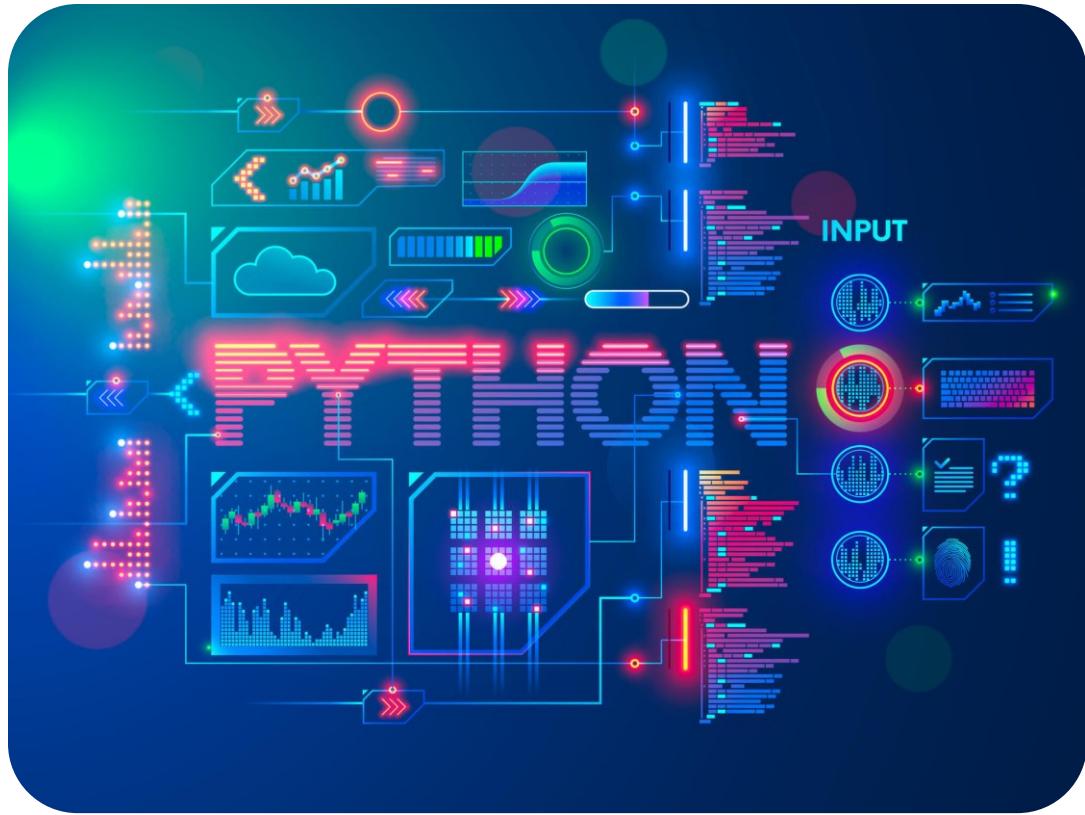


WIIT 7740: Scripting with Python

Week 5: For Loops and While Loops



Expectations Of Your Code

- Docstrings
- Clear, descriptive variable names
- Consistent variable naming convention
 - “`snake_case`” for variable names
 - “`UPPER_SNAKE_CASE`” for constants
- Follow instructions precisely
- Fix standard code style issues with PEP8 tools

```
32 self._file = None
33 self._fingerprints = set()
34 self._logdups = True
35 self._debug = debug
36 self._logger = logging.getLogger(__name__)
37 if path:
38     self._file = open(os.path.join(path, 'fingerprint.log'), 'a')
39     self._file.seek(0)
40     self._fingerprints.update(self._file.read().split())
41 @classmethod
42 def from_settings(cls, settings):
43     debug = settings.getbool('SUPERVISOR_DEBUG')
44     return cls(job_dir(settings), debug)
45
46 def request_seen(self, request):
47     fp = self.request_fingerprint(request)
48     if fp in self._fingerprints:
49         return True
50     self._fingerprints.add(fp)
51     if self._file:
52         self._file.write(fp + os.linesep)
53
54 def request_fingerprint(self, request):
55     return request_fingerprint(request)
```

Expectations Of Your Code: PEP-8 and Style Guide

Style Guide:

<https://www.python.org/dev/peps/pep-0008/>



The screenshot shows a portion of the Python PEP 8 - Style Guide for Python Code page. At the top left, there is a breadcrumb navigation bar with links to 'Python', 'PEP Index', and 'PEP 8'. Below this, the title 'PEP 8 – Style Guide for Python Code' is displayed in a large, bold, dark font. Underneath the title, there are three data entries: 'Author' (Guido van Rossum <guido at python.org>, Barry Warsaw <barry at python.org>, Nick Coghlan <ncoghlan at gmail.com>), 'Status' (Active), and 'Type' (Process). The background of the page is white, and the overall layout is clean and professional.

