16.Functions

May 7, 2020

0.1 Prime

number which only divisible by one and itself

number which does not divisble by any number between 2-num-1

[29]:
```
%%timeit
num = 127
for check in range(2, num):
    #print(f"{num} % {check} = {num%check}")
    if num % check == 0:
        #print("Not prime")
        break
else:
    #print("Prime")
    pass
```

7.27 µs ± 660 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)

[31]:
```
%%timeit
num = 127
for check in range(2, num//2):
    #print(f"{num} % {check} = {num%check}")
    if num % check == 0:
        #print("Not prime")
        break
else:
    #print("Prime")
    pass
```

3.49 µs ± 440 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)

prime theory --> just start with 2 to sqrt or num

[36]:
```
from math import ceil
```

[35]:
```
from math import sqrt
int(sqrt(127))
```

[35]: 11

[37]:
```
ceil(sqrt(127))
```

[37]: 12

[ ]:

[48]:
```
%%timeit
num = 127
limit = ceil(sqrt(num)) + 1
for check in range(2, limit):
    if num % check == 0:
        #print("Not prime")
        break
else:
    #print("Prime")
    pass
```

```
1.96 µs ± 103 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)
```

[61]:
```
def prime(number):
    if number <= 1:
        return False
    elif number <= 3:
        return True
    else:
        limit = ceil(sqrt(number)) # sqrt(4) --> 2 --> 2
        for check in range(2, limit+1):
            if number % check == 0:
                return False
        else:
            return True
```

[62]:
```
prime(1217)
```

[62]: True

[68]:
```
def prime_range(start, end):
    p = [ ]
    for number in range(start, end+1):
        if prime(number):
            p.append(number)
    print(f"Total Prime in range {start}-{end} are {len(p)}")
    return p
```

[69]:
```
start = 10000
end = 99999
ans = prime_range(start, end)
```

```
Total Prime in range 10000-99999 are 8363
```

[70]:
```
print(ans[:5])
```

```
[10007, 10009, 10037, 10039, 10061]
```

[71]:
```
print(ans[-5:])
```

```
[99929, 99961, 99971, 99989, 99991]
```

Type of arguments in Functions

* Positional Arguments
* Default Arguments
* Variable Length Positional Arguments
* Variable Leangth Key-word Arguments

[73]:
```
def func(x, y): # positional argument  [ formal arguments]
    #    0, 1 # are compulsory arguments in python
    """
        x, y are keywords and are positional arguments

        local variables
    """
    print(f"x = {x} \n y = {y}")
```

[74]:
```
func(10, 20)  # positional arguments [ actual arguments ]
#    0, 1
```

```
x = 10
 y = 20
```

[75]:
```
func()
```

```
␣
↪---------------------------------------------------------------------------

       TypeError                                 Traceback (most recent call␣
↪last)

       <ipython-input-75-bd1982955a12> in <module>
   ----> 1 func()
```

```
TypeError: func() missing 2 required positional arguments: 'x' and 'y'
```

[76]:
```
func(10)
```

```
␣
↪---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call␣
↪last)

<ipython-input-76-0baf65f248dd> in <module>
----> 1 func(10)

TypeError: func() missing 1 required positional argument: 'y'
```

[77]:
```
func(10, 20, 30)
```

```
␣
↪---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call␣
↪last)

<ipython-input-77-2ef4a7b1b38b> in <module>
----> 1 func(10, 20, 30)

TypeError: func() takes 2 positional arguments but 3 were given
```

[78]:
```
func(0, 1)
```

```
x = 0
 y = 1
```

[81]:
```
def func(x, y): # positional argument  [ formal arguments]
    #    0, 1 # are compulsory arguments in python
    """
        x, y are keywords and are positional arguments

        local variables
    """
    print(f"x = {x} \ny = {y}")
```

[82]:
```
func(y=10, x=20) # keyword arguments
```

```
x = 20
y = 10
```

[88]:
```
func(10, y=30) # k=3, 4
# pos, keyword / default
```

```
x = 10
y = 30
```

[87]:
```
func(y=20, x=20)
```

```
x = 20
y = 20
```

[84]:
```
func(y=20, 10) # ?  # precedence of positional argument is much higher than key␣
↪word arguments
```

```
  File "<ipython-input-84-d60de7acfa07>", line 1
    func(y=20, 10) # ?  # precedence of positional argument is much higher␣
↪than key word arguments
                   ^
SyntaxError: positional argument follows keyword argument
```

