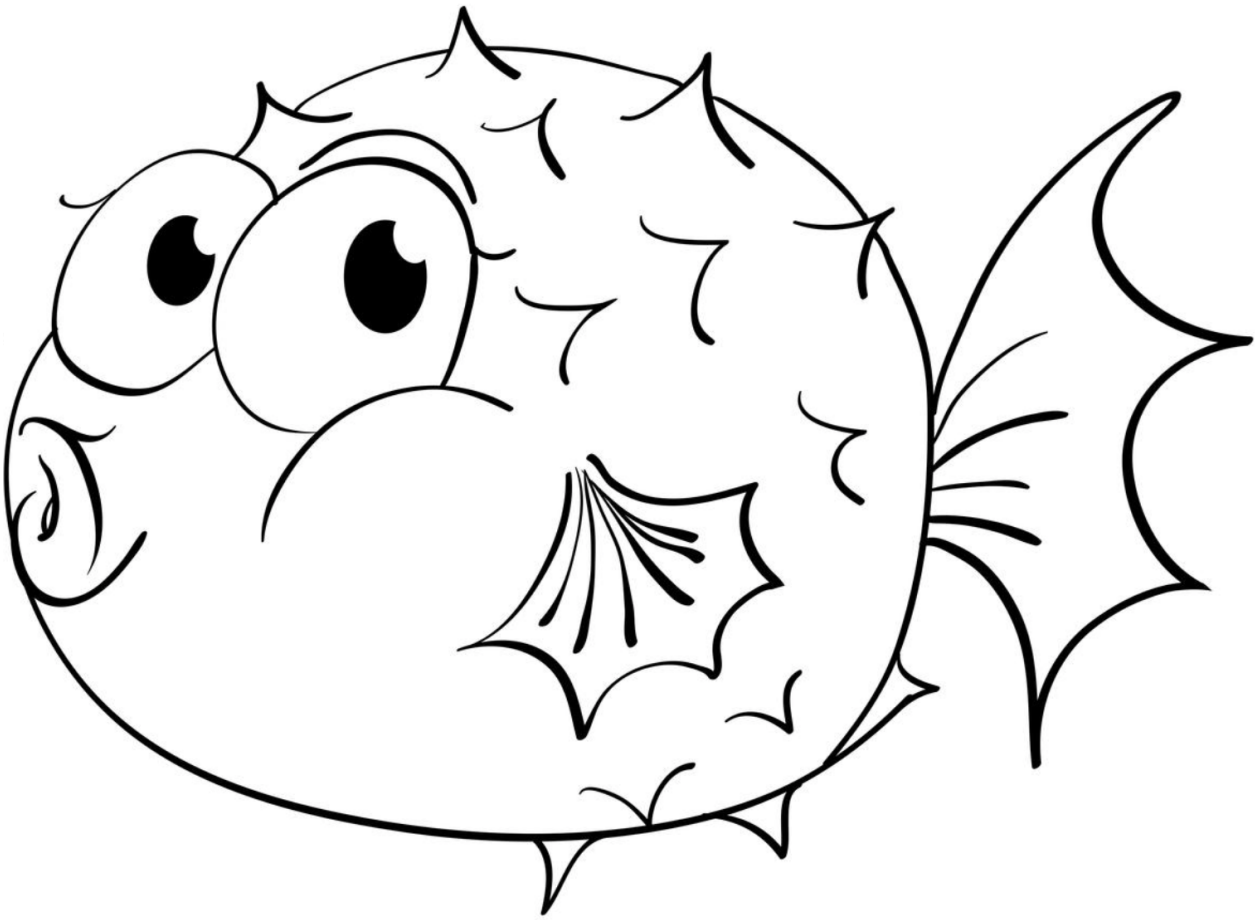
Choose animal

Possibilities are: (elephant, bird, turtle, fish)

In [1]:
```
from PIL import Image

chosenAnimal = 'fish'

orininalImage = Image.open('resources/generatedAnimals/originalDoodle/%s.png' % chosenA
orininalImage
```