Python » PEP Index » PEP 8

PEP 8 – Style Guide for Python Code

Author Guido van Rossum <guido at python.org>, Barry Warsaw <barry at python.org>, Nick Coghlan <ncoghlan at gmail.com>

Status Active

Type Process



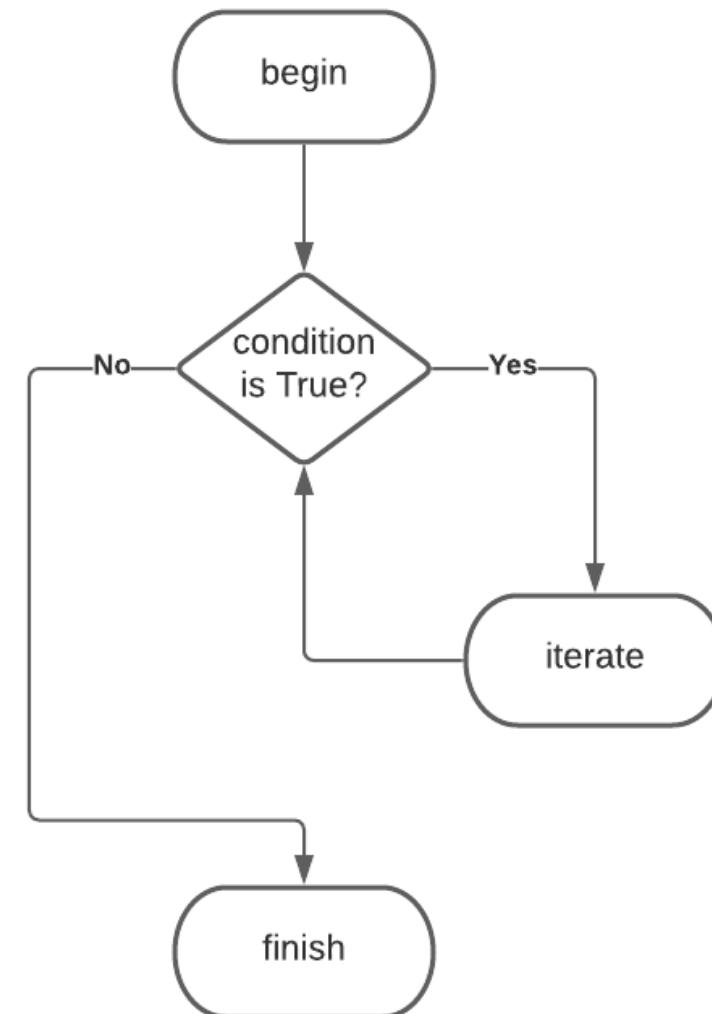
- Design mistakes into your code on purpose to see what happens
- Be patient with yourself **AND** your computer
- Use Pseudocode to write out your program in plain English
- <https://brython.info/>



“While” Loops

Used to repeat one or more lines of code (code blocks) “while” a condition holds true:

```
begin()
while condition:
    iterate()
finish()
```



“While” Loop Conditions

while condition:

happens 0 or more times

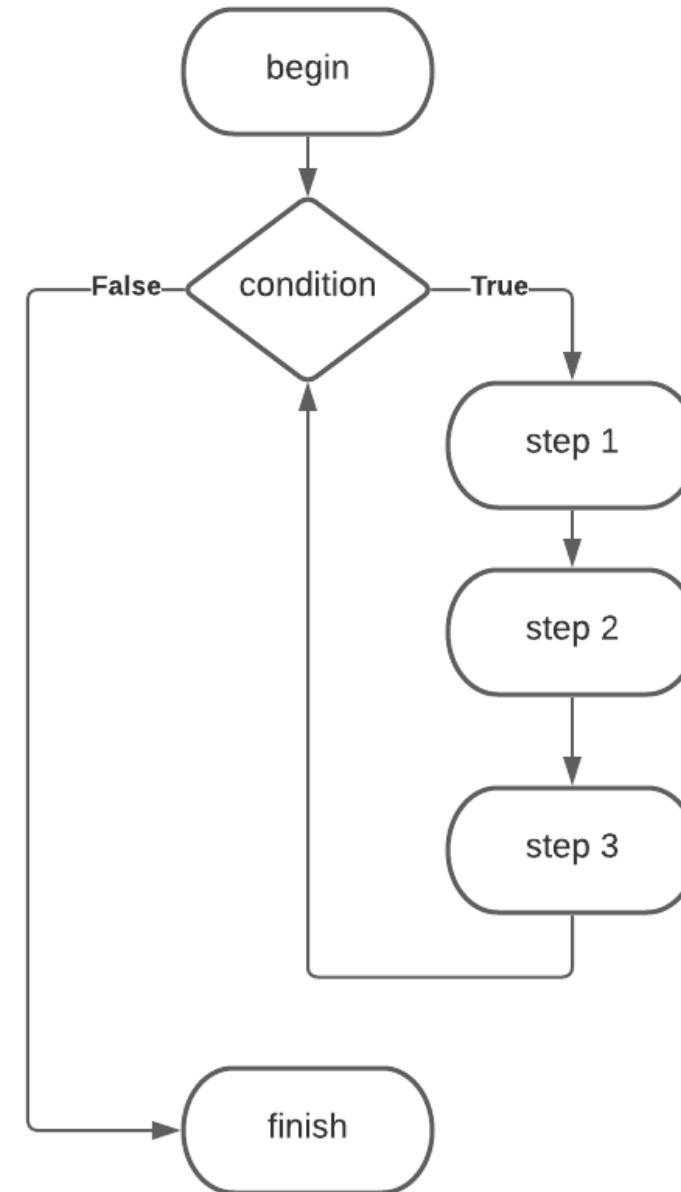
is analogous to

if condition:

happens 0 or 1 time

“While” Loop Body

```
begin()
while condition:
    do_step_1()
    do_step_2()
    do_step_3()
finish()
```



Example “While” Loop

Any expression that produces either True or False can be used:

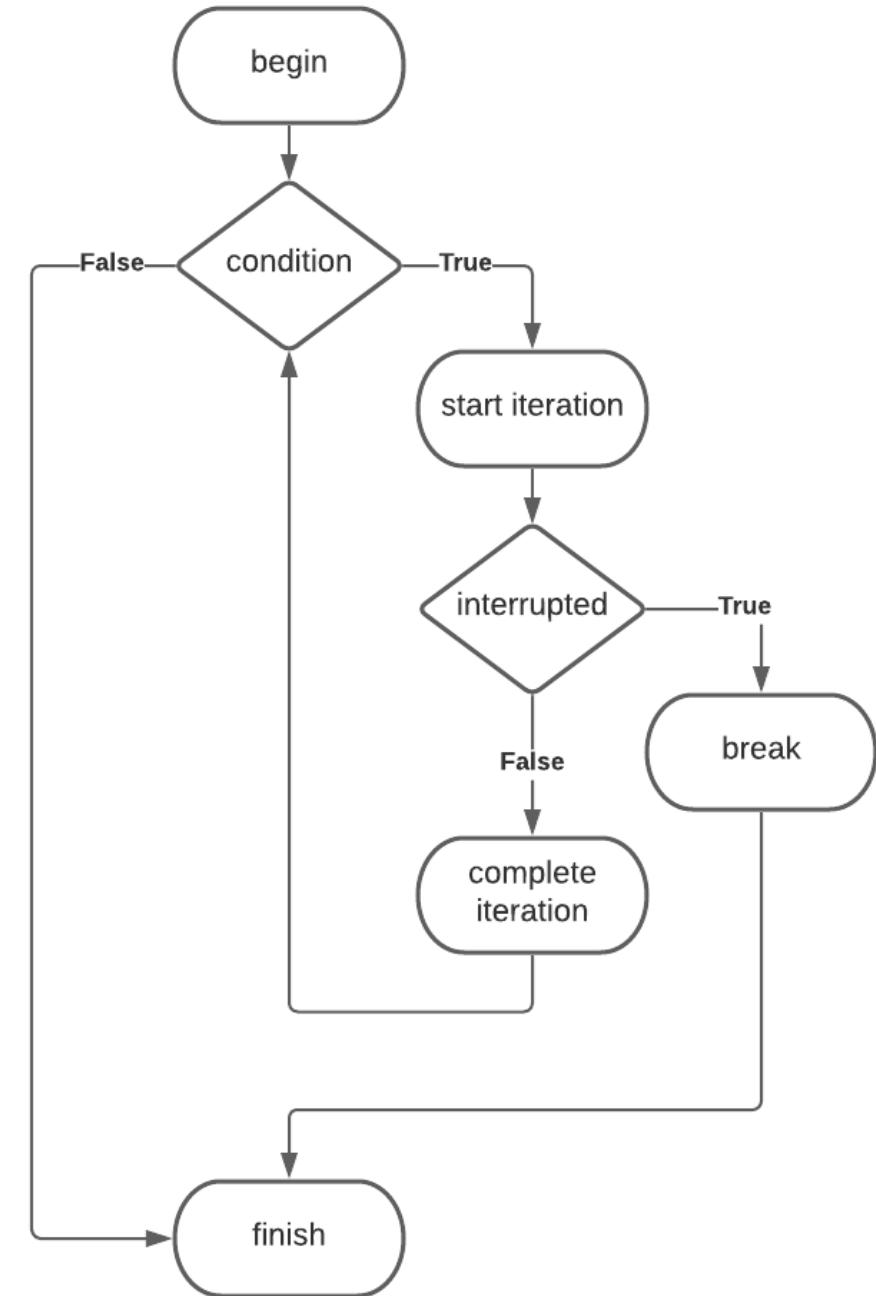
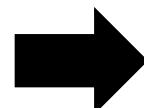
```
n = 10  
while n > 0:  
    print(n)  
    n = n - 1  
  
print('Lift off!')
```

