Imperative Programming

HADAIQ ROLIS SANABILA

FAKULTAS ILMU KOMPUTER UNIVERSITAS INDONESIA

Outline

- Execution Control
- User Defined Function
- Python Variable Assignment
- Parameter Passing

Execution Control

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia
FACULTY OF COMPUTER SCIENCE

- Boolean expression / condition
  - Associated with statements like a selection or repetition
  - Has True and False value
  - If True
    - Execute one sequence of statement
  - If False
    - Execute different sequence of statement

Execution Control

- Example of Boolean expression
  - X>5
  - 1<y<5
  - Z != “B”

python	Math notation	Meaning
<	<	Less than
<=	≤	Less than or equal to
==	=	Equal to
>=	≥	More than or equal to
>	>	More than
!=	≠	Not equal to

Execution Control

- If Statement

```
If <condition>:
    <body>
```

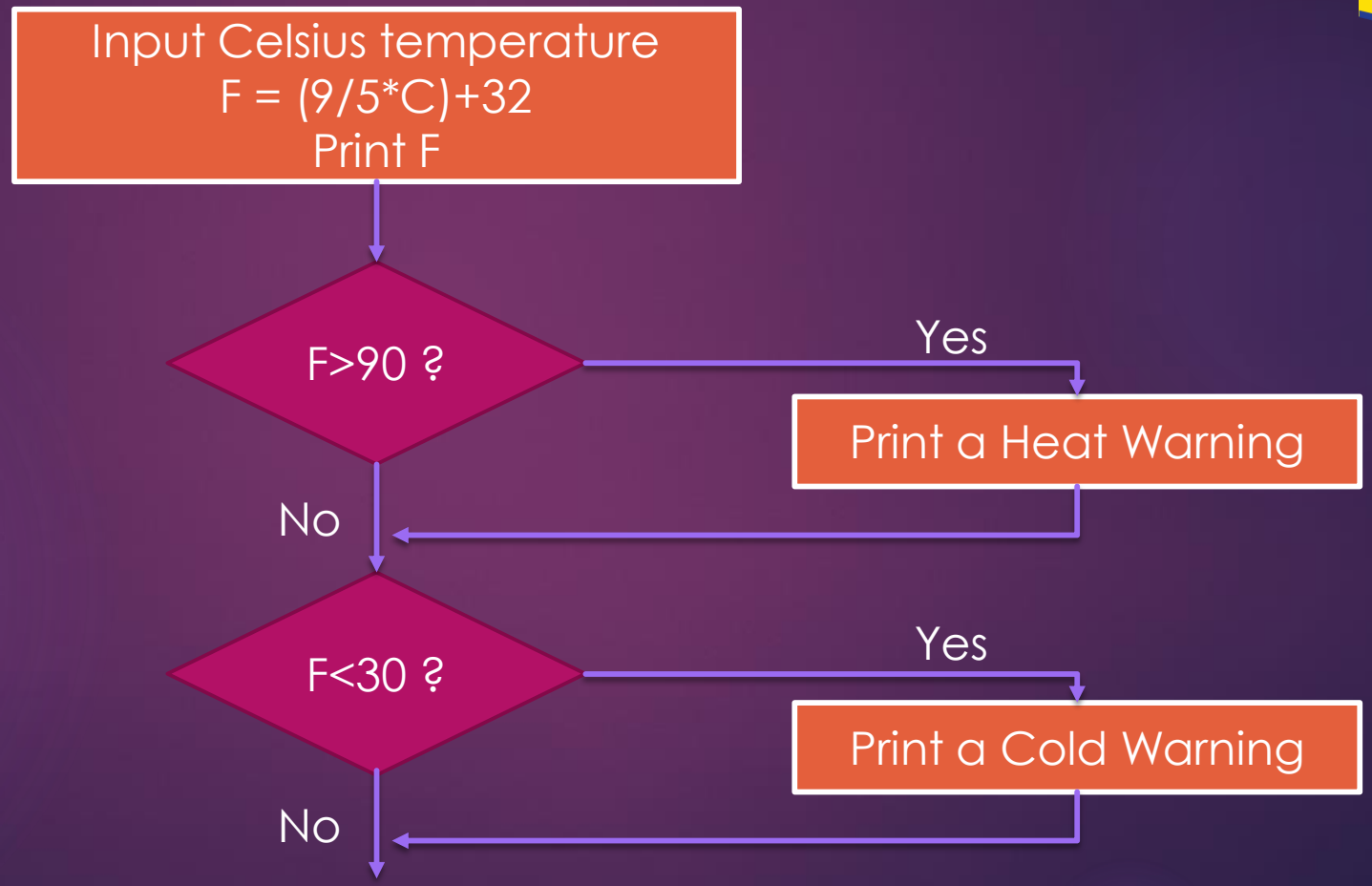
- `<body>` is a sequence of one or more statements indented under the if heading
- `<condition>` is a Boolean expression

