Advanced Python

ADVANCED FLOW CONTROL

Robert Smallshire
COFOUNDER – SIXTY NORTH
@robsmallshire rob@sixty-north.com

Advanced Python

Advanced Python

Python – Beyond the Basics

Advanced Python

Python – Beyond the Basics

Python Fundamentals

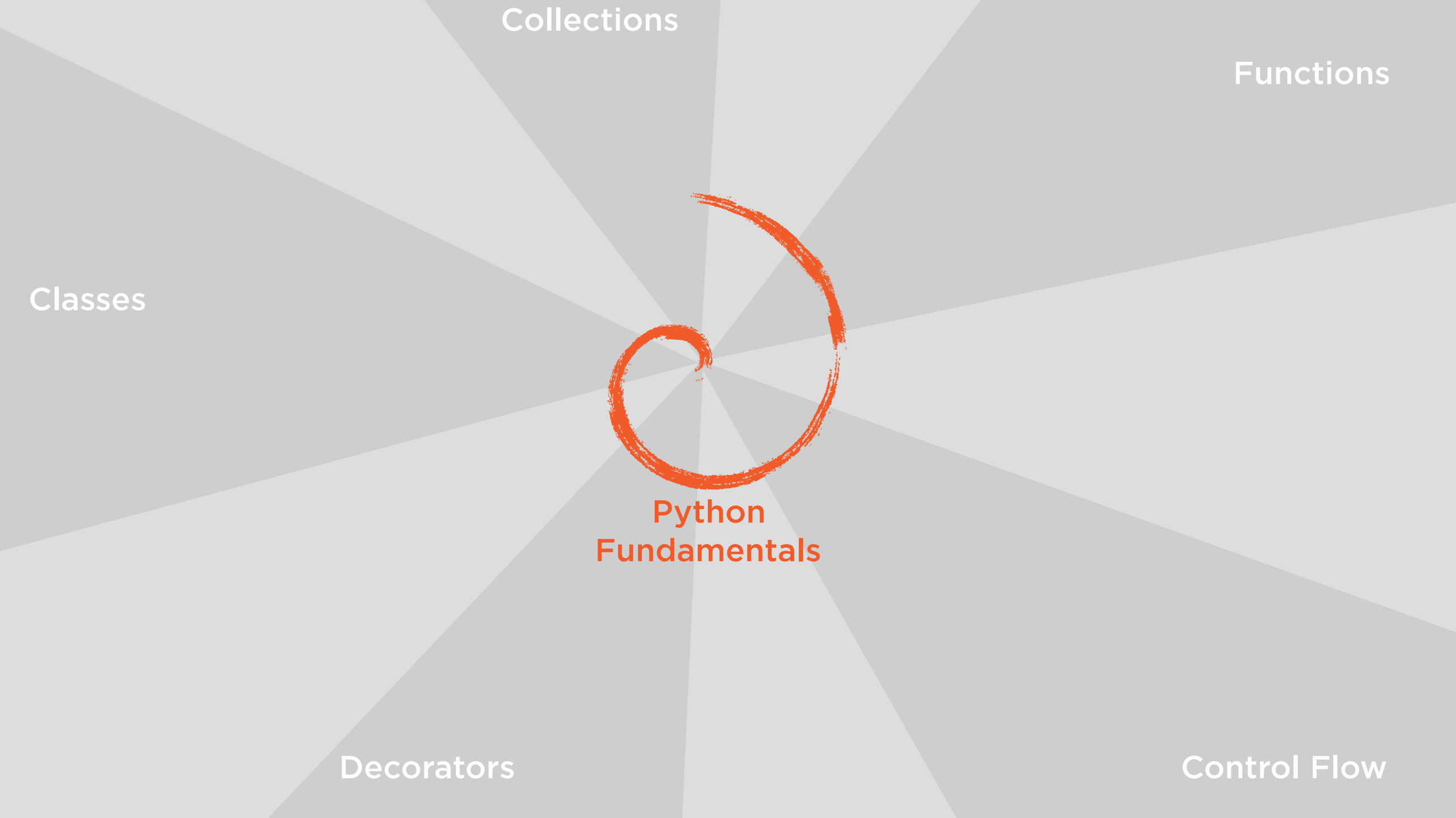
python™

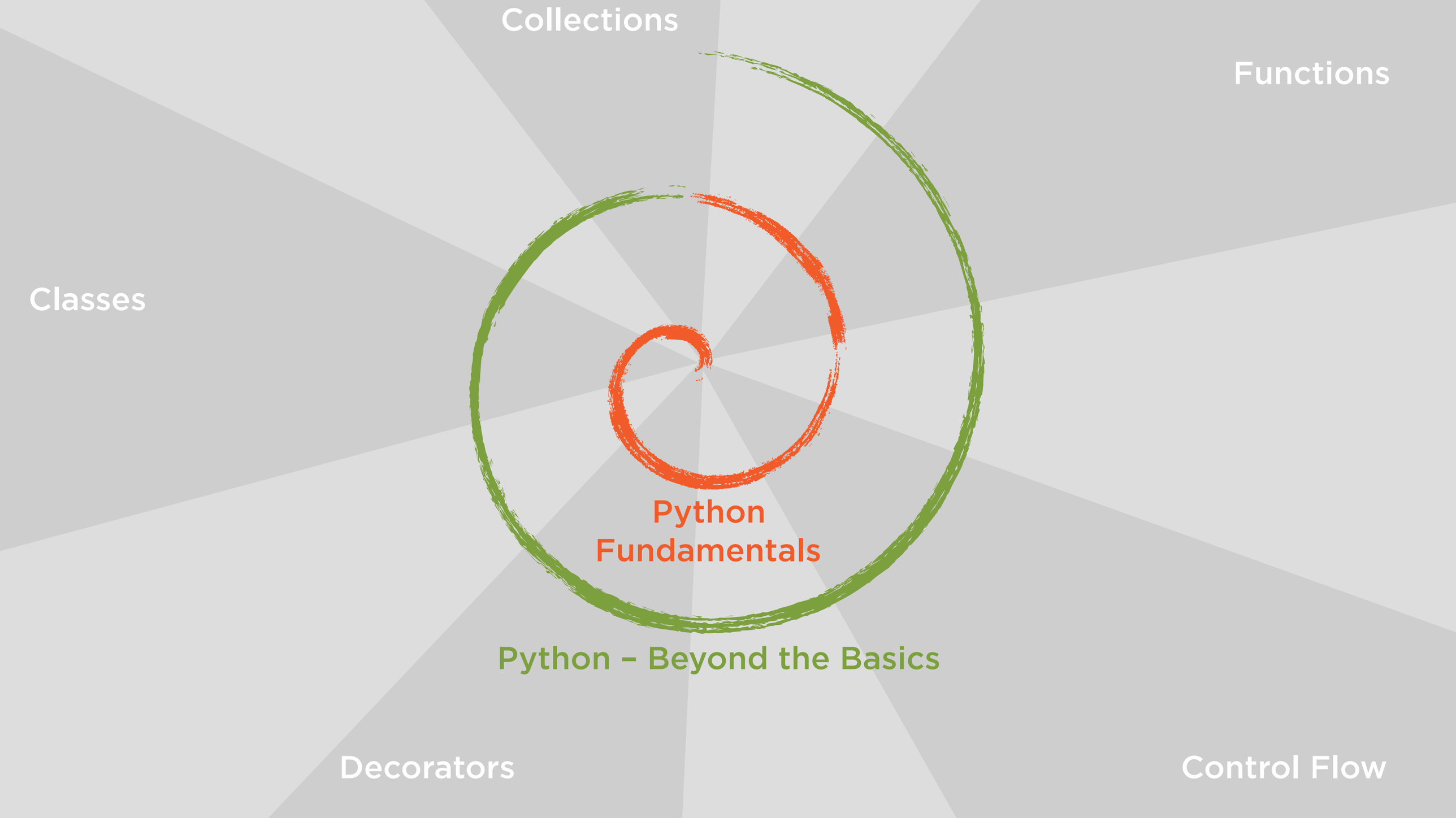
Advanced Python

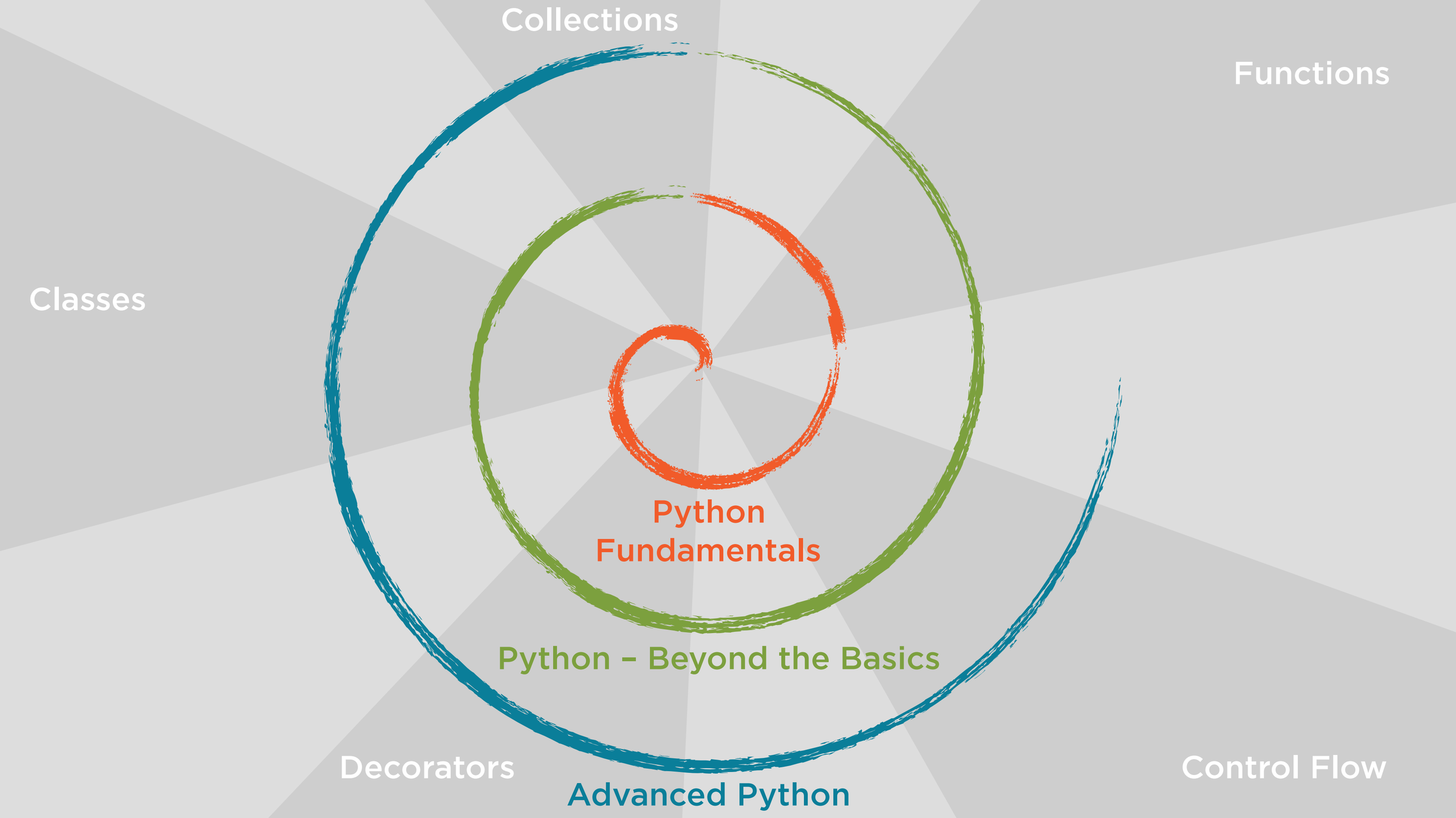
