Exercise

May 20, 2023

[55]:
```
#Exercise 1 - Debug code for calculating GC content
#Debug the following code for calculating GC content:

# Calculate the frequency of G & C nucleotides in a sequence

seq = "ATATGCTACTACTCGGCTACG"
gc_content = seq.count(G) + seq.count(C)/len(seq)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [55], in <cell line: 7>()
      1 #Exercise 1 - Debug code for calculating GC content
      2 #Debug the following code for calculating GC content:
      3
      4 # Calculate the frequency of G & C nucleotides in a sequence
      6 seq = "ATATGCTACTACTCGGCTACG"
----> 7 gc_content = seq.count(G) + seq.count(C)/len(seq)

NameError: name 'G' is not defined
```

[ ]:
```
#Correct answer
#G and C were not in ''
seq = "ATATGCTACTACTCGGCTACG"
gc_content = seq.count('G') + seq.count('C')/len(seq)
gc_content
```

[ ]:
```
4.285714285714286
```

[5]:
```
#Exercise 2 - Debug a random bird generator
#Debug the following code for generating random bird common names.

from random import choice
n_birds_to_generate = 10

descriptors = ["Emperor","Red-breasted","Warbling","Vampire","Night",\
              ␣
 ↪"Sea","Greater","Pond","Jungle","Barn","Drab","Lesser","Spotted",\
```

```
               "Northern","Southern","Long-beaked","Crested","Fairy","Bald"]

bird_types = ["Falcon","Merganser","Owl","Eagle","Hawk","Penguin",\
              "Dodo","Gull","Warbler","Fowl","Goose","Hummingbird","Snowcock"]

random_birds = []

for i in range(n_birds_to_generate)
    descriptor = choice(descriptors)

    bird_type = choice(bird_types)

    random_bird = f"{descriptor} {bird_type}"
    random_birds.append(random_bird)
```

```
  Input In [5]
    for i in range(n_birds_to_generate)
                                       ^

SyntaxError: invalid syntax
```

[8]:
```
from random import choice
n_birds_to_generate = 10

descriptors = ["Emperor","Red-breasted","Warbling","Vampire","Night",\
               ␣
 ↪"Sea","Greater","Pond","Jungle","Barn","Drab","Lesser","Spotted",\
               "Northern","Southern","Long-beaked","Crested","Fairy","Bald"]

bird_types = ["Falcon","Merganser","Owl","Eagle","Hawk","Penguin",\
              "Dodo","Gull","Warbler","Fowl","Goose","Hummingbird","Snowcock"]

random_birds = []

for i in range(n_birds_to_generate):
    descriptor = choice(descriptors)

    bird_type = choice(bird_types)

    random_bird = f"{descriptor} {bird_type}"
    random_birds.append(random_bird)
print(random_birds)
#There was no : to begin the for loop
```

```
['Barn Owl', 'Northern Warbler', 'Long-beaked Penguin', 'Night Dodo', 'Spotted
Fowl', 'Warbling Eagle', 'Long-beaked Merganser', 'Drab Snowcock', 'Drab
```

```
Merganser', 'Spotted Snowcock']
```

[15]:
```
#Exercise 3 Debug code for simulating Mendelian Inheritance
#Debug the following code. Note that there is more than one mistake with the␣
 ↪code that you will need to fix.

#This code simules Mendelian inheritance
#Each parent has two alleles or genetic variants: A and a

#Each gamete (sperm or egg) gets one random allele from
#the parent that produced that gamete

#The offspring genotype is a combination of these

from random import choice

maternal_alleles = ["A","a"]
paternal_alleles = ["A","a"]

egg_allele = choice(maternal_alleles
sperm_allele = choice(paternal_alleles)

offspring_genotype = sorted(egg_allele + sperm_alele)
print(f"The genotype of the offspring is {offspring_genotype}")
```

```
  Input In [15]
    sperm_allele = choice(paternal_alleles)
    ^
SyntaxError: invalid syntax
```

[12]:
```
from random import choice

maternal_alleles = ["A","a"]
paternal_alleles = ["A","a"]

egg_allele = choice(maternal_alleles)
sperm_allele = choice(paternal_alleles)

offspring_genotype = sorted(egg_allele + sperm_allele)
print(f"The genotype of the offspring is {offspring_genotype}")
#There was a missing parenthesis after (maternal_alleles
#There was also a type for alleles
```

```
The genotype of the offspring is ['A', 'A']
```

