COMP 204

Review and final exam preparation

Yue Li

CSUS review session and helpdesk hours

CSUS has planned a COMP 204 review session run by Helpdesk tutors for **Friday, April 26th from 6pm-9pm in MAASS 112**

If this time does not work for you, they are also welcome to drop by the Helpdesk (Trottier 3090) from 10am-5pm on any weekday until classes end with any questions you may have.

Final exam info

- Date: **April 30, 6:30-9:30 PM**; Location: TBD
- Weight: 35% of your final grade (or 55% if better than midterm grade for students who opted the second non-programming midterm assignment option)
- Closed book but 8.5 x 11 double-sided crib sheet allowed.
- Questions:
  - 9 multiple choice questions (total 27%). Answer on Scantron (**not on exam**). Follow instructions for each questions: For some questions you need to indicate the only ONE correct answer. For other questions you need to indicate ALL correct answers.
  - Answer the rest of the questions directly on exam
  - 8 short answer questions (7 questions each worth 4 points and 1 question worth 5 points) (total: 33%).
  - 1 bonus short answer question worth 5 point.
  - 4 long answer questions (10 point per question; total: 40%).

Final exam content

Main materials that are covered in the final exam include:

- Basics: functions, loops, variables, data types (string, list, tuple, dictionary, sets), difference between pass by copy and pass by memory addresses
- Algorithms: Searching (linear and binary search) and sorting (insertion and selection sort)
- Pattern searching by string indexing and regular expression (simple ones)
- Object oriented programming: class, attributes, class inheritance, class methods
- BioPython sequence handling covered in class (I will remind you what the methods are in the exam)
- Machine learning: know what supervised, unsupervised, reinforcement learning are, problems they can solve, TPR, FPR, overfitting, cross-validation, ROC, decision trees
- Image processing: basic understanding of going from a pixel in the image to numpy ndarray
- What to memorize? Nothing. Use cribsheet to note the

Preparing for the final exam

How best to prepare for the exam:

- Practice, practice, practice.
- Review all lecture notes, assignment solutions, midterm solutions
- Practice on the problems we've posted on MyCourses-Content
- Attend CSUS review session:
- Come to my office hours:
  - Wednesday: 11:30-12:30

Functions

What prints out?

```
1  def myfun(x, y):
2      x = x + 1
3      y = y + 1
4      return x + y
5
6  x = 0
7  y = 1
8  z = myfun( myfun(x,y), x)
9  print(z)
```

Functions (pass by memory address)

What prints out?

```
1  def myfun(x, y):
2      x[0] = x[0] + 1
3      y[0] = y[0] + 1
4      return [x[0] + y[0]]
5
6  x = [0]
7  y = [1]
8  z = myfun( myfun(x,y), x)
9  print(z)
```

Linear and binary search

How to search number 9 in this list by linear search and binary search? [2,5,7,9,10]

Selection and insertion sort

How to sort the following list by selection sort and insertion sort?
[2,10,5,9,7]

Sequence alignment (A2)

Given match score $+3$, mismatch score -2, gap score -1. What's the similarity score between sequence GGC with sequence GTC?

List comprehension

Convert the following for loop into list comprehension with one line of code:

```
1  x = []
2  for i in range(5):
3      x.append(-2*i)
```

String pattern matching

Choose ALL of the correct boolean expression(s) that will match with a string s that starts with AUG and ends with stop codon UAG, UAA, and UGA

A
```
s[0:3]=="AUG" and s[-3:] in ["UAG", "UAA", "UGA"]
```

B
```
s[0:3]=="AUG" and s[len(s)-3:len(s)] in ["UAG",
"UAA", "UGA"]
```

C
```
re.research("^AUG.*(UAG|UAA|UGA)$", s)
```

D
```
re.research("AUG.*(UAG|UAA|UGA)", s)
```

E
```
re.research("^AUG.*[UAG|UAA|UGA]$", s)
```

F
```
s == "^AUG.*(UAG|UAA|UGA)$"
```

Object oriented programming: attributes

What are attributes in `Myclass`

```
1  class MyBus:
2      def __init__(self, stationID, passengers):
3          self.s = stationID
4          self.p = passengers
5          terminal = 0
```

Object oriented programming: methods

What prints out?