Execution Control

- Warning about indentation
  - The statements must all be indented the same number of spaces/tabs
  - Do not mix between spaces and tabs for the indentation
  - Mostly, python programmer using 4 spaces for the indentation

Execution Control

- Example: Temperature Warning



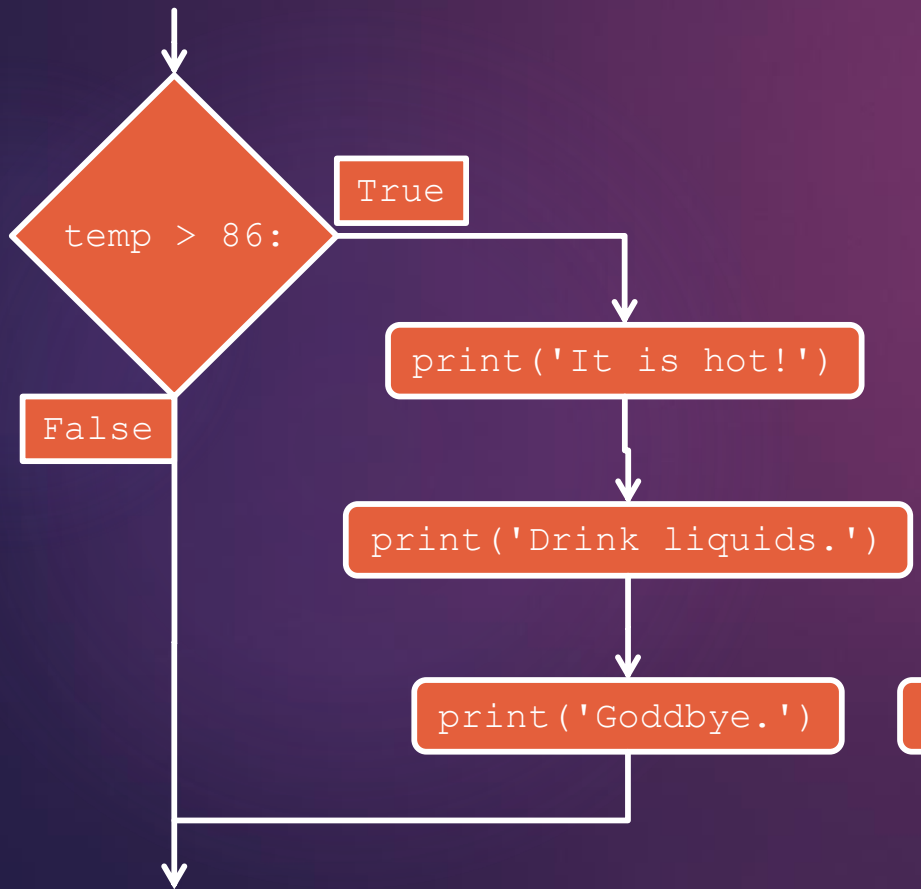
Execution Control

- Example: Temperature Warning

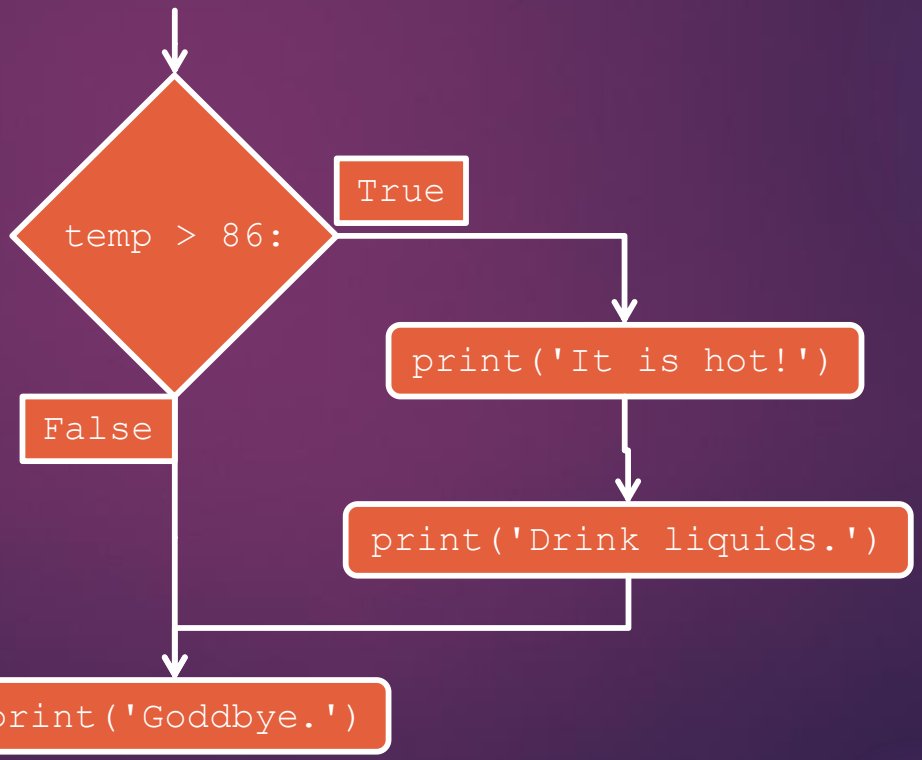
```
C = 50
F = 9/5*C +32
Print ('The temperature is', F, 'degrees F')
If (F> 90)
    print ('Its really hot out there, Be careful!')
If (F < 30)
    print ('Brrrr.. Be sure dress warmly')
```

Indentation is Critical

```
if temp > 86:
    print('It is hot!')
    print('Drink liquids.')
    print('Goodbye.')
```




```
if temp > 86:
    print('It is hot!')
    print('Drink liquids.')
print('Goodbye.')
```