[ ]:
```
Exercise 4 - Debug broken code for outputting genome analysis results
Debug the following code that merges the results of a genomic analysis into a␣
 ↪tab-delimited output line:

#Imagine we'd calculated several parameters for the genome
header_fields = ["Genus","Species","Strain","Chromosome Type","Genome␣
 ↪Length","Coding Regions","GC content"]
header_line = "\t".join(header_fields)+"\n"
print(header_line)


#Most commonly, this type of code would be inside a for loop
#where we were analyzing many genomes, and generating one line of
#results per genome analyzed.
#(here I just hard-code the results for simplicity).

gc_content = 57.0
genome_length_nt = 4195195
chromosome_type = "circular"
coding_regions = 4276

#Get the genus name and species name from the full strain id
strain_id = "Bacillus subtilis SZMC 6179J"
genus,species,strain_id_part1,strain_id_part2 = strain_id.split()

result_fields =␣
 ↪[genus,species,strain_id,chromosome_type,genome_length_nt,coding_regions,gc_content]
result_line = "\t".join(result_fields) + "\n"
print(result_line)

#We would go on to open a results file
#and write the results to it.
```

[24]:
```
header_fields = ["Genus","Species","Strain","Chromosome Type","Genome␣
 ↪Length","Coding Regions","GC content"]
header_line = '\t'.join(header_fields)+"\n"
print(header_line)

gc_content = 57.0
genome_length_nt = 4195195
chromosome_type = "circular"
coding_regions = 4276

strain_id = "Bacillus subtilis SZMC 6179J"
genus,species,strain_id_part1,strain_id_part2 = strain_id.split()

result_fields = [strain_id]
```

```
result_line = "\t".join(result_fields) + "\n"
print(result_line)

#Put strain_id in the result_field =[] instead of all of the headers
```

```
Genus   Species Strain  Chromosome Type Genome Length   Coding Regions  GC
content

Bacillus subtilis SZMC 6179J
```

[47]:
```
#Exercise 5 - Debug code for mapping scientific names into common taxon names
#Hint: there are 3 mistakes with the code that you'll need to fix. The first␣
 ↪two are straightforward. Fix them first, and you will encounter the 3rd bug,␣
 ↪which is a bit more subtle.

#Look up an informal, non-scientific common name for each species
species = ["Homo sapiens","Gallus gallus","Bacillus thuringiensis",\
           "Bacillus subtilis SZMC 6179J","Porites asteroides","Acropora␣
 ↪palmata"]

common_name_map = {"Homo":"Mammal","Gallus":"Bird","Bacillus":"Bacterium",\
                   "Porites":"Stony coral","Acropora":"Stony coral"}

for binomial_name in species:
    genus,species = binomial_name.split(maxsplit=2)
    common_taxon_name = common_name_map[genus]
    print(f"{binomial_name} is a {common_taxon_name}")
```

```
Homo sapiens is a Mammal
Gallus gallus is a Bird
Bacillus thuringiensis is a Bacterium
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [47], in <cell line: 11>()
      8 common_name_map = {"Homo":"Mammal","Gallus":"Bird","Bacillus":
 ↪"Bacterium",\
      9                    "Porites":"Stony coral","Acropora":"Stony coral"}
     11 for binomial_name in species:
---> 12     genus,species = binomial_name.split(maxsplit=2)
     13     common_taxon_name = common_name_map[genus]
     14     print(f"{binomial_name} is a {common_taxon_name}")

ValueError: too many values to unpack (expected 2)
```

[53]:
```
species = ["Homo sapiens","Gallus gallus","Bacillus thuringiensis",\
           "Bacillus subtilis SZMC 6179J","Porites asteroides","Acropora␣
 ↪palmata"]

common_name_map = {"Homo":"Mammal","Gallus":"Bird","Bacillus":"Bacterium",\
                   "Porites":"Stony coral","Acropora":"Stony coral"}

for binomial_name in species:
    genus,species = binomial_name.split(maxsplit=1)
    common_taxon_name = common_name_map[genus]
    print(f"{binomial_name} is a {common_taxon_name}")
```

```
Homo sapiens is a Mammal
Gallus gallus is a Bird
Bacillus thuringiensis is a Bacterium
Bacillus subtilis SZMC 6179J is a Bacterium
Porites asteroides is a Stony coral
Acropora palmata is a Stony coral
```

[ ]: