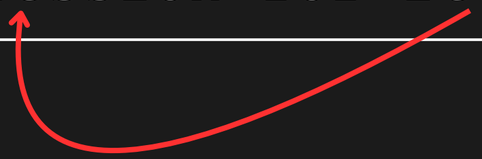
# list

## comprehensions (Element modify)

Syntax

Syntax

```
iterable_out = [expression for item in iterable]
```



expression is the value that will be added into iterable_out. It’s also allow to modify value into add.

# list
## comprehensions (Element modify)

Example

**Example 1: modify value to square the number into new list**

```
numbers = [1, 2, 3, 4, 5]
squares = [n**2 for n in numbers]
# squares = [1, 4, 9, 16, 25]
```

# list

## comprehensions (Element modify with if-else)

Syntax

**Syntax**

```
iterable_out = [expression for item in iterable]
```

```
iterable_out = [exp_true if cond else exp_false for item in iterable]
```

** This technique combines the shorthand if-else and list comprehensions.

# list

## comprehensions (Element modify with if-else)

Example

```
numbers = [1, 2, 3, 4, 5]
parity = ["even" if n % 2 == 0 else "odd" for n in numbers]

# output
# parity = ["odd", "even", "odd", "even", "odd"]
```

# list

## comprehensions (Filter element)

Syntax

Syntax

```
iterable_out = [expression for item in iterable if condition]
```

If the condition is correct, the item will be pass to expression.

expression is the value that will be added into iterable_out. It’s also allow to modify value into add.

# list

## comprehensions (Filter element)

Example

**Example 1: filter the even numbers**

```
numbers = [1, 2, 3, 4, 5]
even_numbers = [n for n in numbers if n % 2 == 0]


# even_numbers = [2, 4]
```

# list

## comprehensions (Search in element)

Syntax/Example

**Example**

```
fruits = ["apple", "banana", "cherry"]
is_contains_apple = "apple" in fruits
print (is_contains_apple)

# Output is True
```

** The variable is_contains_apple will be True if "apple" is in the list, otherwise False

Advance
Computer
Programming
for RAI

# Function

# def

;

## Purpose

The def keyword is used to define a function in Python. Functions allow you to encapsulate reusable blocks of code, making your programs modular and easier to manage.

## Syntax

```
def function_name(parameters):
    # Code block
    return value (optional)
```

# def

;

## Example

```
def greet(name):
    message = f"Hello, {name}!"
    return message

# Usage
print(greet("Alice")) # Output: Hello, Alice!
```

In this example, the greet function takes a name parameter, constructs a greeting message, and returns it. The function is then called with the argument "Alice".

Advance
Computer
Programming
for RAI

# Anonymous Functions

lambda functions, also known as anonymous functions, allow the creation of small, unnamed functions for short-term use.

# lambda

;

## Syntax

```
lambda arguments: expression
```

## Regular function VS Lambda function

```
# Regular function
def add(x, y):
    return x + y
```

```
# Lambda function
add = lambda x, y: x + y
```

** x, y are defined as input argument for function add.

```
# Regular function
def square(x):
    return x*x
```

```
# Lambda function
square = lambda x: x*x
```

** x is defined as input argument for function square.

# lambda

Use cases

## Using lambda with map()

To modify element in list with specific algorithm

## Example

```
numbers = [1, 2, 3, 4, 5]
squared = list(map(lambda x: x**2, numbers))
# squared = [1, 4, 9, 16, 25]
```

# lambda

Use cases

## Using lambda with filter()

To filter element in list with specific algorithm

## Example

```
numbers = [1, 2, 3, 4, 5]
evens = list(filter(lambda x: x % 2 == 0, numbers))
# evens = [2, 4]
```

# lambda

Use cases

## Using lambda with sort

To sort element in list with specific algorithm

## Example

```
# List of dictionaries
students = [
    {"name": "Alice", "grade": 85},
    {"name": "Bob", "grade": 92},
    {"name": "Charlie", "grade": 78}
]
```

Sort the nesteset list of dictionary

```
# Output:
# [
# {'name': 'Charlie', 'grade': 78},
# {'name': 'Alice', 'grade': 85},
# {'name': 'Bob', 'grade': 92}
# ]
```

# lambda

Use cases

## Example

```
# List of dictionaries
students = [
    {"name": "Alice", "grade": 85},
    {"name": "Bob", "grade": 92},
    {"name": "Charlie", "grade": 78}
]

# Sorting the list of dictionaries by the 'grade' key
sorted_students = sorted(students, key=lambda student: student["grade"])

# Output the sorted list
print(sorted_students)

# Output:
# [{'name': 'Charlie', 'grade': 78}, {'name': 'Alice', 'grade': 85},
{'name': 'Bob', 'grade': 92}]
```

In this example, the lambda function lambda student: student["grade"] is used to extract the grade value from each dictionary in the list. The sorted() function then sorts the list of dictionaries based on these grade values in ascending order.

Advance
Computer
Programming
for RAI

# Done!


## Dr. Sarucha Yanyong

Department of Robotics and AI Engineering
School of Engineering, KMITL

EMAIL ADDRESS — sarucha.ya@kmitl.ac.th

WEBSITE — www.reallygreatsite.com


**Q/A**