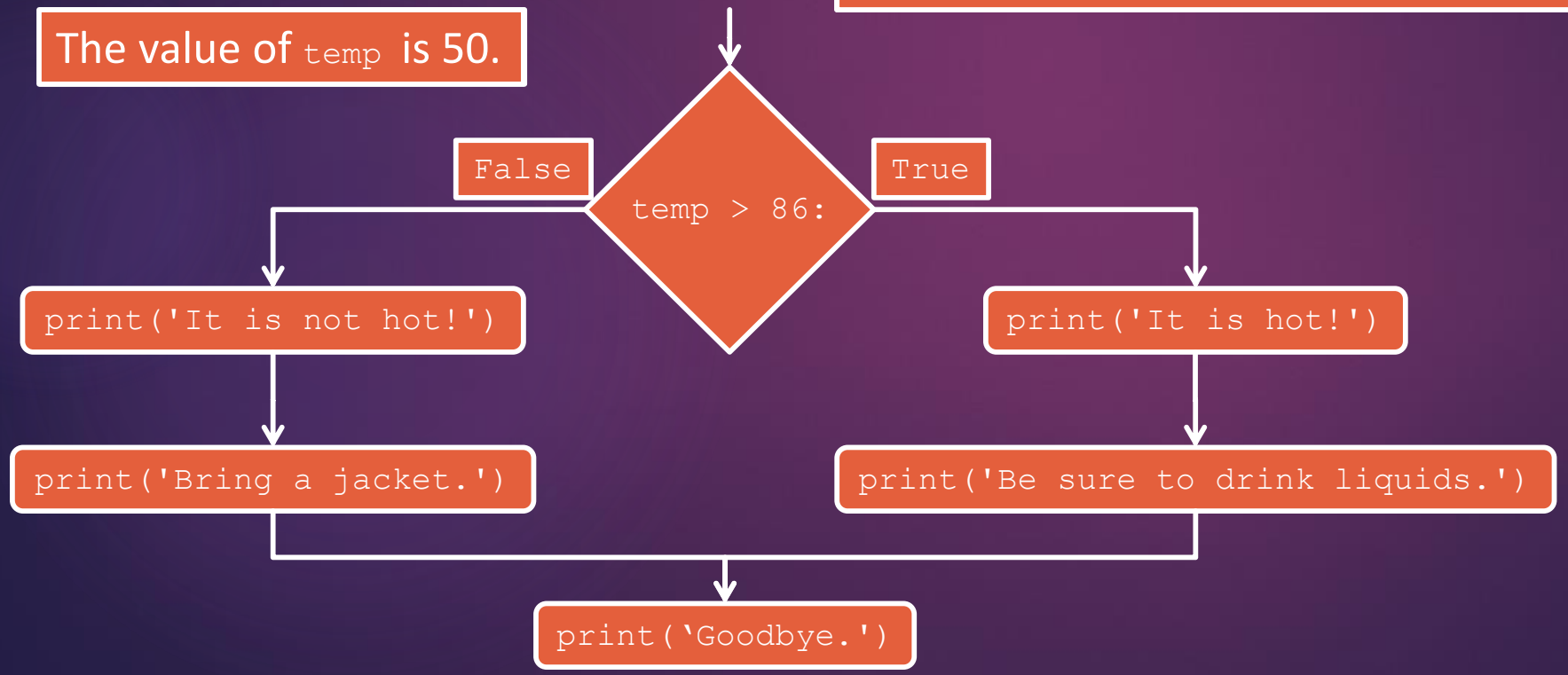


Two-way if statement

```
if <condition>:
    <indented code block 1>
else:
    <indented code block 2>
<non-indented statement>
```

```
if temp > 86:
    print('It is hot!')
    print('Be sure to drink liquids.')
else:
    print('It is not hot.')
    print('Bring a jacket.')
print('Goodbye.')
```

The value of `temp` is 50.



Multi-way if statement

```
if <condition1>:
   <body1>
elif <condition2>:
   <body2>
elif <condition3>:
   <body3>
...
else :
   <default statements>
```

Multi-way if statement

- This form mutually exclusive code blocks
- Python evaluates each condition in turn looking for the first one that is true
  - If true condition is found
    - The statements indented under that condition is executed
  - The control passes to the next statement after the entire `if-elif-else`
- If none are true, the statements under `else` are executed
- The `else` is optional. If there is no `else`, its possible no indented block will be executed

Repetition statements

- Beside selecting which statements to execute, a fundamental need in a program is repetition
  - Repeat set of statements under some condition
- With both selection and repetition, we have two most necessary programming statement

While and for statement

- The `while` statement is more general repetition construct. It repeats a set of statements while some condition is True
- The `for` statement is useful for iteration, moving through all the elements of data structure, one at time.