Input Validation Loop

```
is_valid = False
while not is_valid:
    diameter = float(input('Enter diameter'))
    if diameter < 0:
        print('Invalid diameter')
    else:
        is_valid = True
# diameter is a valid value
```

Breaking out of a Loop

```
begin()
while condition:
    start_iteration()
    if interrupted:
        break
    complete_iteration()
finish()
```

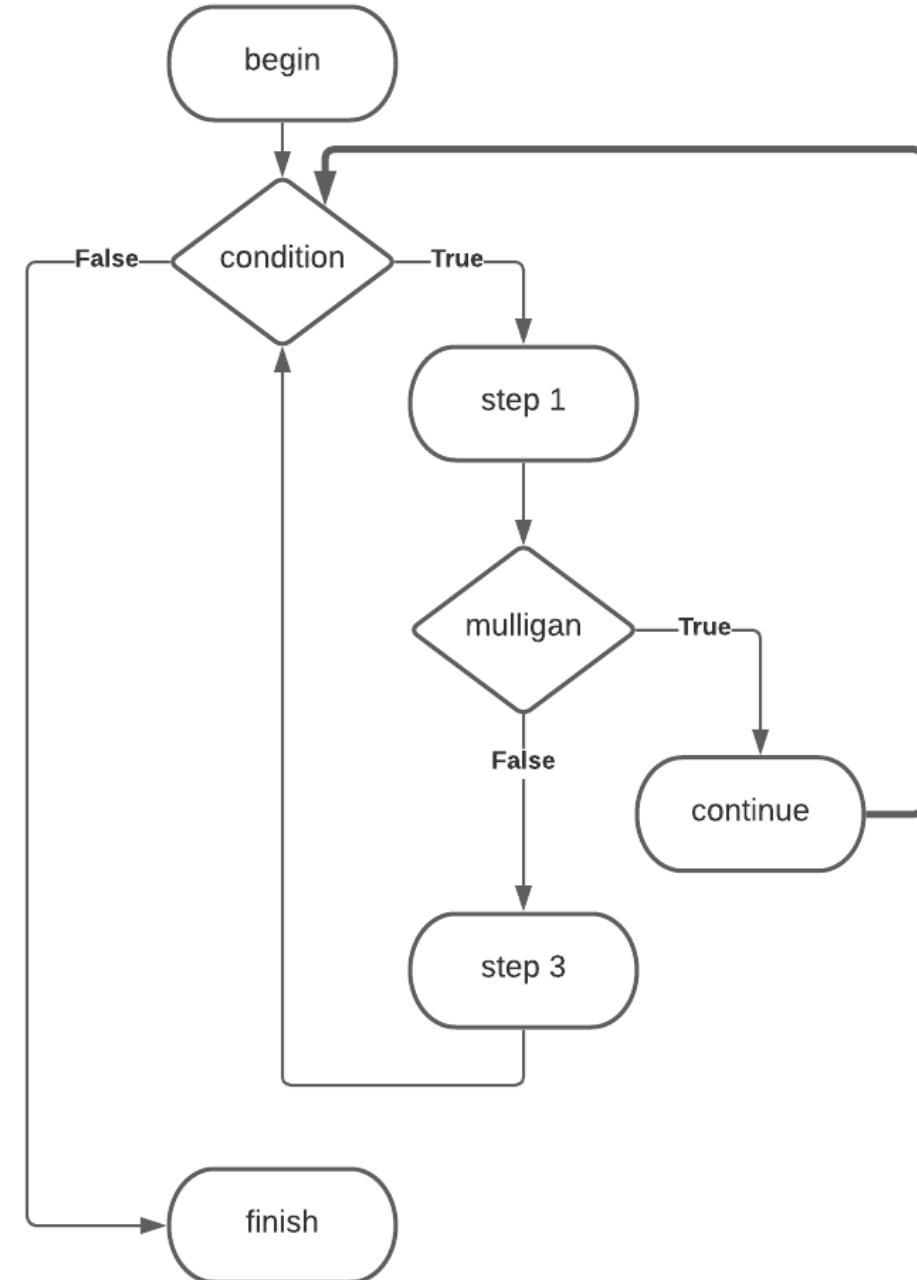


While/Else

```
before_loop()  
while condition:  
    iterate()  
else:  
    after_loop()
```

While/Else (continued)

```
begin()
while condition:
    start_iteration()
    if mulligan:
        continue
    complete_iteration()
finish()
```



Iterating List Items with “while”

```
item_index = 0
while item_index < len(items):
    item = items[item_index]
    print('list contains', item)
    item_index += 1
```

Iterating List Items with “for”

No need to deal with an index or length of the list.

```
for item in items  
    print('list contains', item)
```

Using the enumerate() function

What if you still want to refer to the index during a “for” loop?

```
for item_index, item in enumerate(items):  
    print(  
        'list contains',  
        item, 'at', item_index  
    )
```

Example: Empty List

What will this do?

```
items = []  
  
for item in items:  
    print('list contains', item)
```

Range and Range (n)

Range

Returns a range object, which can be converted to a list if needed:

`list(range(3))` will return [0, 1, 2].

Range(n)

- Start at **0**, and go up to, but do not include **n**.
- Loop directly over range, for example:

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

Range with offset

The first parameter number is inclusive, the second is **NOT** inclusive:

```
for number in range(3, 7) :  
    print(number)