Python – Beyond the Basics

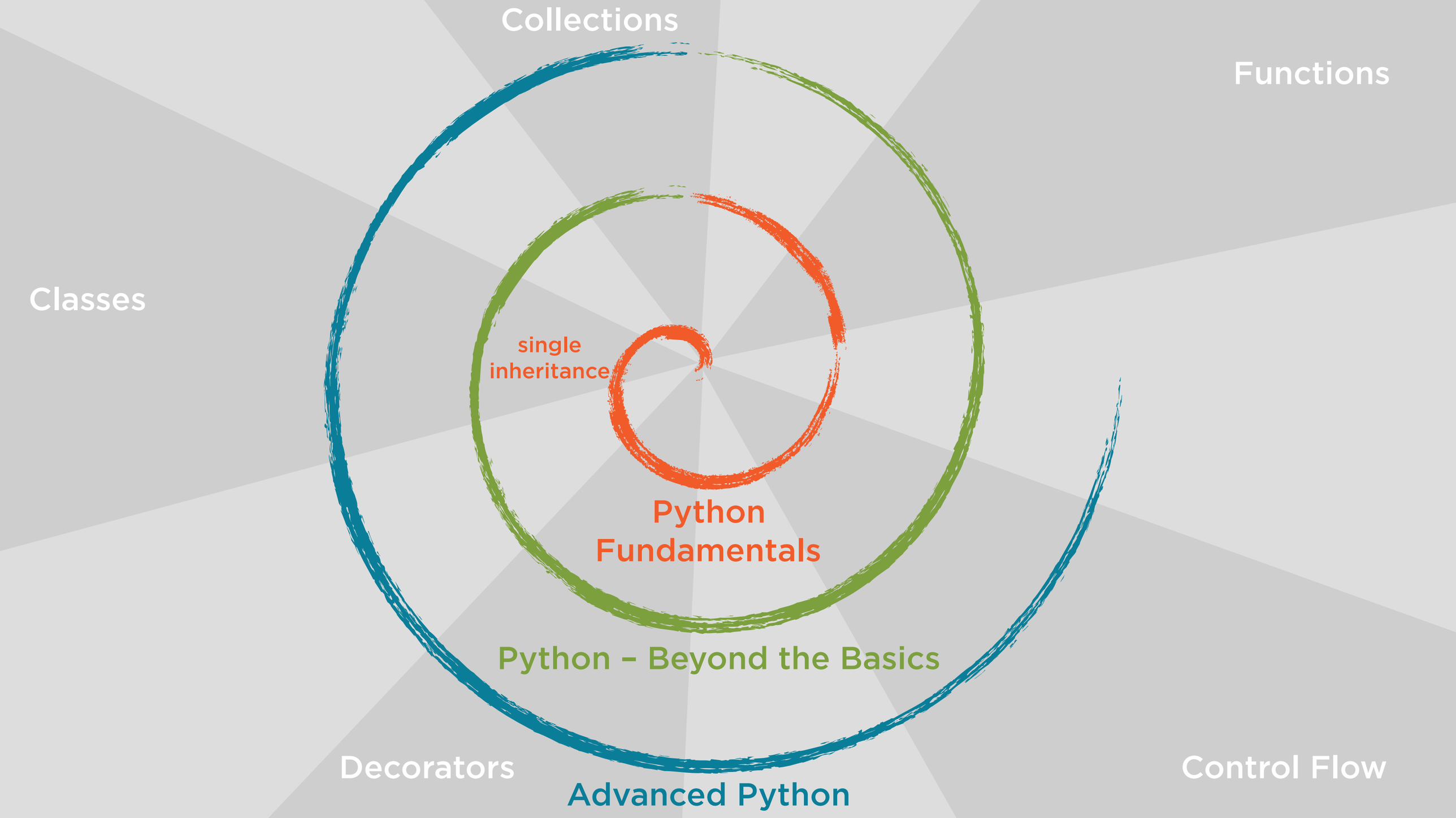
Python Fundamentals

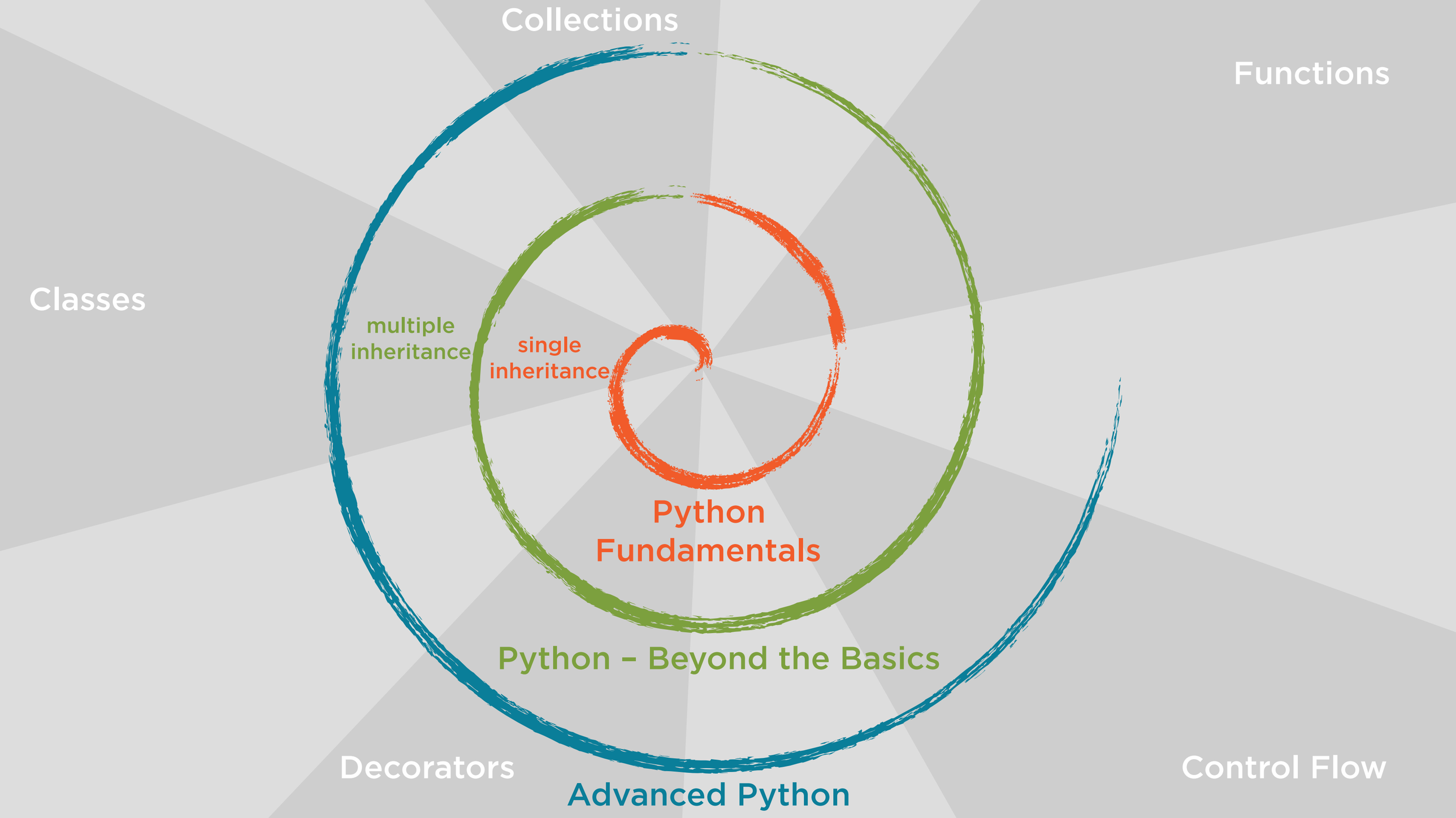
Collections
Functions
Classes
Decorators
Control Flow