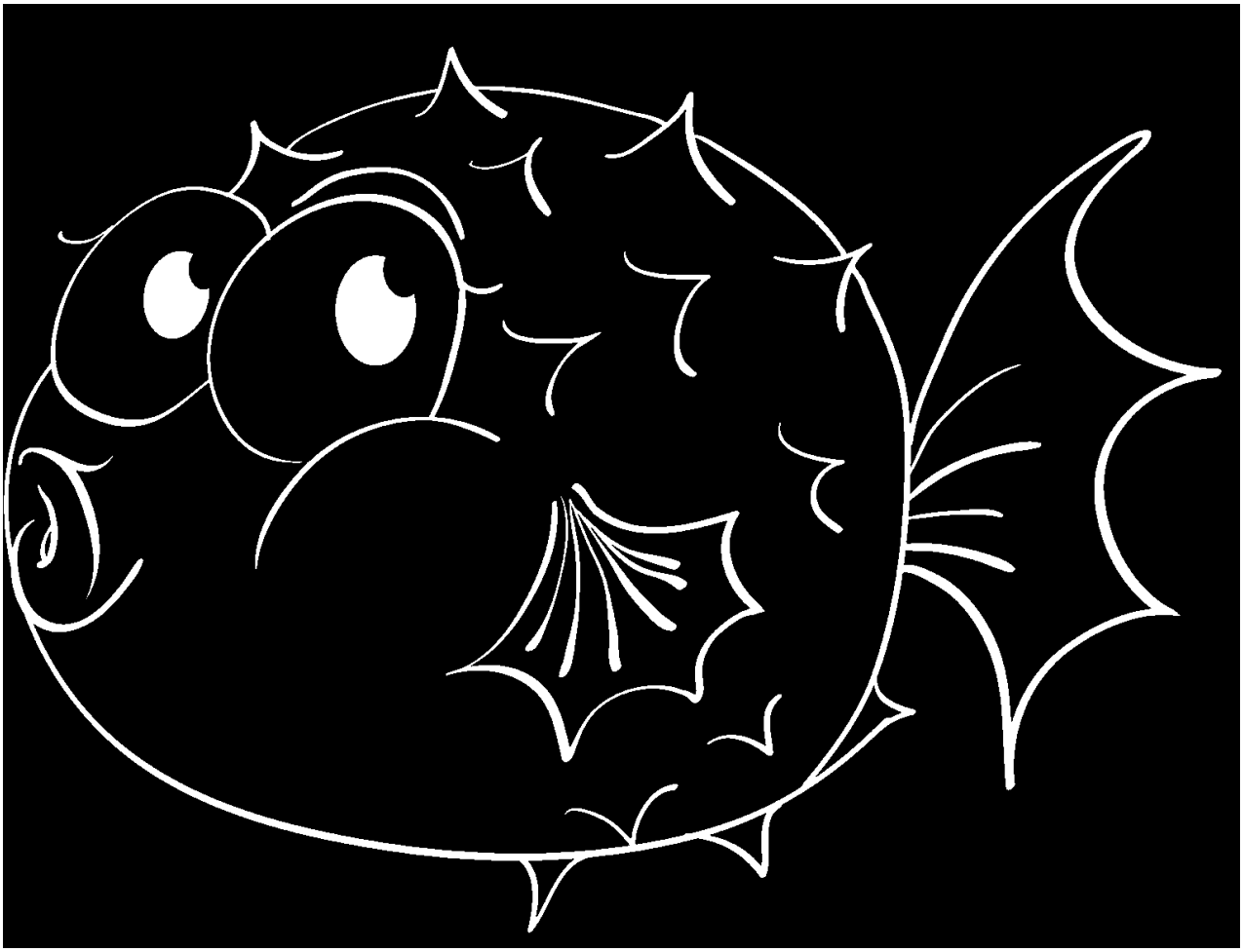
Out[1]:



Extract shape of contour

In [2]:
```
import numpy as np

image = orininalImage.convert('L').point(lambda x : 0 if x > 100 else 255).convert('1')

img = np.rot90(np.asarray(image), k=3)
width, height = img.shape
possiblePositions = list(zip(*np.where(img)))

image
```

Out[2]:



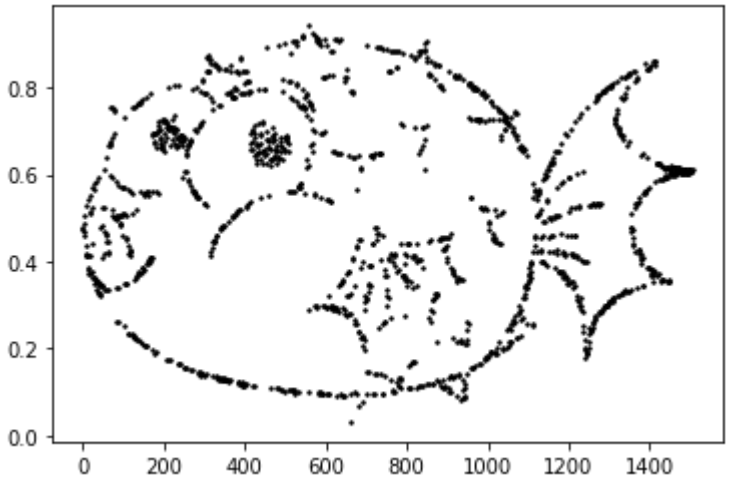
Horizontal scan and select a random value from contour

In [3]:

```
import pandas as pd
import matplotlib.pyplot as plt

sampleAndScale = lambda gr: gr['y'].sample().values[0] / height

data = pd.DataFrame(possiblePositions, columns=['x', 'y']).groupby('x').apply(sampleAnd
data = data.reindex(np.arange(0, width), fill_value=-10)

data = data[data > 0]
numbPoints = len(data)
data.index = range(numbPoints)

plt.scatter(range(numbPoints), data, color='k', s=2)
```

Out[3]:

```
<matplotlib.collections.PathCollection at 0x1821613b370>
```



Import all relevant functions and set value of τ

In [4]:
```
# set tau = 12

from helperFunctions import *
```

In [5]:
```
decimalInitial = findInitialCondition(data)
```

```
significance = 18072 bits ; 5439 digits (base-10) ; ratio = 3.323
```

