Core Python: Robust Resource and Error Handling

REVIEW

Austin Bingham
COFOUNDER - SIXTY NORTH

@austin_bingham

Robert Smallshire
COFOUNDER - SIXTY NORTH

@robsmallshire

Review

guess.py

```
from random import randrange


def main():
    number = randrange(100)
    while True:
        guess = int(input("? "))
        if guess == number:
            print("You win!")
            break


if __name__ == '__main__':
    main()

```

Exceptions Review

```
$ python guess.py
? 10
? 37
? 53
? 22
? Traceback (most recent call last):
  File "guess.py", line 14, in <module>
    main()
  File "guess.py", line 7, in main
    guess = int(input("? "))
KeyboardInterrupt

$ python guess.py
? 45
? 21
? 99
? seven
Traceback (most recent call last):
  File "guess.py", line 14, in <module>
    main()
  File "guess.py", line 7, in main
    guess = int(input("? "))
ValueError: invalid literal for int() with base 10: 'seven'
$
```

guess.py

```
1  from random import randrange
2
3
4  def main():
5      number = randrange(100)
6      while True:
7          try:
8              guess = int(input("? "))
9          except:
10             continue
11         if guess == number:
12             print("You win!")
13             break
14
15
16 if __name__ == '__main__':
17     main()
18
```

Exceptions Review

```
$ python guess.py
? 10
? 17
? seven
? 9
? ? ? ? ? ?
```

Catching everything also catches `KeyboardInterrupt`!

It's generally wrong to catch all exceptions.

guess.py

```
1  from random import randrange
2
3
4  def main():
5      number = randrange(100)
6      while True:
7          try:
8              guess = foo(input("? "))
9          except:
10             continue
11         if guess == number:
12             print("You win!")
13             break
14
15
16 if __name__ == '__main__':
17     main()
18
```


Unresolved reference 'foo'
8:24 UTF-8 4 spaces Python 3.8 (code_editor)

Exceptions Review

```
$ python guess.py
```

guess.py

```
1  from random import randrange
2
3
4  def main():
5      number = randrange(100)
6      while True:
7          try:
8              guess = foo(input("? "))
9          except ValueError:
10             continue
11         if guess == number:
12             print("You win!")
13             break
14
15
16 if __name__ == '__main__':
17     main()
18
```

Exceptions Review

```
$ python guess.py
Traceback (most recent call last):
  File "guess.py", line 17, in <module>
    main()
  File "guess.py", line 8, in main
    guess = foo(input("? "))
NameError: name 'foo' is not defined
$
```

words

guess.py

```
1  from random import randrange
2
3
4  def main():
5      number = randrange(100)
6      while True:
7          try:
8              guess = int(input("? "))
9          except ValueError:
10             continue
11         if guess == number:
12             print("You win!")
13             break
14
15
16 if __name__ == '__main__':
17     main()
18
```

Exceptions Review

```
$ python guess.py
? 10
? 15
? seven
? 3
? Traceback (most recent call last):
  File "guess.py", line 17, in <module>
    main()
  File "guess.py", line 8, in main
    guess = int(input("? "))
KeyboardInterrupt

$
```

You should almost always specify an exception type in an except statement.

Summary

Errors are almost always communicated using exceptions.

Overly-broad exception handlers can catch exceptions you don't want to catch.

- `KeyboardInterrupt`
- `NameError`

Generally specify an exception type in your except statements.