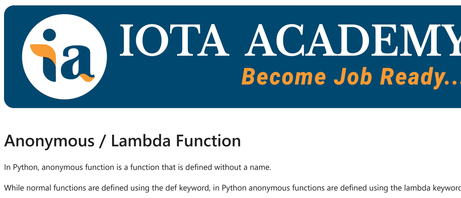

Anonymous / Lambda Function

In Python, anonymous function is a function that is defined without a name.

While normal functions are defined using the def keyword, in Python anonymous functions are defined using the lambda keyword.

Lambda functions are used extensively along with built-in functions like filter(), map()

We use the `lambda` keyword instead of def to create a lambda function.

without an argument

In [18]:
```
da = lambda : print('IOTA Academy')
# we have created a lambda function that prints 'Hello World'.
# assigned it to the da variable.
```

In [19]:
```
# call the lambda function
da()

# When we call the lambda function, the print() statement inside the lambda function is executed.
```

```
IOTA Academy
```

Python lambda Function with an Argument

- Similar to normal functions, the lambda function can also accept arguments. For example,

syntax:

```
lambda arguments: expression
```

Example:

In [1]:
```
# Lambda function with argument

double = lambda x: x*2  # x is an argument

print(double(5)) # here, double is variable which refers to the above lambda function
```

```
10
```

In [2]:
```
def double(x):
    return x * 2

print(double(5))  # here, double is a user-defined function.
```

```
10
```

In [3]:
```
# create a function
def seconditem(x):
    return x[1]
```

In [4]:
```
# sort the tuple based on the 2nd item

tuple1 = ((1,'hemant'),(2,'iota'),(3,'academy'))

sorted(tuple1,key=seconditem)
```

Out[4]:
```
[(3, 'academy'), (1, 'hemant'), (2, 'iota')]
```

In [5]:
```
# above program with lambda
sorted(tuple1, key=lambda x: x[1])
```

Out[5]:
```
[(3, 'academy'), (1, 'hemant'), (2, 'iota')]
```

In [14]:
```
# Example use with filter(function, iterable)

lst = [1, 2, 3, 4, 5]
even_lst = list(filter(lambda x: (x%2 == 0), lst)) # filter even numbers from a list
print(even_lst)
```

```
[2, 4]
```

In [8]:
```
# filter
lst = [33, 3, 22, 2, 11, 1]
filter_list = list(filter(lambda x: x > 10, lst)) # filter the numbers that are greater than 10
print(filter_list)
```

```
[33, 22, 11]
```

In [10]:
```
# Example use with map(function, iterable)

l = [11,22,33,4,5]
new = list(map(lambda x:x**2,l))  # square all the elements of a list
print(new)
```

```
[121, 484, 1089, 16, 25]
```

In [11]:
```
# Example use with reduce(function, iterable)
from functools import reduce

lst = [1, 2, 3, 4, 5]
product_lst = reduce(lambda x, y: x*y, lst)
print(product_lst)
```

```
120
```

In [12]:
```
# Example 1

x = lambda a : a + 10
print(x(5))
```

```
15
```

In [13]:
```
x(10)
```

Out[13]:
```
20
```

In [14]:
```
# Example 2:
# addition of three numbers
add_of_three = lambda a, b, c : a + b + c
print(add_of_three(5, 6, 2))
```

```
13
```

In [15]:
```
# Example 3:

a=lambda x: x.replace(",","")
```

In [16]:
```
a("1122,333")  # replace "," with ""
```

Out[16]:
```
'1122333'
```

It's also possible to use a lambda function to perform conditional operations. Below is a lambda analog for a simple if-else function:

In [17]:
```
fun = (lambda x: x if(x > 10) else 10)
```

In [18]:
```
fun(23)
```

Out[18]:
```
23
```

In [19]:
```
fun(5)
```

Out[19]:
```
10
```

In [25]:
```
# Example
# user define function
def check_conditions(x):
    if x > 10:
        return x * 10
    elif x < 5:
        return x * 5
    else:
        return x

check_conditions(11)
```

Out[25]:
```
110
```

In [26]:
```
l = lambda x: x * 10 if x > 10 else (x * 5 if x < 5 else x)
l(11)
```

Out[26]:
```
110
```

Exercise:

1) Write a program (WAP) to create a lambda function that adds 15 to a given number passed in as an argument,

2) Write a program to create a lambda function that multiplies argument x with argument y and print the result.

3) Write a program to sort the list by second element using lambda function.

> **input:**
>
> lst = [("English",88), ("Science",90), ("Maths",97), ("Social sciences",82)]
>
> **Expected Output:**
>
> [('Social sciences', 82), ('English', 88), ('Science', 90), ('Maths', 97)]

4) Write a Python program to filter even integers from a list using lambda function.

> **input:**
>
> num_list = [123, 23, 43, 45, 33, 22, 45, 67, 34, 20, 28]
>
> **Expected Output:**
>
> [22, 34, 20, 28]

5) Write a Python program to remove None value from a given list using lambda function.

> **Input:**
>
> lst = [12, 0, None, 23, None, -55, 234, 89, None, 0, 6, -12]
>
> **Expected Output:**
>
> [12, 0, 23, -55, 234, 89, 0, 6, -12]

6) Write a Python program to count the occurrences of the items in a given list using lambda function.

> **input:**
>
> nums = [3, 4, 5, 8, 0, 3, 8, 5, 0, 3, 1, 5, 2, 3, 4, 2]
>
> **Expected Output:**
>
> {3: 4, 4: 2, 5: 3, 8: 2, 0: 2, 1: 1, 2: 2}

7) Write a program to change data type of numbers inside the tuple to integers using lambda function.

> **Input:**
>
> tup = (('233','ABCD','33'),('1416','EFGH','55'),('2345','WERT','34'))
>
> **Expected Output:**
>
> ((233, 33), (1416, 55), (2345, 34))

8) Write a Python program to add two given lists using map and lambda function.

> **Input:**
>
> a = [1, 2, 3]
>
> b = [4, 5, 6]
>
> **Expected Output:**
>
> [5, 7, 9]

9) Write a Python program to find numbers divisible by 19 or 13 from a list of numbers using Lambda.

> **Input:**
>
> lst = [19, 65, 57, 39, 152, 639, 121, 44, 90, 190]
>
> **Expected Output:**
>
> [19, 65, 57, 39, 152, 190]

10) WAP a Python program to find the sublist with maximum length using lambda function.

> **Input:**
>
> a = [[0], [1, 3], [5, 7], [9, 11], [13, 15, 17],[8]]
>
> **Expected Output:**
>
> [13, 15, 17]

Great Job!