In [6]:
```
decimalInitial
```

Out[6]:
```
mpf('0.475650620201832731461641499581931445186638081460117055963118390689860789990605653
00729911918235529503166955822130985650378365417179250475697799520167414504517183196682020
51301831675957798427236131097178032104339225907197560993843204288246614067068963075699140
71409998412578565274870082233050402134737303146268965568050012386875496855637691270780960
96910050484973175311882391974517400778219579356433621555401257264440895234191769650160500
88398828660882840320149055470066790846138029441670627763194261845628948993300768699214810
45757725463843424454515346050722693076269172134432631808470491843213591329585656548945970
48033398592764239725505977086670563177614604310826439138972774783648776578016801338687810
57340690576693622636079035175601765070401761541075319223729039811366091135839411378179910
13532929725125651319545384370609447125594511909562105822560724275820790008983306170073190
33105822107960580156259422955540483320134106096822151016853056704241301191029500283412570
45488946354775034159617877734288168948213826665595614756295441199407189902548401001056790
35072975952431048754720476479342742303061861552980852709961488988640574316828705909794000
63287448069642027787461311396603877095121207387731843714576687085618558617687037283921800
27713113334818735174308664812888145417440619351676381930104198518655324222953668733922890
37102903002784346833607892405783272738336712163688121245313747102339992302874318076148980
19659690224035773298239210382260250121318942555088813208732137155124981146559434078679980
56867744391617759192834217623359452727112435577429053461440972299305764634351166336124350
55749932829501022021153613816131909489981452654076498307655897220508120243211714667678100
99501328335286801965447799809701205338633668465041888255743012773751372924282931740892080
05952072180834317640811825306189472869225970703954876075680145656428234424583314158714290
22847520709562117265615729698566034239246662632445536470465757065102711381394982206577420
80249841838743583676261366544292835324558142070970190395780330407548308256372890529856290
82028200924425263984358956853674116103901721337690136963010431476179145545859662705230110
96191862223487277867660927167648853260094989876383734004172833043022170396854206721804130
42524167693151709756823088042927078396709913153143204906942671611095654167844497042778550
87348272864380391414636787042316473166653118996695995890436381340497548784694864342898570
15041365198963943186392901829186071818010590435183376700875695244597200305980097055713810
```

```
434554567166341096555654152984872712832446057248166884201315066946958062307576427453793651190464649755879263384596067079804422160325484642087518224641982748827042664433473799322618481354721868392325804320219425172767787631311071864462075790047999541873480781847865338315370833991009028900982637668935380986028065299392746199147154376581593362147774231139353738126037186291171484319828078275644204412864972956417145633435337873384648919711545216420855532230723860189081936222621580386719849522912217713276181857414119409459827954794898616150510611711043251253350356846282109558004566479859178224182901686808739428095010484110389039403783752362146583318642895740125171372534307686009648352326211053389915529074619520085281820975035182678415744888249297772370409275023586749069402474159417608143733930574008805300233753560218360742035148725746431264376799395025507630410587022820460167361995664946176882221280915167533693793944460172119626061297507109272680566450615978840870689763270696648239756914138782472021267470867211083658481859957682897160489308700572367852270985031505538160112362438384624185162298380113328915619953436626017154636326107027410777410393461962264712756267036326511891718344736918945869505228227904229665790426938056112421851813442005011373130013289571281065122338225832359168480841040051441565870001822688223000353459982026077517866217236115843238373609956868803676203512421793234088802030860366832446073225981073911060518286606271958248468135449459099898542184199572178604911804159464687924120825653058792272848469191765233843051885827023380156555087660933597657436269085785901796092619866787181463027016057079793605585847138228215175180092854480676583072807395050639432285286000054960494559174617278644123159249174705710171484178664265600699326413805388277266764742793057238233349585982335860688019193046819044651044015267574908190017527978449325370686853899527023167888640410290559059933307387087737596215815318812093957133125173800640588169006924310851030946533498693847545482740217064630784380431504666105285946374741575690857690084787923712509841914542262113993165461309531655571886767983286497805712539015475032422937662724369173317364508770063523564304448563353006443211517601139966532875836400817831200926251299299412759702520347604284928242008127885963649200115858829795827709630820969500755242514986863987208982392925955238365117532093903252579881185609172710609695471723131732613490539489889310337068141250148666601707159188515650388392988440033121173075833596805827864701690642647499188071381816054612143400140465869324989170518835322907735671626445544038473221300455291497711688485501789552432939124653292495483722797990977130169451710888738505229495925335952321874402098624883164765032818809320899758975702529850474983952478189285698802015457355715146708324572971643524781443546734345460104673747003012763584311799350762234901214831454048257376585589713351538481710037842488482127690965551267142762869628917688988024963070772207439948206765532726981')
```

In [7]:

```
decodedValues = generateData(decimalInitial, len(data))
```

In [8]:

```
# decodedValues = list(filter(lambda x: x > 1 / 2 ** tau, decodedValues))

colorMap = {'elephant': 'darkblue', 'bird': 'darkorange', 'turtle': 'darkgreen', 'fish'
```

In [9]:

```
plt.figure(figsize=(6, 6))
plt.title(r'$\alpha$ = %.8f $\, \cdots$' % float(decimalInitial), {'size': 20, 'fontnam
# darkgreen ; darkblue ; darkorange ; red
plt.scatter(range(len(decodedValues)), decodedValues, color=colorMap[chosenAnimal], s=5
plt.axis('off');

from matplotlib import pyplot as mp

mp.savefig('resources/generatedAnimals/%s.png' % chosenAnimal, bbox_inches='tight')
```

$\alpha = 0.47565062\ \cdots$