```

Output:

```
3  
4  
5  
6
```

In this case, 3 is printed, but 7 is not due to this property.

Reversed Range

- `range(start, stop[, step])` allows the programmer to determine the direction and “step” of the range:

```
for number in range(5, 0, -1):  
    print(number)  
else:  
    print('Blastoff!')
```

Output:

5

4

3

2

1

Blastoff!

Nested Loops

Loops can be “nested” such that the inner loop will iterate some number of times for each iteration of the outer loop:

```
for first_name in ['Alice', 'Bob', 'Carol']: # Outer loop
    for last_name in ['Smith', 'Jones', 'Brown', 'Rogers']: # Inner loop
        print(first_name, last_name)
```

Output:
Inner loop 1st iteration → Alice Smith
Alice Jones
Alice Brown }
Outer loop 1st iteration
Inner loop 4th iteration → Alice Rogers
Bob Smith
Bob Jones }
Outer loop 3rd iteration
...
Carol Brown }
Inner loop 12th iteration → Carol Rogers

Example: Elevator

- Input starting floor S
- Input number of floors to move N
- Start at S^{th} floor (0 means basement)
- Open doors at every N^{th} floor after that
- The 20th floor is the highest
- Never stop at 13th floor
- Always retire at 7th floor

Glossary: Keywords & Function

Keywords

`for`

denotes the start of a "for" loop that will execute by iterating over each element in some object

Example: `for element in my_list:`

`while`

denotes the start of a "while" loop that will execute until a condition is met

Example: `while some_condition:`

`else`

default condition of a condition tree; can be used in conjunction with `while` to run some code after the loop has been exhausted (see example from an earlier slide)

Function

`range(start, stop, step)`

returns a sequence of numbers between `start` and `stop`

Practice Coding: Class Activity