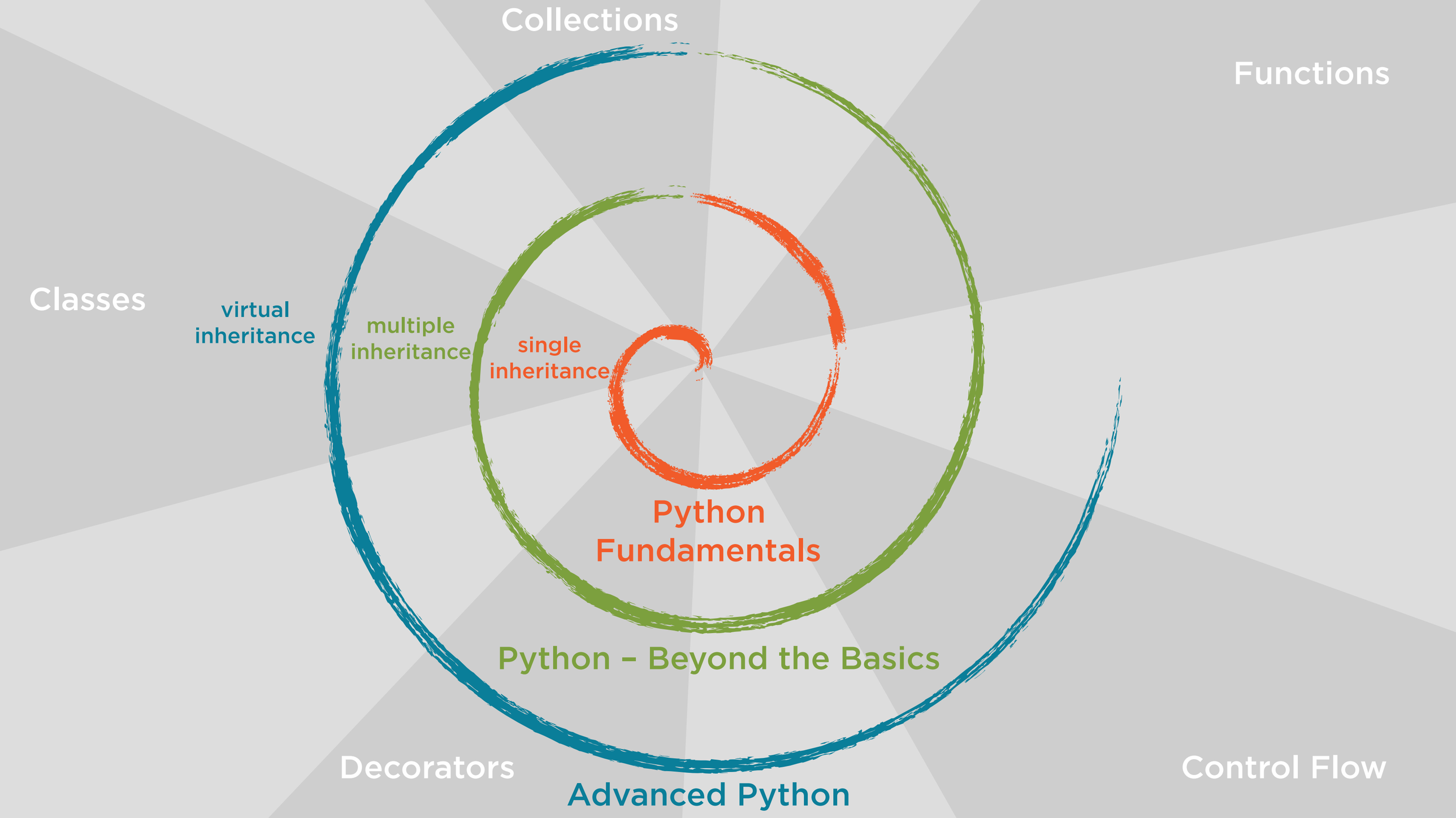

Collections
Functions
Classes
Python
Fundamentals
Decorators
Control Flow

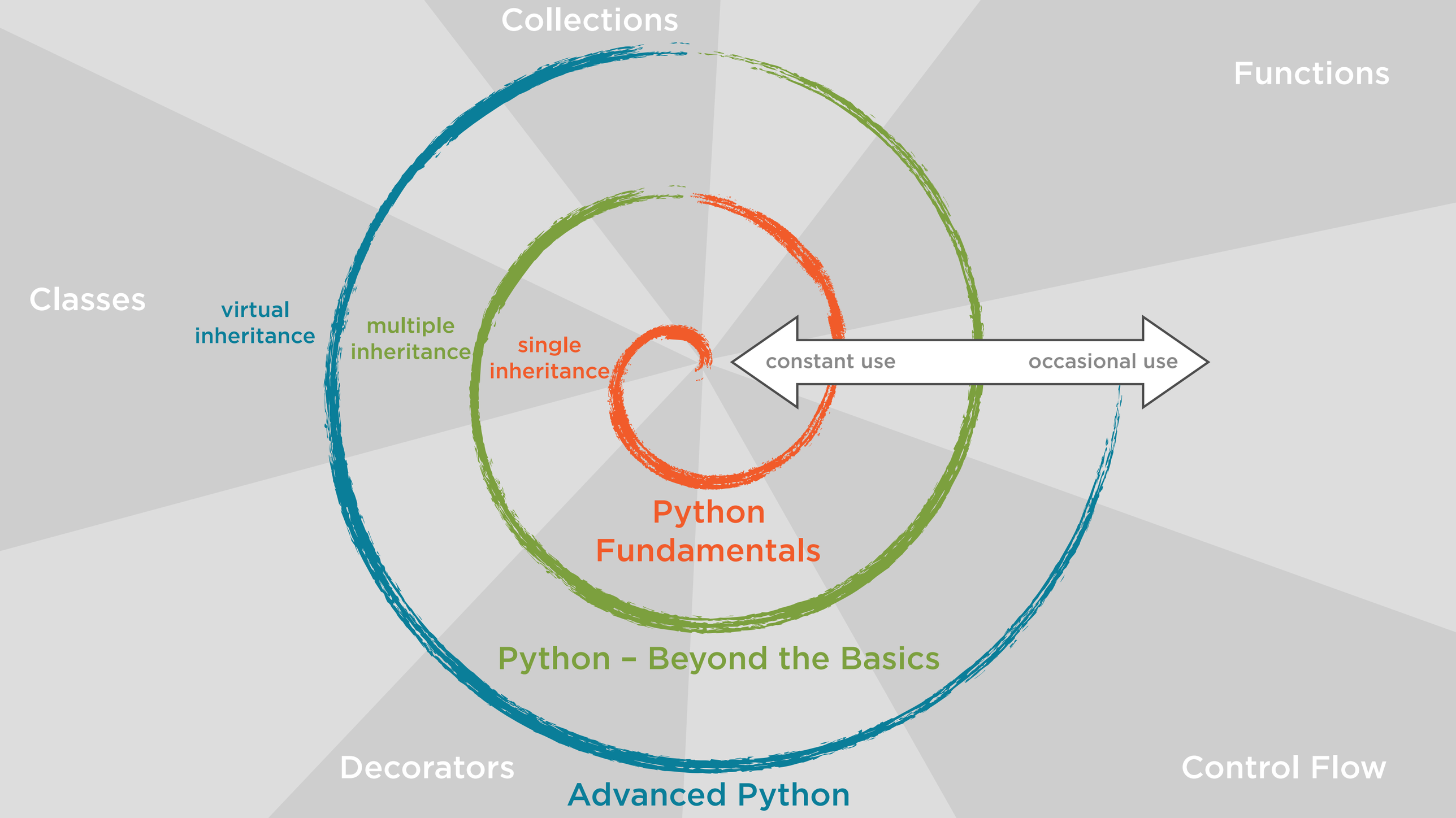

Collections
Functions
Classes
Python
Fundamentals
Python – Beyond the Basics
Decorators
Control Flow


Collections
Functions
Classes
Python
Fundamentals
Python – Beyond the Basics
Decorators
Control Flow
Advanced Python


Collections
Functions
Classes
single
inheritance
Python
Fundamentals
Python – Beyond the Basics
Decorators
Control Flow
Advanced Python


Collections
Functions
Classes
multiple inheritance
single inheritance
Python Fundamentals
Python – Beyond the Basics
Decorators
Control Flow
Advanced Python


Collections
Functions
Classes
virtual inheritance
multiple inheritance
single inheritance
Python Fundamentals
Python – Beyond the Basics
Decorators
Advanced Python
Control Flow


Collections
Functions
Classes
virtual inheritance
multiple inheritance
single inheritance
constant use
occasional use
Python Fundamentals
Python – Beyond the Basics
Decorators
Advanced Python
Control Flow