[85]:
```
func(20, x=20) # ?
```

```
      ␣
↪---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call␣
↪last)

<ipython-input-85-9b5c08732feb> in <module>
----> 1 func(20, x=20) # ?

TypeError: func() got multiple values for argument 'x'
```

[89]:
```
func(x=20, 10)
```

```
  File "<ipython-input-89-34aa06bc4307>", line 1
```

```
        func(x=20, 10)
                   ^
    SyntaxError: positional argument follows keyword argument
```

any doubt in positional arguments

Default Arguments

[92]:
```
def get_square(x, y=0):
    """ x is positional parameter
        y is default parameter
    """
    print(f"x : {x}")
    print(f"y: {y}")
    print(f"{x}**2 + {y}**2 = {x**2+y**2}")
```

[93]:
```
get_square(5) #
```

```
x : 5
y: 0
5**2 + 0**2 = 25
```

[94]:
```
get_square(5, 10)
```

```
x : 5
y: 10
5**2 + 10**2 = 125
```

[95]:
```
get_square(y=3, x=2)
```

```
x : 2
y: 3
2**2 + 3**2 = 13
```

[96]:
```
get_square(x=20)
```

```
x : 20
y: 0
20**2 + 0**2 = 400
```

[97]:
```
get_square(y=10)
```

```
      ␣
↪---------------------------------------------------------------------------
```

```
TypeError                                 Traceback (most recent call last)

<ipython-input-97-ac8814b6effa> in <module>
----> 1 get_square(y=10)

TypeError: get_square() missing 1 required positional argument: 'x'
```

[ ]:
```
def get_square(x, y=0):
    """ x is positional parameter
        y is default parameter
    """
    print(f"x : {x}")
    print(f"y: {y}")
    print(f"{x}**2 + {y}**2 = {x**2+y**2}")
```

Function Overloading

[99]:
```
from math import sqrt
def area(side1, side2=None, side3=None):
    if side2 == None and side3 == None:
        print("Calculating Area of Circle")
        a = 3.14*side1*side1
    elif side2 != None and side3 == None:
        print("Calculating Area of Rectangle")
        a = side1 * side2
    else:
        print("Calculating Area of Triangle")
        s = (side1 + side2 + side3) / 2
        a = sqrt( s*(s-side1)*(s-side2)*(s-side3) )
    return a
```

[100]:
```
area(100)
```

```
Calculating Area of Circle
```

[100]:
```
31400.0
```

[101]:
```
area(10, 12)
```

```
Calculating Area of Rectangle
```

[101]:
```
120
```

[102]:
```
area(60, 60, 60)
```

```
Calculating Area of Triangle
```

[102]:
```
1558.8457268119896
```

Variable Length positional Argument

[103]:
```
def square_add(a , b):
    return a**2 + b**2
```

[104]:
```
square_add(1, 2)
```

[104]:
```
5
```

[105]:
```
square_add(1)
```

```
␣
↪---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call␣
↪last)

<ipython-input-105-176ca89eab83> in <module>
----> 1 square_add(1)


TypeError: square_add() missing 1 required positional argument: 'b'
```

[106]:
```
square_add(1, 2, 3, 4,5 )
```

```
␣
↪---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call␣
↪last)

<ipython-input-106-1dd8695408df> in <module>
----> 1 square_add(1, 2, 3, 4,5 )


TypeError: square_add() takes 2 positional arguments but 5 were given
```

[108]:
```
print("Hello", "world", "how", "are", "you",)
```

```
Hello world how are you
```

[110]:
```
def func(*args):
    """ x,y = (1, 2,3 ) # tuple unpacking
        print(*[1, 2,3 , 4]) # tuple unpacking

        def func(*args) # tuple packing
    """
    print(type(args))
    print(args)
```

[111]:
```
func()
```

```
<class 'tuple'>
()
```

[112]:
```
func(1)
```

```
<class 'tuple'>
(1,)
```

[113]:
```
func(1, 2)
```

```
<class 'tuple'>
(1, 2)
```

[114]:
```
func(1, 2,3 ,4, 5,6 , 8, )
```

```
<class 'tuple'>
(1, 2, 3, 4, 5, 6, 8)
```

[10]:
```
def squre_add(*args):
    s = 0
    #print(type(args), args)
    for item in args:
        s += item**2
    return s
```