While Loop

- Top-tested loop
  - Test the Boolean expression before running
  - Test the Boolean expression before each iteration of the loop

```
while <boolean expression>:
    body
```

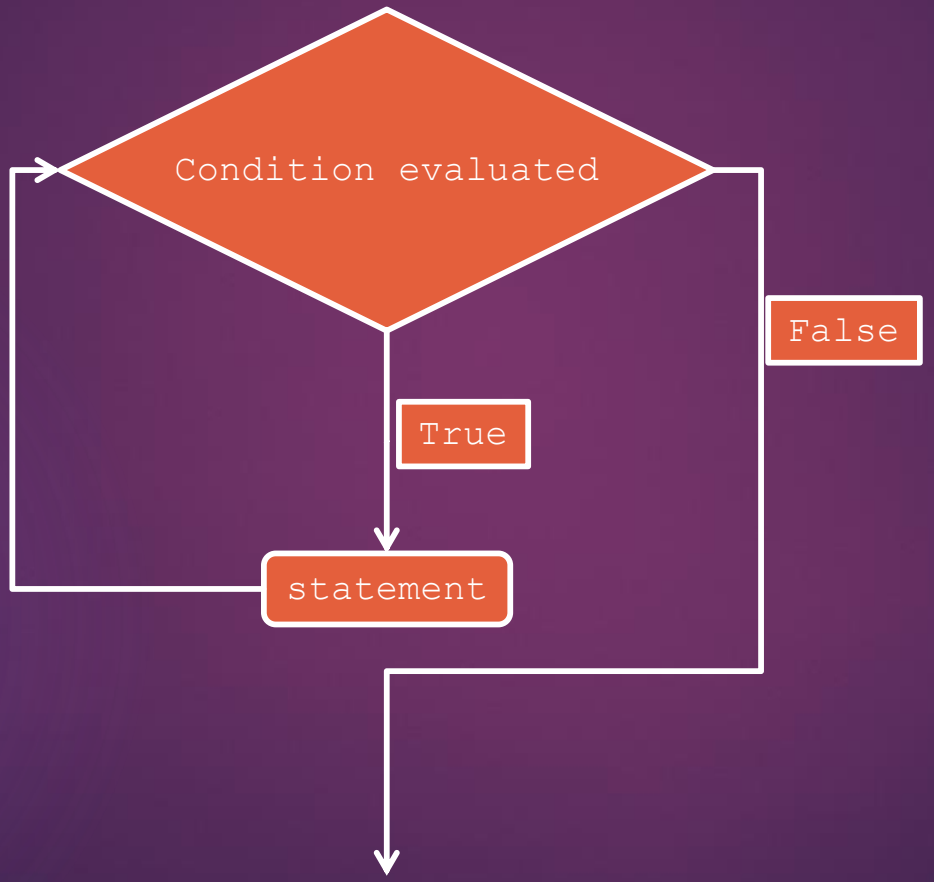
While Loop

- A `while` statement executes a block of code repeatedly
- A condition controls how many times the loop is executed

```
while <boolean expression>:
    body
```

- It means
  - Evaluate the condition
  - If the condition true, execute the statement and then go to step 1 again. If the condition is false, the loop is finished

Logic of While loop



While Loop

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia
FACULTY OF COMPUTER SCIENCE

- Example

```
x= 5
while x>0:
    print (x, end='')
    x -= 1
print (x)
```

Output:

5 4 3 2 1
0

While Loop

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia
FACULTY OF COMPUTER SCIENCE