250MM
LUX-TOOLS

Summary

Summary

Summary

Advanced flow control

Summary

Advanced flow control

Byte-oriented programming

Summary

Advanced flow control

Byte-oriented programming

Object internals

Summary

Advanced flow control

Byte-oriented programming

Object internals

Descriptors

Summary

Advanced flow control

Byte-oriented programming

Object internals

Descriptors

Instance creation

Summary

Advanced flow control

Byte-oriented programming

Object internals

Descriptors

Instance creation

Metaclasses

Summary

Advanced flow control

Byte-oriented programming

Object internals

Descriptors

Instance creation

Metaclasses

Class decorators

Python Fundamentals

built-ins

Python Fundamentals

built-ins

functions

Python Fundamentals

built-ins

functions

Python Fundamentals

classes

built-ins

functions

Python Fundamentals

classes

modules

Python – Beyond the Basics

built-ins

functions

Python Fundamentals

classes

modules

decorators

Python – Beyond the Basics

built-ins

functions

Python Fundamentals

classes

modules

decorators

staticmethod

Python – Beyond the Basics

built-ins

functions

Python Fundamentals

classes

modules

decorators

staticmethod

Python – Beyond the Basics

classmethod

built-ins

functions

Python Fundamentals

classes

modules

decorators

staticmethod

Python – Beyond the Basics

classmethod

properties

built-ins

functions

Python Fundamentals

classes

modules

Advanced Python

decorators

staticmethod

Python – Beyond the Basics

classmethod

properties

built-ins

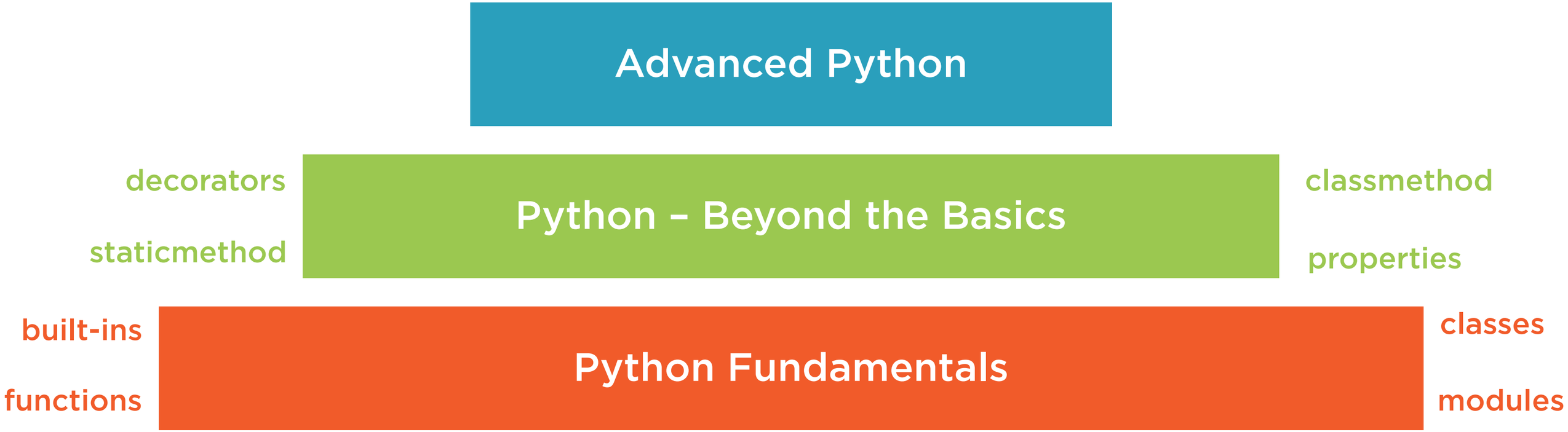
functions

Python Fundamentals

classes

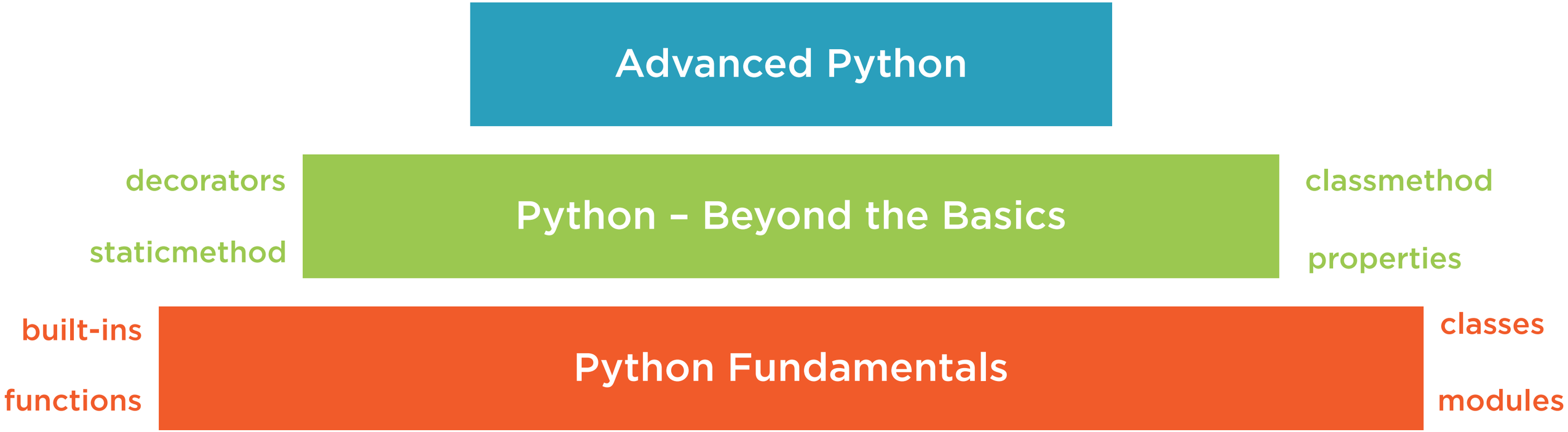
modules

```
$ python3
Python 3.6.0 (default, Jan  6 2017, 14:13:24)
[GCC 4.2.1 Compatible Apple LLVM 7.3.0 (clang-703.0.31)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```



Python 3.5 minimum

```
$ python3
Python 3.6.0 (default, Jan  6 2017, 14:13:24)
[GCC 4.2.1 Compatible Apple LLVM 7.3.0 (clang-703.0.31)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```



Python 3.5 minimum

```
$ python3
Python 3.6.0 (default, Jan  6 2017, 14:13:24)
[GCC 4.2.1 Compatible Apple LLVM 7.3.0 (clang-703.0.31)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Advanced Python

decorators

staticmethod

Python – Beyond the Basics

classmethod

properties

built-ins

functions

Python Fundamentals

classes

modules

Terminology for Python Special Methods

`__<method-name>__`

Terminology for Python Special Methods

`__<method-name>__`

"dunder"

Terminology for Python Special Methods

`__<method-name>__`

"dunder"

"double underscore"

Terminology for Python Special Methods

`__len__`

“dunder len”

Companion **Python Craftsman** Book Series

Pluralsight
Advanced Python

https://leanpub.com
/python-master
/c/pluralsight

Companion **Python Craftsman** Book Series

Pluralsight
Python Fundamentals

https://leanpub.com
/python-apprentice