[11]:
```
squre_add()
```

[11]: 0

[12]:
```
squre_add(1)
```

[12]: 1

[13]:
```
squre_add(1, 2)
```

[13]: 5

[14]:
```
squre_add(1, 2,3, 4,5 ,6 ,7 )
```

[14]:
```
140
```

[15]:
```
s = (1*3*5) # expressiong (x*y+c), (1) --> integer
print(s, type(s))
s = (1*3*5, ) # tuple
print(s, type(s))
```

```
15 <class 'int'>
(15,) <class 'tuple'>
```

[ ]:
```

```

[16]:
```
def func(*args):
    print(args, type(args))
```

[17]:
```
func()
```

```
() <class 'tuple'>
```

[18]:
```
func(1, 2, 3, 4)
```

```
(1, 2, 3, 4) <class 'tuple'>
```

[19]:
```
t = (1, 2, 3, 4, 5)
for item in t:
    print(item**2)
```

```
1
4
9
16
25
```

[20]:
```
def square(*args):
    for item in args:
        print(item**2)
```

[21]:
```
square()
```

[22]:
```
square(1)
```

```
1
```

[23]:
```
square(1, 2, 3)
```

```
1
4
9
```

Precedence

positional, default, multiple-length

[24]:
```
def func(pos, default='ha ha ha', *args):
    print("Positional: ", pos)
    print("Default: ", default)
    print("Multiple Length Tuple Argument: ", args)
```

[25]:
```
func(10)
```

```
Positional:  10
Default:  ha ha ha
Multiple Length Tuple Argument:  ()
```

[26]:
```
func(10, 20)
```

```
Positional:  10
Default:  20
Multiple Length Tuple Argument:  ()
```

[27]:
```
func(10, 20, 30, 40, 50, 60)
```

```
Positional:  10
Default:  20
Multiple Length Tuple Argument:  (30, 40, 50, 60)
```

[28]:
```
func(5, default=10, 10, 20, 40)
```

```
  File "<ipython-input-28-4ef86a38a581>", line 1
    func(5, default=10, 10, 20, 40)
                        ^
SyntaxError: positional argument follows keyword argument
```

[29]:
```
func(5, 1, 2, 3, 4, 5, default='sachin')
```

```
␣
↪---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call␣
↪last)
```

```
    <ipython-input-29-60dd265affb9> in <module>
----> 1 func(5, 1, 2, 3, 4, 5, default='sachin')


TypeError: func() got multiple values for argument 'default'
```

[32]:
```
print("hello", "world", sep='\n') # ?
```

```
hello
world
```

[33]:
```
print("Hello", sep='\n', "world")
```

```
  File "<ipython-input-33-5aae9441f139>", line 1
    print("Hello", sep='\n', "world")
                             ^
SyntaxError: positional argument follows keyword argument
```

[39]:
```
def func(pos, *args, k=0):
    print(pos)
    print(args)
    print(k)
```

[40]:
```
func(1, 2, 3, 4, 5, 6, 7)
```

```
1
(2, 3, 4, 5, 6, 7)
0
```

[42]:
```
func(1, 2, 3, 4, 5,6 ,7, 8, 9, k=10)
```

```
1
(2, 3, 4, 5, 6, 7, 8, 9)
10
```

[41]:
```
func(1, 2, 3, 4, 5,6 ,7, 8, 9, sachin=10)
```

```
␣
↪---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call␣
↪last)
```

```
    <ipython-input-41-f3341a4f9932> in <module>
----> 1 func(1, 2, 3, 4, 5,6 ,7, 8, 9, sachin=10)


    TypeError: func() got an unexpected keyword argument 'sachin'
```

[43]:
```
import sys
```

[44]:
```
sys.stdout.write('hello world')
```

```
hello world
```

Write your own print function without print statement

[47]:
```
print('hello', 'world')
```

```
hello world
```

[48]:
```
print("hello", "world", sep='\n')
```

```
hello
world
```

[49]:
```
def myprint(string):
    sys.stdout.write(string)
```

[50]:
```
myprint('hello world')
```

```
hello world
```

[51]:
```
myprint("hello", "world")
```

```
      ␣
↪---------------------------------------------------------------------------

    TypeError                                 Traceback (most recent call␣
↪last)

    <ipython-input-51-575013711367> in <module>
----> 1 myprint("hello", "world")


    TypeError: myprint() takes 1 positional argument but 2 were given
```

[ ]:
```

```