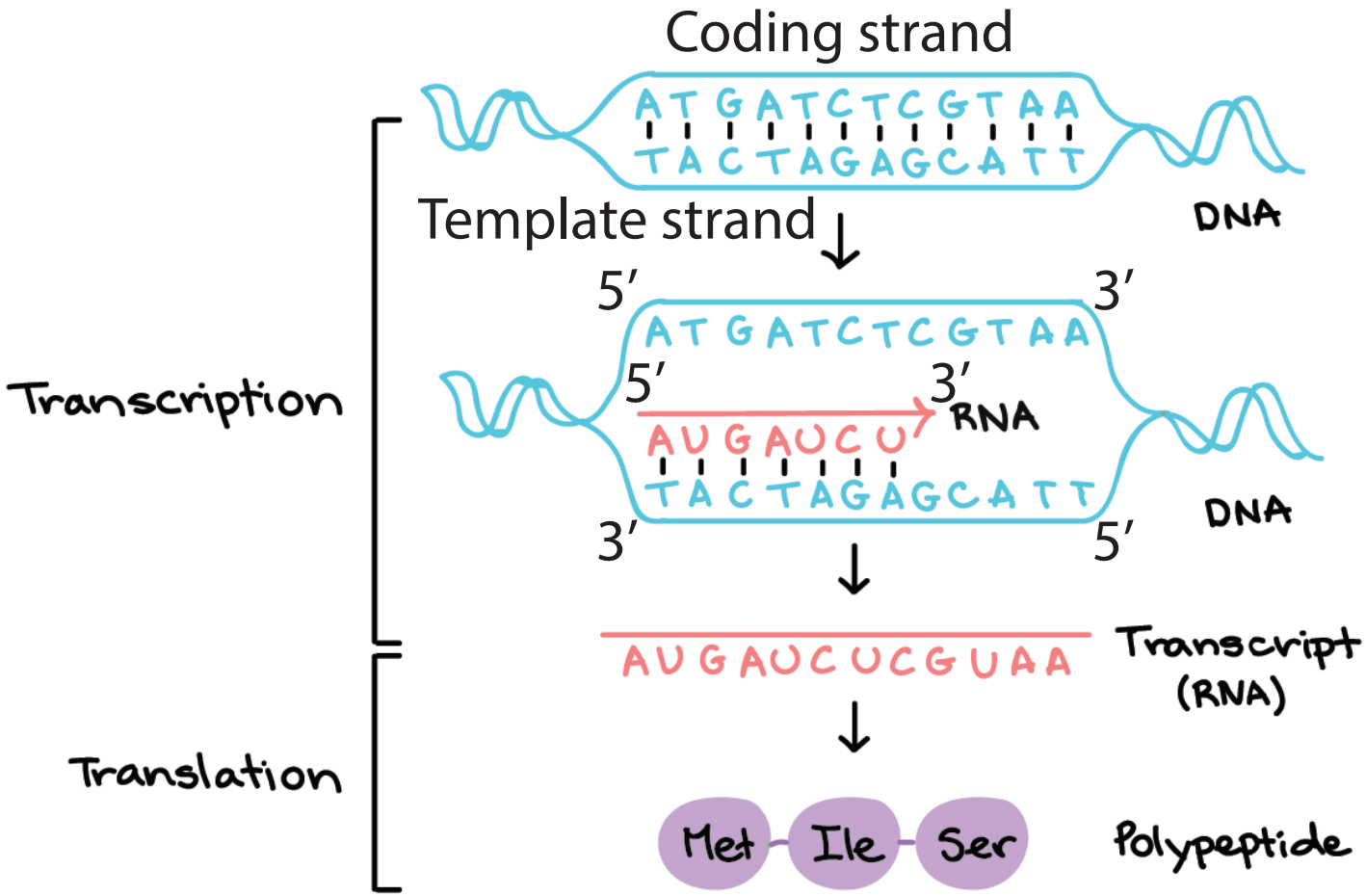
```
1  class Animal:
2      def __init__(self, age):
3          self.age = 0
4          self.claws=0
5      def grow(self):
6          self.age += 1
7          claws = self.claws + 1
8  animal = Animal()
9  animal.grow()
10 print(animal.age)
11 print(animal.claws)
```

