Lambda Expressions

Lambda Expressions vs DEF

- Lambda Expressions have the form:

lambda arg1, arg2, ...argN : expression using arguments

Ex : lambda x, y: x + y

Lambda Expression

Lambda Expressions

- ***They return a function or expression***
- ***That you can evaluate***

```
f = lambda x, y: x + y
print(type(f))
# Evaluate the function
print(f(1,2))
```

```
<class 'function'>
3
```

Compare Lambda to Def

Here ***def*** and ***lambda*** do the same thing

```
def f_def(x, y):
    return x + y

print("def function call " ,f_def(1,2))

f_lambda = lambda x, y : x + y

print("lambda expression ",f_lambda(1,2))
```

```
def function call  3
lambda expression  3
```

Compare Def and Lambda

- Def can contain multiple statements in a block of code
- Lambda is a one-liner

```
def f_def(x, y):
    x = x + 1
    y = y + 2
    return x + y

print("def function call " ,f_def(1,2))

f_lambda = lambda x, y : x + y

print("lambda expression ",f_lambda(1,2))
```

```
def function call  6
lambda expression  3
```

When you cannot use lambda

You cannot check exceptions with lambda expressions. Example divide by zero

```
divide = lambda x, y : x / y
print(divide(8,2))
print(divide(8,0))
```

```
4.0

---------------------------
ZeroDivisionError
<ipython-input-20-83a0f0df7c43
```

Use a Def if exceptions are possible

```
def divide(x, y):
    try: return x/y
    except ZeroDivisionError: return 0


print(divide(8,0))
print(divide(8,2))
```

```
0
4.0
```

What are tuples?

- How to declare them and how to access their values

```
tuple = (2,3)
print(type(tuple))
print(tuple[0])
print(tuple[1])
```

```
<class 'tuple'>
2
3
```

Tuples and Lambda Expressions

```
#Notice x is a tuple - here we pa
f2 = lambda x: x[0] + x[1]
tuple = (2, 3)
print (f2( tuple ))
```

```
5
```

Lambda Expressions and String Type

- String Replacement

```
my_replace = lambda str, old, new: str.replace( old, new )

str = "this is string example....wow!!! this is really string"
print(my_replace(str, "is", "was"))
```

```
thwas was string example....wow!!! thwas was really string
```

Another Example

- Used to parse text files

```
str = "$abc$efg.hijk.lmn"
print(my_replace(str, "$", ""))
```

```
abcefg.hijk.lmn
```