LAMBDA EXPRESSIONS

- **Lambda** is called an anonymous function, meaning it does not require a **def** name.
- Does not require a **return**.
- Lambdas are single line expressions.
- Can only be used as a substitute for basic functions.
- Cannot add a docstring.

In [29]:

```
def add(arg1, arg2):
    return arg1 + arg2

add(10, 20)
```

Out[29]:

```
30
```

In [39]:

```
g = lambda  x, y , z: x + y + z
```

In [40]:

```
type(g)
g(10,20, 100)
```

Out[40]:

```
130
```

In [ ]:

```
g()
```

In [66]:

```
g2 = lambda num1, num2 : num1*2 + num2
```

In [67]:

```
g2(10, 5)
```

Out[67]:

```
25
```

In [105]:

```
def check(num):
    return num % 2 == 0 or num > 5

check(9)
```

```
c = lambda num : num %2 == 0 or num > 5

check(10)
c(9)
```

Out[105]:

```
True
```

In [154]:

```
def chop(num):
    if num>= 10 and  num < 30:

        return num

chop(50)

c1 = lambda num2 : num2 > 10 and num2 < 30

c1(20)
```

Out[154]:

```
True
```

In [119]:

```
list(range(30))[10:20]
```

Out[119]:

```
[10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

In [84]:

```
import random as rd
w = lambda s, num= rd.randint(0,30): "inside" if num
in range(rd.randint(0,s)) else "outside"
```

In [96]:

```
w(9)
```

Out[96]:

```
'inside'
```

In [95]:

```
def compare(a, b):
    if a > 10:
        return a
    else:
        return b

compare(11, 2)
```

Out[95]:

```
11
```

In [45]:

```
a = 20; b = 5
con = lambda: a if a > 10 else b
```

In [47]:

```
a = 3
con()
```

Out[47]:

```
5
```

In [65]:

```
def size(x):
    if x > 100:
        return "big"
    else:
        return "small"
size(800)
```

Out[65]:

```
'big'
```

In [62]:

```
big = lambda x: "big" if x > 100 else "small"
```

In [63]:

```
big(9)
```

Out[63]:

```
'small'
```

In [4]:

```
f = lambda x:x *x
f(10)
```

Out[4]:

```
100
```

In [5]:

```
[f(x) for x in range(10)]
```

Out[5]:

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

In [6]:

```
[x**2 for x in range(10)]
```

Out[6]:

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

In [11]:

```
f = lambda x: x**2 if x in range(10) else "outside"
```

In [10]:

```
f(20)
```

In [12]:

```
def inside(num):

    if num in list(range(10)):
        return num**2
    else:
        print("outside")
```

In [15]:

```
inside(3)
```

Out[15]:

9