Object oriented programming: class inheritance

What prints out?

```
1  class Animal():
2      def __init__(self):
3          self.age = 0
4          self.claws=0
5      def grow(self):
6          self.age += 1
7          claws = self.claws + 1
8  class Predator(Animal):
9      def __init__(self):
10         Animal.__init__(self)
11         self.horns = 0
12         self.eyes = 0
13     def grow(self):
14         Animal.grow(self)
15         self.horns += 1
16         eyes = self.eyes + 1
17
18 pred = Predator()
19 pred.grow()
20 print(pred.claws, pred.age, pred.horns, pred.eyes)
```

Central dogma



Every three DNA letters (i.e., codon) code for an amino acid

Transcription

Given a DNA string as the template strand say 5'-AGATCAT-3', write a function called `transcribe(dna)` that returns the transcribed RNA sequence (i.e., AUGAUCU)

```
1  def transribe(dna):
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19 return rna
```

Translation: codon table

1st base	2nd base								3rd base
	T		C		A		G		
T	TTT	(Phe/F) Phenylalanine	TCT	(Ser/S) Serine	TAT	(Tyr/Y) Tyrosine	TGT	(Cys/C) Cysteine	T
	TTC		TCC		TAC		TGC		C
	TTA	(Leu/L) Leucine	TCA		TAA	Stop (*Ochre*)[B]	TGA	Stop (*Opal*)[B]	A
	TTG[A]		TCG		TAG	Stop (*Amber*)[B]	TGG	(Trp/W) Tryptophan	G
C	CTT		CCT	(Pro/P) Proline	CAT	(His/H) Histidine	CGT	(Arg/R) Arginine	T
	CTC		CCC		CAC		CGC		C
	CTA		CCA		CAA	(Gln/Q) Glutamine	CGA		A
	CTG[A]		CCG		CAG		CGG		G
A	ATT	(Ile/I) Isoleucine	ACT	(Thr/T) Threonine	AAT	(Asn/N) Asparagine	AGT	(Ser/S) Serine	T
	ATC		ACC		AAC		AGC		C
	ATA		ACA		AAA	(Lys/K) Lysine	AGA	(Arg/R) Arginine	A
	ATG[A]	(Met/M) Methionine	ACG		AAG		AGG		G
G	GTT	(Val/V) Valine	GCT	(Ala/A) Alanine	GAT	(Asp/D) Aspartic acid	GGT	(Gly/G) Glycine	T
	GTC		GCC		GAC		GGC		C
	GTA		GCA		GAA	(Glu/E) Glutamic acid	GGA		A
	GTG		GCG		GAG		GGG		G

Not all mutations lead to a different animo acid

e.g., GCT and GCC both code for Alanine

Translation

Assume the codon table is provided to you as a dictionary `ct` with key as the 3-letter DNA string and value as the amino acid, write a function that translates an RNA into the amino acid sequence

```
1  def translate(rna, ct):
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19 return aa
```

Get candidate cancer cell

Suppose we obtain a collection of unknown cells from a patient. Each cell is a Cell object. We are provided with a function called `cancer_cell_score(cell)` that gives a cancer score to the unknown cell. Write a function that return the highest scoring cell.

```
1  def get_candidate_cancer_cell(unknown_cells):
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17 return ccc # candidate cancer cell
```

Average number of cell-type-specific cells per patient

Suppose we obtain a collection of single cells with known cell types from a set of cancer patient blood samples. We stored this as a dictionary with key as patient ID and values as cell types. For example (not real data):

```
singlecells = {
"patient0":["B-Cell", "B-Cell", "T-Cell", "Neutrophils"],
"patient1":["T-Cell", "T-Cell", "Neutrophils"],
"patient2":["B-Cell", "Neutrophils", "Cytokines"],
"patient3":["B-Cell", "B-Cell", "Cytokines"]}
```

Write a function to calculate the average number of cell-type-specific cells per patient sample.

