Python (Lec 2)

Saturday, 2 October 2021 2:24 PM

print()

.py file extension

* python is interpreted, not assembled

foo = bar var declaration, no difference between dec & assign

↳ types are handled automatically

- int
- float
- str
- bool
- NoneType

* None = null

input() = prompt()

'+' operator can concat like js

ex. "Hello, " + var

fstrings also supported

ex. f"Hello, {var}"

conditionals:

```
if foo > bar:
    statement
elif bar > foo:
    statement
else:
```

* indentation integral!

* input() returns str, handle var types manually

↳ casting: int(), str(), etc.

sequences:

str[n] works.

* Lists are basically js arrays (use .append)

↳ all elements stored as Objects

Tuples are used for things like coords

↳ use (1, 2)

Sets are for unique values set() & set.add

dicts are for key-val pairs {key:val, key:val} >> js object syntax-ish:

↳ dict[key]

for commenting

len(sequence) = length

loops:

for i in _____:

↓

list, range, str, etc

functions:

```
def func(arg):
    return
```

importing:

```
from file import func
```

OR

```
import file ↳ file.func
```

OOP:

```
class ClassName():
    def __init__(self, arg1, arg2):
        self.arg2 = arg1
        self.arg2 = arg2
    def other_func(self, etc.):

var = ClassName(foo, bar)
```

STUFF WE NEED:

decorators: functions can be passed to others as inputs

```
ex. def announce(f):
        def wrapper():
            print("I'm running f")
            f()
        return wrapper
    @announce
    def func()
```

lambda f: "foobar"

↳ inline func declaration

exception handling:

```
import sys
try:
    foo
except error:
    bar
    sys.exit(1)
```