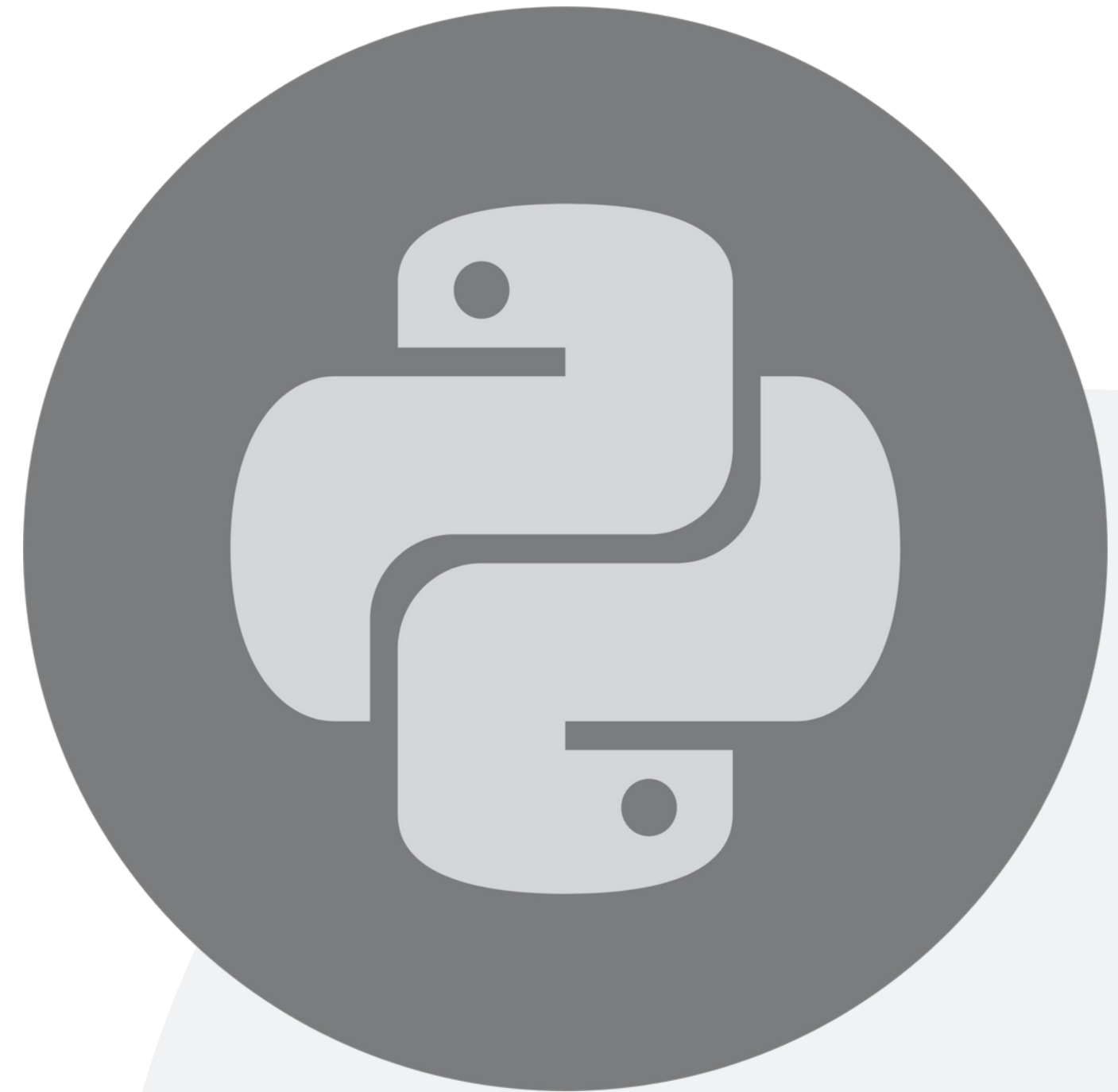


PYTHON COURSE

ENTRY LEVEL

Basics of programming in Python 3.10

This course will cover part of the arguments found in
PCEP™ – Certified Entry-Level Python Programmer
Certification