next page

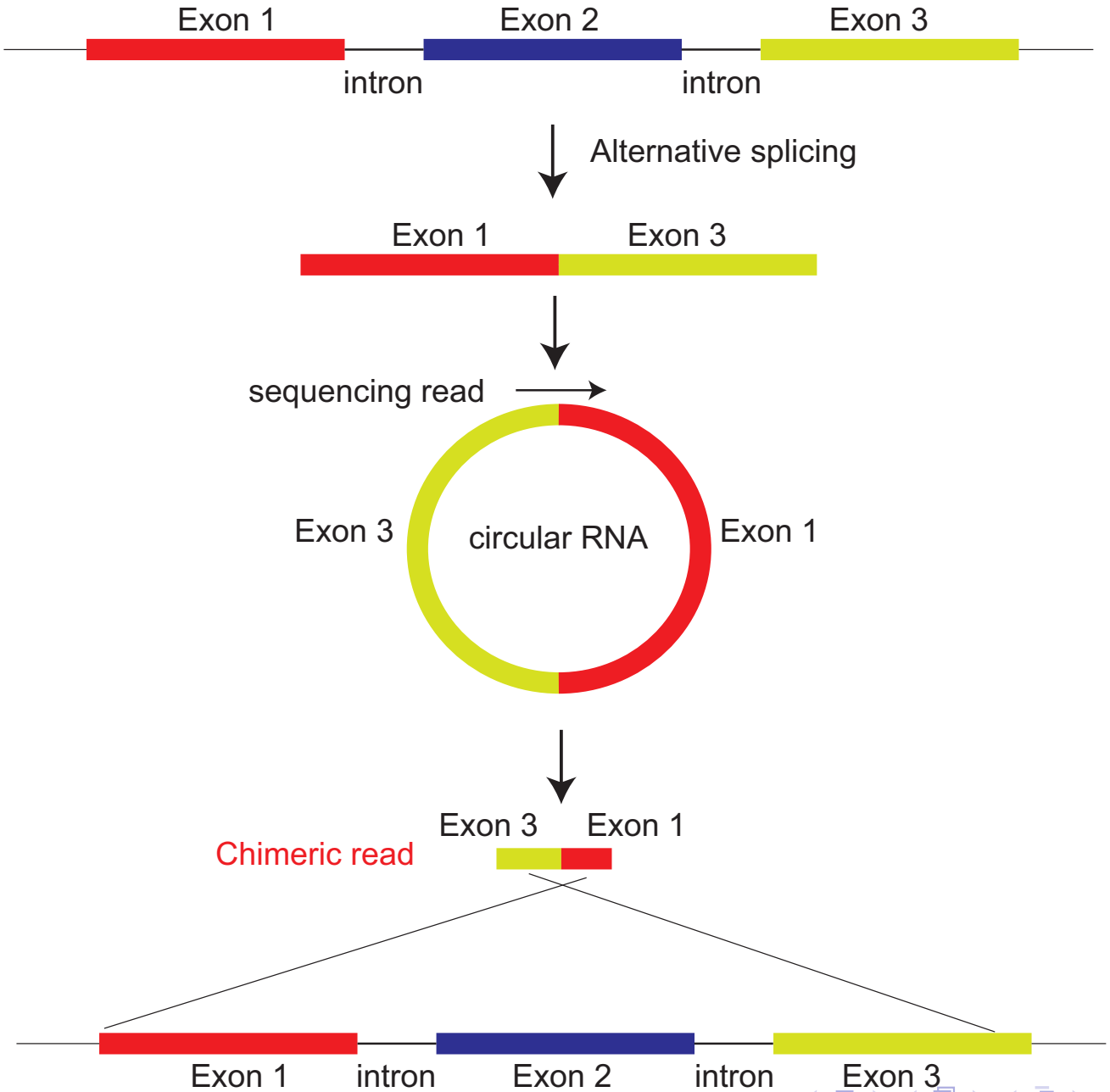
```
1  def average_cell_types(singlecells):
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23 return avg
```

Find all palindromic DNA sequences

You are provided with `is_pal` function that determine whether a DNA sequence is palindromic. The reverse complement of a palindromic sequence is the same sequence. Write a function that returns a list of all of the palindromic sequences in a long DNA sequence. Also print out the longest palindromic sequence.

```
1  def find_all_palindrome(dna):
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17      return all_pals
```

Circular RNA and chimeric reads



Determine chimeric read

Write a function that determines whether a read is a chimeric read. The function takes the candidate read sequence and an ordered list of exon sequences as they appear in the genome (i.e., exon1 is upstream of exon2, and exon2 is upstream of exon3, and so on). Assume exon with the longest match to the first half of the read is the exon that the read truly come from.

```
1  def is_chimeric(read, exons):
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16 return is_chim
```