/c/pluralsight

Pluralsight
Python – Beyond the Basics

https://leanpub.com
/python-journeyman
/c/pluralsight

Pluralsight
Advanced Python

https://leanpub.com
/python-master
/c/pluralsight

Advanced Flow Control

Summary

Summary

Summary

Advanced Flow Control

Summary

Advanced Flow Control

- `else` clauses on loops

Summary

Advanced Flow Control

- `else` clauses on loops
- `else` clauses on try blocks

Summary

Advanced Flow Control

- `else` clauses on loops
- `else` clauses on try blocks
- emulating switch statements

Summary

Advanced Flow Control

- `else` clauses on loops
- `else` clauses on try blocks
- emulating switch statements
- dispatching function calls on type

Summary

Advanced Flow Control

- `else` clauses on loops
- `else` clauses on try blocks
- emulating switch statements
- dispatching function calls on type

Unusual language features

Summary

Advanced Flow Control

- `else` clauses on loops
- `else` clauses on try blocks
- emulating switch statements
- dispatching function calls on type

Unusual language features

- Understand code you encounter

Loop-else Clauses

The `while` … `else` Construct

The `while` … `else` Construct

Guido van Rossum

Python-ideas mailing list
(2009)

The `while` … `else` Construct

Guido van Rossum

Python-ideas mailing list (2009)

Photo: Jason E. Kaplan

The `while` … `else` Cons

Guido van Rossum

Python-ideas mailing list (2009)

Photo: Jason E. Kaplan

Donald Knuth

Structured Programming with go to Statements. (1974) Computing Surveys. ACM.

Photo: Dasha Slobozhanina CC-BY

The `while` … `else` Construct

```
if condition:
    execute_condition_is_true()
else:
    execute_condition_is_false()
```

if … else

The else clause is executed if the condition is false

The `while` … `else` Construct

```
if condition:
    execute_condition_is_true()
else:
    execute_condition_is_false()
```

```
while condition:
    execute_condition_is_true()
else:
    execute_condition_is_false()
```

if … else

The else clause is executed if the condition is false

while … else

The else clause is executed if or when the condition is false

The `while` … `else` Construct

```
while condition:
    execute_condition_is_true()
else:
    execute_condition_is_false()
```

while … else

The else clause is executed if or when the condition is false

The `while` … `else` Construct

```
while condition:
    execute_condition_is_true()
else:
    execute_condition_is_false()
```

```
while condition:
    execute_condition_is_true()

execute_condition_is_false()
```

while … else

The else clause is executed if or when the condition is false

while …

Following statements are executed if or when the condition is false

The `while` … `else` Construct

```
while condition:
    execute_condition_is_true()
else:
    execute_condition_is_false()
```

while … else

The else clause is executed if or when the condition is false

The `while` … `else` Construct