INTRO TO FUNCTIONS AND METHODS



WHAT METHODS ARE

- Methods are, in synthesis, objects' typical and referred functions
- Different objects has different methods
- use `object_name.method_name(arguments)` to access and use object's methods

method
calling

use . to access object methods

`a = "HELLO" # variable a is a string object (class: str)`

`a = a.lower() # returns the same string but lowercase`

`print(a) # hello`

object name

method
name





STRING METHODS

lower (), upper() and title() methods

change strings case with those methods

```
a = "HeLl0 mate" # variable a is a string object (class: str)
b = a.lower() # returns the same string but lowercase
c = b.upper() # returns the same string but UPPERCASE
d = a.title() # returns the same string but Title Case
```





STRING METHODS

strip(), rstrip() and lstrip() methods

lstrip() removes spaces from beginning of string

```
a = "    string1" # variable a is a string object (class: str)

len(a) # 11
b = a.lstrip() # removes all the spaces at beginning of string

print(b) # string1
print(len(b)) # 7
```



STRING METHODS

rstrip() removes spaces from beginning of string

```
a = "string1   " # variable a is a string object (class: str)

len(a) # 11
b = a.rstrip() # removes all the spaces at end of string

print(b) # string1
print(len(b)) # 7
```

STRING METHODS

strip() removes spaces from both left and right of the string

```
a = "    string1    " # variable a is a string object (class: str)

len(a) # 15
b = a.strip() # removes all the spaces at start and end of string

print(b) # string1
print(len(b)) # 7
```

STRING METHODS

removeprefix() and removesuffix()

used to remove a specific substring passed as argument to the method from the beginning or end of the main string

```
url = "https://python.org/" # variable a is a string object (class: str)

# removes the substring from the main string
cleaned_string = url.removeprefix("https://")

# removes the substring from the end of the string
cleaned_string = cleaned_string.removesuffix(".org/")

print(cleaned_string) # prints python
```


STRING METHODS

replace()

replace a specific substring passed as first argument in the main string with a second value passed as second argument

```
url = "//pyt/hon.org/" # variable a is a string object (class: str)

# removes the substring from the main string
cleaned_link = url.replace("/", "")

print(cleaned_link) # prints python.org
```

BUILD FORMATTED STRINGS

in python you can build formatted strings passing dynamical parts without explicit concatenation process

using .format() method

with key-value arguments:

```
a = "Matt"
b = "Groening"
message = "Hi {name} {surname}, Welcome Back".format(name = a, surname = b)
print(message)
```

**use curly brackets to
define variable parts**

**call the format method,
pass as parameter names the variable
string parts' names**

**value passed will be
concatenated in the string at
desired position**



BUILD FORMATTED STRINGS

with positional arguments:

```
a = "Matt"
b = "Groening"
message = "Hi {} {}, Welcome Back".format(a, b)
print(message)
```

Diagram illustrating the code execution flow:

- The variables `a` and `b` are assigned values.
- The `message` string is created using the `.format(a, b)` method, where `a` and `b` are passed as positional arguments to replace the curly braces.
- The `print` function is called with `message` as an argument.

use empty curly
brackets

call the `.format` method and pass
variables or values as positional
arguments





BUILD FORMATTED STRINGS

the easiest way to build formatted string is to use an f before the string:

```
a = "Matt"
b = "Groening"
message = f"Hi {a} {b}, Welcome Back"
print(message)
```

an f before the
string is required



QUESTION TIME

QUESTION 1:

What is the output of the following python program:

```
x = int(input("insert a number: ")) # user insert 2.0  
  
b = x ** 2  
  
message = f"result is {b}"
```

- 1.ValueError at line 1
- 2.TypeError at line 3
- 3.TypeError at line 5
- 4.SyntaxError at line 1

QUESTION 2:

How many lines will the output of this code be?

```
var = "a"  
c = "b"  
  
print(a, end="")  
print(c)
```

1. 0 lines
2. 2 lines
3. 1 line
4. None of the above



QUESTION 3:

Select correct statements:

1. A numeric value can be concatenated to a string without casting
2. Casting is the operation used to change type to a value
3. Casting integer into floats can lead to data losses
4. Casting floats into integer can lead to data losses


QUESTION 4:

Select correct statements:

- 
1. `int()` function is used to cast objects into strings
 2. `float()` function is used to cast objects into floats
 3. `str()`, `int()` and `float()` are methods
 4. `str()`, `int()` and `float()` are functions
- 

QUESTION 5:

Select correct statements:

- 
1. passing both key-value and positional arguments to a function is consented.
 2. key-value arguments must follow positional arguments.
 3. positional arguments can only be strings.
 4. if every key-value argument is passed after positional ones, program will raise Error.
- 