- Example

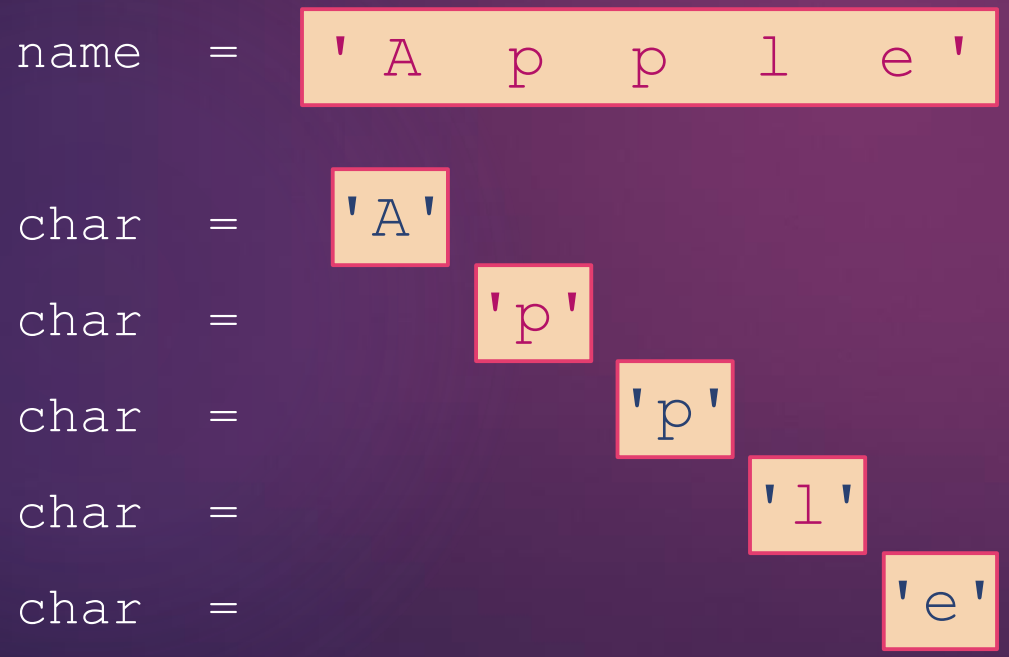
```
x= 5
while x>5:
    print (x, end='')
    x -= 1
print (x)
```

Output:

5

for Loop

- Executes a block of code for every item of a sequence
  - If sequence is a string, items are its characters (single-character strings)




```
>>> name = 'Apple'
>>> for char in name:
        print(char)


A
p
p
l
```

for Loop

- for loop format

```
for <variable> in <sequence>:
    <indented code block >
<non-indented code block>
```

```
for word in ['stop', 'desktop', 'post', 'top']:
    if 'top' in word:
        print(word)
print('Done.')
```

word = 'stop'

word = 'desktop'

word = 'post'

word = 'top'

```
>>>
stop
desktop
top
Done.
```

Exercise

- Write a "spelling" program that:
  - Requests a word from the user
  - Prints the characters in the word from left to right, one per line

```
name = input('Enter a word: ')
print('The word spelled out: ')

for char in name:
    print(char)
```

```
>>>
Enter a word: omnipotent
The word spelled out:
o
m
n
i
p
o
t
e
n
t
```

Loop Built in function `range()`

- Function range() is used to iterate over a sequence of numbers in a specified range

- To iterate over the n numbers 0, 1, 2, …, n-1
  ```
  for i in range(n):
  ```

- To iterate over the n numbers i, i+1, i+2, …, n-1
  ```
  for i in range(i, n):
  ```

- To iterate over the n numbers i, i+c, i+2c, i+3c, …, n-1
  ```
  for i in range(i, n, c):
  ```

```
>>> for i in range(2, 16, 10):
        print(i)

2
12
```

Exercise

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia
FACULTY OF COMPUTER SCIENCE

- Write for loops that will print the following sequences:

a) 0, 1, 2, 3, 4, 5, 6, 7, 8 , 9, 10
b) 1, 2, 3, 4, 5, 6, 7, 8, 9
c) 0, 2, 4, 6, 8
d) 1, 3, 5, 7, 9
e) 20, 30, 40, 50, 60

Exercise

- Write a while or for loop for statement to print the following

```
*
**
***
****
*****
******
*******
********
```

`While` loop with else

- `while` loop can have an associated `else` block
- `else` block is executed when the loop finishes under normal conditions (i.e. without hitting a break)
  - Basically the last thing the loop does as it exits

```
while <boolean expression>:
    <main-statement>
else:
    <else statement>
```

`While` loop with else

- Example

```
x= 5
while x>5:
    print (x, end='')
    x -= 1
else:
    print ("The Final value of i:", i)
```

Output:

10 9 8 7 6 5 4 3 2 1 The Final value of i:0

`While` loop with break statement

- `break` statement in a loop, if executed, exits the loop
- It jumps out of the closest enclosing loop
- It exits immidiaetly, skipping whatever remains of the loop as well as the else statement (if it exits) of the loop

`While` loop with break

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia
FACULTY OF COMPUTER SCIENCE

- Example

```
for in range (1,11):
    if i==5: break
print (i, end="")
```

Output:

1 2 3 4

`While` loop with continue statement

- `continue` statement means to immidieatly jump back to the top of the closest enclosing loop and re-evaluate the condition
- Any remaining parts of the loop are skipped for the one iteration when the continue was executed

`While` loop with break

- Example

```
for in range (1,11):
    if i==5: continue
print (i, end="")
```

Output:

1 2 3 4 6 7 8 9 10

Function

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia
FACULTY OF COMPUTER SCIENCE

- Function
  - Device that groups a set of statements so they can be run more than once in a program
  - Can compute a result value and let us specify parameters that serve as function inputs and may differ each time the code is run
  - Coding an operation as a function makes it a generally useful tool, which we can use in a variety of contexts
  - The most basic program structure python provide for maximizing code reuse

User Defined Function

- A function definition looks like this:

```
def <function name> (<parameter>):
    <body>
```

`def`: function definition keyword

`f`: name of function

`x`: variable name for input argument

```
def f(x):
    res = x**2 + 10
    return res
```

`return`: specifies function output

Function that return values

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia
FACULTY OF COMPUTER SCIENCE

- This function return the square of number

```
def square(x):
    return x*x
```

- When python encounter `return`, it exits the function and returns control to the point where the function was called
- In addition, the value provided in the `return` statement are sent back to the caller as an expression result

Function that return values


```
>>>square (3)
     9
>>> print (square(4))
     16
>>> x = 5
>>> y = square (x)
>>> print (y)
     25
>>> print (square(x)+square(4))
     34
```

Function that return values

- All python function `return` a value, whether they contain a `return` value statement or not
- Function without a `return` hand back a special object, denoted None
- If your value-returning functions produce strange message, check to make sure that you remembered to include `return`!

return vs. print

```
def f(x):
    res = x**2 + 10
    return res
```

```
>>> f(2)
14
>>> 2*f(2)
28
```

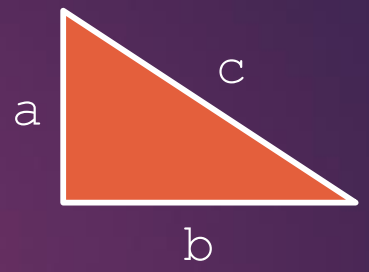
Function returns value of `res` which can then be used in an expression

```
def f(x):
    res = x**2 + 10
    print(res)
```

```
>>> f(2)
14
>>> 2*f(2)
14
Traceback (most recent call last):
  File "<pyshell#56>", line 1, in
<module>
    2*f(2)
TypeError: unsupported operand
type(s) for *: 'int' and
'NoneType'
```

Function prints value of `res` but does not return anything

User defined function



Let's develop function `hyp()` that:

- Takes two numbers as input (side lengths a and b of above right triangle)
- Returns the length of the hypotenuse c

```
>>> hyp(3,4)
5.0
>>>
```

```
import math
def hyp(a, b):
    res = math.sqrt(a**2 + b**2)
    return res
```

Exercise

Write function `hello()` that:

- takes a name (i.e., a string) as input
- prints a personalized welcome message

Note that the function does not return anything

```
>>> hello('Julie')
Welcome, Julie, to the world of Python.
>>>
```

```
def hello(name):
    line = 'Welcome, ' + name + ', to the world of Python.'
    print(line)
```

Exercise

Write function `rng()` that:

- takes a list of numbers as input
- returns the range of the numbers in the list

The range is the difference between the largest and smallest number in the list

```
>>> rng([4, 0, 1, -2])
6
>>>
```

```
def rng(lst):
    res = max(lst) - min(lst)
    return res
```