```
while condition:
    execute_condition_is_true()
else:
    execute_condition_is_false()
```

```
while condition:
    flag = execute_condition_is_true()
    if flag:
        break
execute_condition_is_false()
```

while … else

The else clause is executed if or when the condition is false

while …

The else clause is executed if or when the condition is false

The `while` … `else` Construct

```
while condition:
    execute_condition_is_true()
else:
    execute_condition_is_false()
```

```
while condition:
    flag = execute_condition_is_true()
    if flag:
        break
execute_condition_is_false()
```

while … else

The else clause is executed if or when the condition is false

while …

The else clause is executed if or when the condition is false

The `while` … `else` Construct

```
while condition:
    execute_condition_is_true()
else:
    execute_condition_is_false()
```

```
while condition:
    flag = execute_condition_is_true()
    if flag:
        break
execute_condition_is_false()
```

while … else

The else clause is executed if or when the condition is false

while …

The else clause is executed if or when the condition is false

NEVER

when we break out of the loop, irrespective of the condition

The `while` ... `else` Construct

```
while condition:
    flag = execute_condition_is_true()
    if flag:
        break
execute_condition_is_false()
```

while ...

The else clause is executed if or when the condition is false

NEVER

when we break out of the loop, irrespective of the condition

The `while` … `else` Construct

```
while condition:
    flag = execute_condition_is_true()
    if flag:
        break
execute_condition_is_false()
```

```
while condition:
    flag = execute_condition_is_true()
    if flag:
        break

if not condition:
    execute_condition_is_false()
```

while …

The else clause is executed if or when the condition is false

NEVER

when we break out of the loop, irrespective of the condition

while… if…

An independent if clause with a negated condition can prevent execution when breaking from the loop

The `while` … `else` Construct

```
while condition:
    flag = execute_condition_is_true()
    if flag:
        break
execute_condition_is_false()
```

```
while condition:
    flag = execute_condition_is_true()
    if flag:
        break

if not condition:
    execute_condition_is_false()
```

while …

The else clause is executed if or when the condition is false

NEVER

when we break out of the loop, irrespective of the condition

while… if…

An independent if clause with a negated condition can prevent execution when breaking from the loop

The `while` ... `else` Construct

```
while condition:
    flag = execute_condition_is_true()
    if flag:
        break
execute_condition_is_false()
```

```
while condition:
    flag = execute_condition_is_true()
    if flag:
        break

if not condition:
    execute_condition_is_false()
```

Violates DRY
Don't Repeat Yourself

while ...

The else clause is executed if or when the condition is false

NEVER

when we break out of the loop, irrespective of the condition

while... if...

An independent if clause with a negated condition can prevent execution when breaking from the loop

The `while` ... `else` Construct

```
while condition:
    flag = execute_condition_is_true()
    if flag:
        break

if not condition:
    execute_condition_is_false()
```

Violates DRY
Don't Repeat Yourself

while... if...

An independent if clause with a negated condition can prevent execution when breaking from the loop

The `while` ... `else` Construct

```
while condition:
    flag = execute_condition_is_true()
    if flag:
        break

if not condition:
    execute_condition_is_false()
```

Violates DRY
Don't Repeat Yourself

```
while condition:
    flag = execute_condition_is_true()
    if flag:
        break
else:
    execute_condition_is_false()
```

while... if...

An independent if clause with a negated condition can prevent execution when breaking from the loop

while...else

Allows us to avoid the redundant test

The `while` ... `else` Construct

```
while condition:
    flag = execute_condition_is_true()
    if flag:
        break

if not condition:
    execute_condition_is_false()
```

Violates DRY
Don't Repeat Yourself

```
while condition:
    flag = execute_condition_is_true()
    if flag:
        break
else:
    execute_condition_is_false()
```

while... if...

An independent if clause with a negated condition can prevent execution when breaking from the loop

while...else

Allows us to avoid the redundant test

The `while` ... `else` Construct

```
while condition:
    flag = execute_condition_is_true()
    if flag:
        break

if not condition:
    execute_condition_is_false()
```

Violates DRY
Don't Repeat Yourself

```
while condition:
    flag = execute_condition_is_true()
    if flag:
        break
else:  # nobreak
    execute_condition_is_false()
```

while... if...

An independent if clause with a negated condition can prevent execution when breaking from the loop

while...else

Allows us to avoid the redundant test

Assist readers of your code with a helpful comment

Demo

Demo

Demo

Using `while…else` in practice

Demo

Using `while…else` in practice

Evaluating stack programs

Demo

Using `while…else` in practice

Evaluating stack programs

5 + 2

Demo

Using `while…else` in practice

Evaluating stack programs

5 + 2

Demo

Using `while…else` in practice

Evaluating stack programs

5 + 2

2

+

Demo

Using `while…else` in practice

Evaluating stack programs

5 + 2

Demo

Using `while…else` in practice

Evaluating stack programs

5 + 2

Demo

Using `while…else` in practice

Evaluating stack programs

5 + 2

(5 + 2) * 3

Demo

Using `while…else` in practice

Evaluating stack programs

5 + 2

(5 + 2) * 3

Demo

Using `while…else` in practice

Evaluating stack programs

5 + 2

(5 + 2) * 3

Demo

Using `while…else` in practice

Evaluating stack programs

5 + 2

(5 + 2) * 3

Demo

Using `while…else` in practice

Evaluating stack programs

5 + 2

(5 + 2) * 3

Demo

Using `while…else` in practice

Evaluating stack programs

5 + 2

(5 + 2) * 3

Demo

Using `while…else` in practice

Evaluating stack programs

5 + 2

(5 + 2) * 3

The `for` … `else` Construct

```
for item in iterable:
    if match(item):
        result = item
        break
else:
    # No match found
    result = None

# Always come here
print(result)
```

The `for` … `else` Construct

```
for item in iterable:
    if match(item):
        result = item
        break
else:  # nobreak
    # No match found
    result = None

# Always come here
print(result)
```

Remember – the else clause is really the no-break clause

Demo

Demo

Demo

Using `for…else` in practice.

Demo

Using `for…else` in practice.

Ensuring a list contains at least one integer divisible by a given value.

Demo

Using `for…else` in practice.

Ensuring a list contains at least one integer divisible by a given value.

Demo

Using `for…else` in practice.

Ensuring a list contains at least one integer divisible by a given value.

Are any of these numbers divisible by 12?

Demo

Using `for…else` in practice.

Ensuring a list contains at least one integer divisible by a given value.

Are any of these numbers divisible by 12?

Demo

Using `for…else` in practice.

Ensuring a list contains at least one integer divisible by a given value.

Are any of these numbers divisible by 12?

```
[2, 25, 9, 37, 28, 14]
```

Demo

Using `for…else` in practice.

Ensuring a list contains at least one integer divisible by a given value.

Are any of these numbers divisible by 12?

```
[2, 25, 9, 37, 28, 14]
```

Demo

Using `for…else` in practice.

Ensuring a list contains at least one integer divisible by a given value.

Are any of these numbers divisible by 12?

```
[2, 25, 9, 37, 28, 14]
```

No, so append a number that is.

Demo

Using `for…else` in practice.

Ensuring a list contains at least one integer divisible by a given value.

Are any of these numbers divisible by 12?

```
[2, 25, 9, 37, 28, 14]
```

No, so append a number that is.

loop-else clauses

loop-else clauses

Uncommon

Although for-else more common than while-else

loop-else clauses

Uncommon

Although for-else more common than while-else

Consider readership

Are you sure code maintainers understand?

loop-else clauses

Uncommon

Although for-else more common than while-else

Consider readership

Are you sure code maintainers understand?

Even experts are confused

A majority interviewed at PyCon 2011 could not properly understand loop-else clauses

If loop-else clauses are so bad, what's the alternative?

If loop-else clauses are so bad, what's the alternative?

Refactor! Extract method.

Try-else Clauses

The `try…else` Construct

```
try:
    # This code might raise an exception
    do_something()
except ValueError:
    # ValueError caught and handled
    handle_value_error()
else:
    # No exception was raised
    # We know that do_something() succeeded, so
    do_something_else()
```

The `try…else` Construct

```python
try:
    # This code might raise an exception
    do_something()
except ValueError:
    # ValueError caught and handled
    handle_value_error()
else:
    # No exception was raised
    # We know that do_something() succeeded, so
    do_something_else()
```

```python
try:
    # This code might raise an exception
    do_something()
    do_something_else()
except ValueError:
    # ValueError caught and handled
    handle_value_error()
```

The `try…else` Construct

```python
try:
    # This code might raise an exception
    do_something()
except ValueError:
    # ValueError caught and handled
    handle_value_error()
else:
    # No exception was raised
    # We know that do_something() succeeded, so
    do_something_else()
```

```python
try:
    # This code might raise an exception
    do_something()
    do_something_else()
except ValueError:
    # ValueError caught and handled
    handle_value_error()
```

The `try…else` Construct

```
try:
    # This code might raise an exception
    do_something()
except ValueError:
    # ValueError caught and handled
    handle_value_error()
else:
    # No exception was raised
    # We know that do_something() succeeded, so
    do_something_else()
```

```
try:
    # This code might raise an exception
    do_something()
    do_something_else()
except ValueError:
    # ValueError caught and handled
    handle_value_error()
```

Ambiguous!

Which call raised the ValueError?

The `try…else` Construct

```
try:
    f = open(filename, 'r')
except OSError:  # OSError replaces IOError from Python 3.3 onwards
    print("File could not be opened for read")
else:
    # Now we're sure the file is open
    print("Number of lines", sum(1 for line in f))
    f.close()
```

The `try…else` Construct

```
try:
    f = open(filename, 'r')
except OSError:  # OSError replaces IOError from Python 3.3 onwards
    print("File could not be opened for read")
else:
    # Now we're sure the file is open
    print("Number of lines", sum(1 for line in f))
    f.close()
```

Focussed

Try-block specific to the open operation

Emulating Switch

The **switch** Statement in C

```
switch (menu_option) {
    case 1:  single_player();    break;
    case 2:  multi_player();     break;
    case 3:  load_game();        break;
    case 4:  save_game();        break;
    case 5:  reset_high_score(); break;
    default:
        printf("No such option!");
        break;
}
```

The `switch` Statement in C

```
switch (menu_option) {
    case 1:  single_player();    break;
    case 2:  multi_player();     break;
    case 3:  load_game();        break;
    case 4:  save_game();        break;
    case 5:  reset_high_score(); break;
    default:
        printf("No such option!");
        break;
}
```

Multi-way branching with optional default

Demo

Demo

Demo

There is no switch construct in Python

Demo

There is no switch construct in Python

Option #1 – `if … elif … elif … else`

Demo

There is no switch construct in Python

Option #1 – `if … elif … elif … else`

Option #2 – Mapping of callables

Demo

There is no switch construct in Python

Option #1 – `if … elif … elif … else`

Option #2 – Mapping of callables

Refactor an adventure game from Option #1 to Option #2

Dispatching on Type

Demo

Demo

Demo

To dispatch on type

Demo

To dispatch on type

- Function selected based on type of arguments

Demo

To dispatch on type

- Function selected based on type of arguments
- Methods: called implementation depends on type of `self`

Demo

To dispatch on type

- Function selected based on type of arguments
- Methods: called implementation depends on type of `self`
- Regular functions: switch-emulation is ungainly

Summary

Summary

Summary

while...else

Summary

while...else

- else-clause executed when condition becomes False

Summary

while...else

- else-clause executed when condition becomes False
- else-clause is the no-break clause

Summary

while...else

- else-clause executed when condition becomes False
- else-clause is the no-break clause
- only useful when break is present

Summary

Summary

Summary

for...else

Summary

for...else

- else-clause is no-break clause

Summary

for...else

- else-clause is no-break clause
- most useful in searching algorithms

Summary

for...else

- else-clause is no-break clause
- most useful in searching algorithms
- else-clause useful as not-found clause

Summary

for...else

- else-clause is no-break clause
- most useful in searching algorithms
- else-clause useful as not-found clause
- most developers don't understand loop-else clauses!

Summary

Summary

Summary

try...except...else

Summary

try...except...else

- else-clause is success clause

Summary

try...except...else

- else-clause is success clause
- use to narrow scope of try block

Summary

Summary

Summary

switch-emulation

Summary

switch-emulation

- no switch construct in Python

Summary

switch-emulation

- no switch construct in Python

- `if … elif … elif … else`
  error prone

Summary

switch-emulation

- no switch construct in Python
- `if … elif … elif … else`
  error prone
- use mapping of callables

Summary

Summary

Summary

dispatch on type

Summary

dispatch on type

- `@singledispatch` decorator

Summary

dispatch on type

- `@singledispatch` decorator
- generic functions

Summary

dispatch on type

- `@singledispatch` decorator
- generic functions
- module scope functions only